Computing without Mice and Keyboards: Text and Graphic Input Devices for Mobile Computing

Robert Rosenberg

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

1998

Abstract

Computers have been progressively becoming smaller and more mobile. The past few years have seen computers evolve from the desktop to the notebook to current handheld systems. Each new device is more portable and less cumbersome than last. The next step in this progression is the wearable computer: a computer consisting of parts small enough to be worn as clothing or accessories. However, the portability gained from a wearable computer is lost if the input devices are no smaller than existing ones. The main concept behind this thesis is that, in order for wearable computers to be effective, they need to throw away the mouse and keyboard interfaces in favour of a new style of input. These new interfaces should be designed to be easily transportable, quickly accessible, easily usable in a variety of environments, and minimise interference with "real world" interactions. In order to test this theory we have built a graphic (Biofeedback Pointer) and a text input device (Chording Glove) based on these design principles. These devices are designed to work in tandem, with the Chording Glove's text input complementing the Biofeedback Pointer's graphic input.

The Chording Glove is a text input device made from mounting the keys of a chord keyboard onto the fingers of a glove. Chords are made by pressing the fingers against a firm surface. After 11 hours of use, the average input speed on the Chording Glove was 16.8wpm, with an error rate of 17.4%. The Chording Glove's entire 97 character keymap takes, on the average, 80 minutes to learn, with users needing to look up chords around 0.5% of the time after 6 hours of use. There is evidence of keymap retention after significant absence from the device, but this remains to be proven. Evidence also shows that there is no significant difference in performance on the Chording Glove while standing or sitting, implying that the device is as mobile as intended.

The Biofeedback Pointer is a graphic input device controlled by wrist motion. Moving the wrist causes the pointer to move in that direction. The pointer works by measuring the Electromyograms (EMGs) of four of the main muscles used to move the wrist. This data is interpreted by a neural network which is trained for each user. The network takes about half a minute to learn to recognise the user's EMG signals. After the training the user learns to fine tune the pointer control through a biofeedback process. Fitts' Law was used to compare the performance of the Biofeedback Pointer with a standard mouse. The average index of performance of the Biofeedback Pointer was found to be 1.06, while the mouse was 7.10. This is about half of the lowest performance of a common graphic input device. Using a more sophisticated neural network or better training may improve the device to a more comparable level.

These results show that the design criteria established here for mobile input devices can be used to build workable computer interfaces which encompass both graphic and text input. Hence input devices are shown to exist which are transportable and usable, thereby making wearable computers a viable technology.

Acknowledgements

First off, I would like to give a special thanks to my supervisor Mel Slater. His constant advice and support throughout the entire course of my degree are very much appreciated.

I would also like to thank Mike Craggs for building the Biofeedback Pointer hardware and by being an invaluable resource in helping me understand the practical side of bioelectric measurement. Derek Coppen and Tim Barnes also deserve credit for their help in designing and building the Chording Glove's circuitry.

I am extremely grateful to the members of the Virtual Environments and Graphics Group at UCL for just being there to answer questions, offer advice, or just to bounce random ideas off. The subjects of the experiments deserve thanks for volunteering to help. The data they provided were essential to this thesis.

Another special thanks go to Anu Hyttinen who one day asked, "Is there anything I can do to help?" Except for a few computer-generated pictures, all the illustrations in this thesis were drawn by Anu. I am indebted to her continued help despite the fact that every time she thought she was finished, I would find I needed "just one more" drawing.

Lastly, I would like to thank my parents for their unwavering encouragement and support.

A final note: The work for this thesis began at Queen Mary and Westfield College. This is where the bulk of the Chording Glove research, including the experiment, was performed. Due to a transfer in 1996, the remainder of the research was carried out at University College London.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Limitations of Mobile Computing

The past two decades have seen continuous improvements in computer power and miniaturisation. In the early 1990's this technology reached the point where it was possible to fit all the components of a desktop personal computer into a small and lightweight box, while still retaining most of the speed and power of the desktop model. When this style of computer is designed to be portable and easily used outside of an office, it is called a *mobile computer.* The extent of the mobility of these machines are determined mostly by their size and weight, and to a certain degree by their shape.

The portability of the older mobile computers was limited by the size of the internal components, such as disk drives and the CPU. As the technology progressed, the size of the components was no longer the limiting factor. The computer itself could be the size of a small keyboard, but it could not become any smaller. It could become thinner and lighter, but any further reduction in size would make the keyboard interface more difficult to use. The requirement of a keyboard interface imposed limitations on the situations in which these computers could be used. The mobile computers were small and light enough to be carried anywhere, but operating them required the user to sit down and place the device on their lap, a table, or a similar surface. Mobile computers had reached the point where their input devices were preventing them from becoming any more portable.

In recent years handwriting recognition has come of age. Systems have been built with more than 97% accuracy (MacKenzie & Zhang, 1997). This has enabled a new spurt in computer miniaturisation. Computers are again partly limited by the size of the components, but they are also limited by the stylus input. The stylus needs a flat surface to write, which forces the computer into a notepad-like shape.

The usefulness of a mobile computer is dependent on how well it functions in a *mobile environment*, a place where there are no standard surroundings and the facilities at hand (i.e. power supply, resting surfaces, etc) cannot be guaranteed. A notebook computer is portable, but not especially usable in a mobile environment, because of the afore mentioned limitations imposed by the keyboard. A handheld computer is portable and usable in mobile environments, but is limited in size and shape by the stylus-and-notepad interface. The next step in computer miniaturisation is the wearable computer. A *wearable computer* is a computer made up of parts small enough to be worn on or as clothing. The monitor is often worn like glasses (although often much bulkier). The CPU is usually attached to a belt, and the input devices are located on the CPU or mounted somewhere on the body. This frees the user from needing to carry the computer, making it even more portable than a handheld computer. The problem with wearable computers is that no common input devices are mobile enough to easily interact with them.

1.2 Designing Mobile Input Devices

Since no existing input device is up to the task, in order to make wearable computers a feasible option, we need to find a new style of input device which does not put any constraints on the overall shape or size of the computer. With this primary requirement, we can derive a set of basic design goals by considering four factors. These factors are: the environment the devices will be used in, the kind of person we expect to use the devices, the tasks the devices will be used for, and the hardware the devices will interact with.

These devices should be designed to be used in a mobile environment. In this setting we cannot know what surroundings there will be, so the devices must be able to operate without requiring the user to be in a specific position, such as sitting down, nor require any surfaces on which to brace the device. This can range from hostile environments, such as underwater to more everyday uses such as working outside or while commuting.

These devices must accommodate the type of person who currently uses mobile computers, and also those who would benefit from the additional mobility of a wearable computer. The former type of user is the *nomadic user*, who takes their computer from place to place, but only uses it while not moving. This sort of user is easy to accommodate, even with existing systems. The latter type of user is the *continuous user*, one who uses their computer while moving, and, in all likelihood, performing non-computer tasks simultaneously. These people are more difficult to accommodate, since they need to access information on a computer quickly, easily, and in a way that does not interfere with their performance of real-world tasks. These users need a mobile interface which is invisible enough that they can switch between computer and real-world tasks at will and with detriment to neither.

The expected tasks for these devices are also similar to the normal mobile PC usage. Handheld computers are commonly used for personal information management. Notebook computers in a mobile environment are often used for some of the simpler tasks performed on a desktop computer. Consequently, the new devices should be designed to be capable of operating a windows-style interface as well as being able to handle general text input, including basic composition.

Finally, the new input devices should be designed to run using existing wearable computer systems. This means that they can be attached to the computer in a standard way, i.e. via keyboard, mouse, serial or parallel ports or by using PCMCIA cards. They should also be compatible with existing operating systems, preferably windows-based. The input devices should not require excessive computing power to run. These devices should be able to be used with existing computer speeds and memory, i.e. requiring no more than 133MHz or 16Mb RAM.

These basic design goals are the minimum requirements we have for a mobile input device. In order to determine how to design and build a mobile input device we need to examine existing devices and extrapolate from their design principles to arrive at a set of guidelines for creating mobile interfaces. This is the subject of the next chapter. The remaining sections of this chapter outline the objectives, scope, main contributions and structure of this thesis.

1.3 Objectives

This thesis examines the important factors in the design of text and graphic input devices and reconciles them with the limitations imposed by a mobile environment. Using these as guidelines we have developed new input devices, designed specifically to be unobtrusive, easy to use, and portable. These devices are the Chording Glove and the Biofeedback Pointer.

The Chording Glove is a chord keyboard, with the keys mounted directly on the fingers of a glove. Instead of the fingers pressing buttons on a board, the buttons are on the fingers and can be pressed against any hard surface. The Biofeedback Pointer uses the bioelectric impulses generated by moving the wrist to control a two dimensional graphic pointer. Moving the wrist horizontally or vertically causes the computer pointer to move in that direction.

The performance of each device was examined empirically. The goal of this is to show that the concept behind the devices is feasible and that devices based on one or both of these could work as input devices for mobile computers.

1.4 Scope

Our primary concern with mobile computing is graphic and text input and how these devices affect the portability of a computer. We will not be considering the details of mobile computers themselves, only the design of the input devices. Mobile computers are only considered for their size and shape. The internal workings of the computer are irrelevant to our discussion. We will merely assume that they work.

The input devices will be considered primarily for their hardware design. Software design will be limited only to where it directly involves use of the device, such as pattern recognition or motion scaling. The graphic user interfaces will all be assumed to be based on the standard windows interface. As a consequence, the input devices analysed will either be two dimensional pointing devices or devices for entering written English text. Three dimensional interfaces and speech-based operating systems are discussed briefly, but only for comparison. Any further analysis of such devices is beyond the scope of this text.

A fair amount of understanding of physiology is necessary to fully appreciate the biological systems discussed in this thesis. In order to simplify matters we will limit discussion to bioelectric signals which can be detected by surface measurements. In addition, only those bioelectric signals which are involved with movement will be considered. This should limit the amount of necessary background to allow a fair discussion of the processes without getting bogged down by details.

The devices introduced in this thesis are for proof of concept. The intention is not to make a marketable device, but to investigate the uses of such a device and how it could be made more efficient.

1.5 Contributions

The following are the three main contribution of this thesis.

1. Several text and graphic input devices are reviewed in order to evaluate their design, determine their portability, and compare their performance. The current types of mobile computers are discussed, concentrating on how their shape and input devices limit their mobility. These factors are used to determine the most portable styles of input devices.

2. The results of the input device analysis are used to design two new input devices with the intent to maximise mobility. The Chording Glove is introduced as a wearable text input device. Pressure sensitive triggers mounted on the fingertips of the glove replace the keys on a chord keyboard. Chords are generated by pressing the fingers against any solid surface. In creating the chord keymap for the Chording Glove, a method for generating a chord keymap was developed. This can be adapted to create a keymap for any chording system, or develop new, specialised keymaps for

the Chording Glove. In addition, a theoretical method for comparing different chord keymaps is introduced.

3. The Biofeedback Pointer is introduced as a new portable graphic input device which translates wrist movements into motion of a two-dimensional pointer. This is done by sensing and analysing the bioelectric signals generated from the muscles which control the wrist. The methods used to control the Biofeedback Pointer can easily be adapted to other muscles or other graphic input styles (such as navigation).

1.6 Organisation

This thesis is organised into seven chapters, the contents of which are as follows:

Chapter two provides an overview of existing text and graphic input devices in order to determine their portability, performance, and to generate a set of design criteria to assist in creating new input devices. Special attention is paid to the use of Fitts' Law as a tool for determining the performance of graphic input devices and to the physiology necessary to understand bioelectric computer interfaces. The various styles of mobile computers are discussed and four factors are introduced by which their input devices can be rated for portability.

Chapter three introduces the Chording Glove and the Biofeedback Pointer. Details of the development phases of the devices are given, along with the final hardware and software designs. Potential uses each of the devices are discussed, both separately and as a unified system.

Chapter four describes the experiment used to determine the input speed, learning rate, error rate and other performance features of the Chording Glove.

Chapter five describes the experiment to determine the relative performance of the Biofeedback Pointer and the mouse. In addition, the neural networks used to analyse the subjects' bioelectric signals are examined and compared.

Chapter six considers the potential sources of error in the experiments in an attempt to ascertain any effects on the validity of the experiments. The second part of the chapter discusses areas for future research and experimentation.

Chapter seven provides a summary of the thesis with a discussion of the contributions and their implications.

Chapter 2

Input Devices and Mobile Computing

2.1 Introduction

For many mobile computers, there is little difference between their text input and the methods for text input on the typewriters of over a hundred years ago. The keyboards used by many notebook computers have more keys and functions than the original typewriters, but the layout and interaction methods are essentially the same, despite a century of progress. On the other hand, many handheld computers, like the PalmPilot, have forsaken the keyboard in favour of the much more portable stylus input, but this introduces an entirely new set of limitations.

Unlike text input devices, pointers have varied widely in the past decade alone. In many mobile systems mice have been replaced by trackballs, trackpads and other novel and more mobile pointers. All of these new devices are small, but still large enough to impose limits on the size of the device.

New input devices need to be built which are more portable and can free the computer from their constraints on its size and shape, allowing it to take a more comfortable and efficient form. In order to design these new devices, we need to review existing and historical input devices focusing on their design criteria, the extent of their portability, and their performance. By observing the factors which went in to building these devices we intend to develop an encompassing set of design criteria that we can use to create new devices in order to make them as ergonomic and efficient as possible. The portability of existing devices is examined in order to determine what the limiting factors are. By finding these constraints we hope to avoid such limitations in the new devices. Examining the performance of existing devices should provide a basis for comparison by which we can judge the new devices.

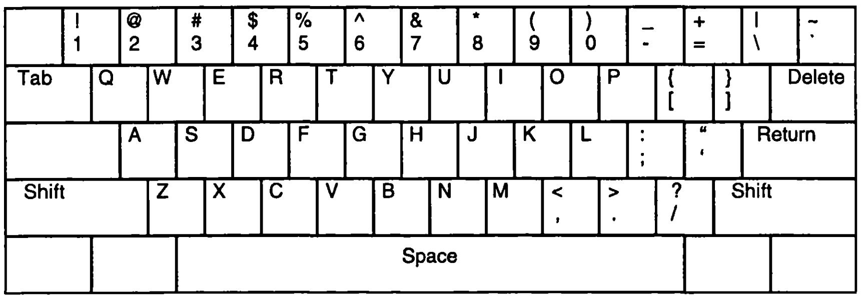
2.2 An Overview of Text Input

2.2.1 Conventional and Alternative Keyboards

The keyboard is by far the most common text input for computers. The momentum built up from over a century of use has guaranteed its dominance despite the problems with both the layout and the basic keyboard shape. Over the years, several alternatives to the QWERTY keyboard have been developed which claim to solve some its problems. The first alternative keyboards simply redesigned the key layout for maximum touch-typing efficiency. More recent ones have reshaped the keyboard in an attempt to make it more comfortable to use.

The QWERTY Keyboard

Figure 2.1: The QWERTY layout


! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = | \ ~ `
Tab Q W E R T Y U I O P { [}] Delete
A S D F G H J K L : ; " ' Return
Shift Z X C V B N M < , > . ? / Shift
Space

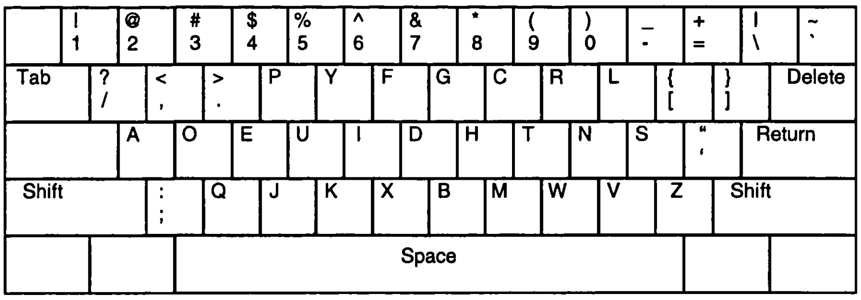
The first typewriter keyboard was patented in 1868 by Sholes, Glidden, and Soulé. Since there were no data to choose one layout over another, they chose an alphabetic layout. It soon became evident that because of frequent typebar jams, this layout was impractical. Ten years later Sholes patented the QWERTY layout, which was developed to solve the jamming problem. The QWERTY physically separates frequent letter pairs, or *digrams*. Since the keys are further apart, the chance of jamming is reduced, allowing the user to type longer and faster without problems.

The common belief that the QWERTY layout was intended to slow down typists is not true. In fact, modern studies comparing alphabetic and QWERTY keyboards show that typing on the QWERTY is as fast or even faster than the alphabetic layout it replaced (Noyes, 1983). However, spacing digrams further apart requires the fingers to travel longer distances, and consequently do more work. The seemingly random placement of the keys on the QWERTY keyboard requires frequent, erratic motions of the fingers over a small area. These motions are difficult to learn and require a long time for proficiency. Expert-level typing takes even longer to achieve and quickly decays with disuse (Gopher & Raij, 1988).

The shape of the QWERTY keyboard is ergonomically unsound. The hands are held close together with the wrists bent outward (*ulnar deviation* or *adduction*), and often upwards as well (*extension*). Excessive finger use with bent wrists has been linked with Repetitive Strain Injuries (RSI) like tenosynovitis (Ilg, 1987). Back and shoulder muscle problems can arise from the slouching and hunched shoulders often caused by poor typing posture.

The Dvořák Simplified Keyboard

Figure 2.2: The Dvořák Simplified Keyboard layout



One of the first alternatives to the QWERTY was the Dvořák Simplified Keyboard. August Dvořák developed the Simplified Keyboard in 1936 as a solution to the problems of the outdated QWERTY. Several design principles were used to solve the problems he saw with the QWERTY keyboard. While these principles were intended for a keyboard, they can easily be applied to many kinds of input devices.

One design principle was to minimise the distance travelled by the fingers. The QWERTY layout requires fingers to move large distances because the digrams are placed far apart. The DSK places digrams close together, reducing the distance fingers travel by as much as 90% (Potosnak, 1988). The QWERTY was designed for use by two fingers, even though by 1936 touch-typing was standard practice. Dvořák attacked this problem by distributing the work amongst the fingers, giving the strongest ones the most work. Also, by noting that simple motions are easier to learn than complex ones, Dvořák was able to exploit the frequent use of certain letter sequences, using simple motions to type them. He hoped that these easier motions would be easier to learn and less tiring than the seemingly random finger motions of the QWERTY.

There have been several experiments which compare the performances of the DSK and QWERTY. In general, most of these show that the DSK is on the order of 2%-5% faster than the QWERTY (Potosnak, 1988), which is a minimal improvement. If there is any real advantage it is not that the user types faster, but that the user does less work. Judging by comments made by users of the DSK, the reduction of finger work does help reduce typing injuries.

Split Keyboards

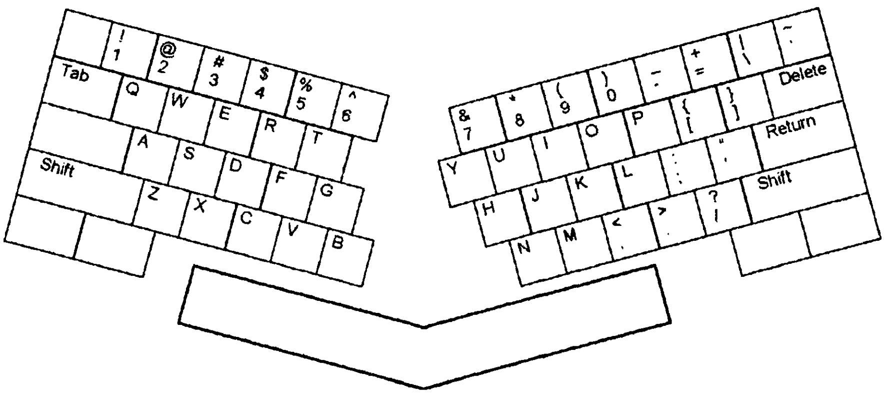


Figure 2.3: A generalised split keyboard

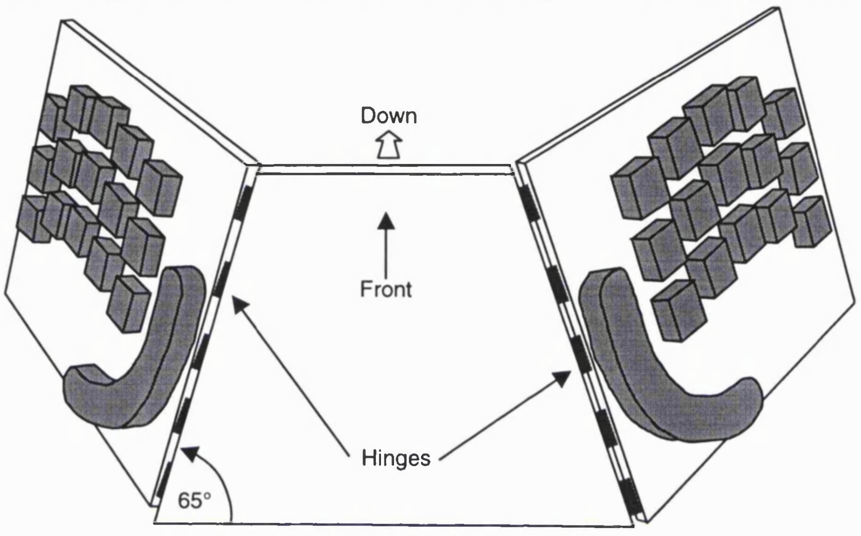
Split keyboards are designed to tackle the problem of ulnar deviation when touch typing, in an attempt to reduce typing-related injuries. On a standard keyboard, touch typing is performed by placing the fingers on the home keys, which are fairly close together. In order to align the fingers on these keys, the wrists must be bent, with the hands facing outward. A split keyboard straightens the wrists by splitting the keyboard down the middle (usually along the line between the T-G-B and Y-H-N) and bending the sides back to make a V shape. Some split keyboards are jointed and allow the user to control the angle between halves. However, most split keyboards have a fixed angle. It has been confirmed experimentally that a split keyboard with a wrist rest can significantly reduce ulnar deviation and wrist extension (Rempel et al., 1996).

With the rising media interest in RSI, split keyboards have become more common. The Microsoft Natural Keyboard alone has sold over a million units worldwide (Microsoft Corporation, 1997). Experiments were performed using people with hand and wrist pains on three different kinds of split keyboards when used over a period of 3 months. The Apple Adjustable and Comfort HealthCare Keyboards showed a decrease of 18% and 11% in the level of pain, while the Microsoft Natural Keyboard showed a 48% decrease in pain levels (Tittiranonda et al., 1996).

While the split keyboard solves some of the ergonomic problems of a standard keyboard, only trained touch typists benefit from its use. Touch typing assigns each half of the keyboard exclusively to one hand. A split keyboard physically separates these halves. Touch typists can easily adjust to this because of their training. Non-trained typists often will have their own way of typing which does not split the keyboard in the same manner. These people might find the split keyboard inconvenient to use, thus reducing their productivity.

Tilt Keyboards

Figure 2.4: Top view of the K-Keyboard, with left and right sections bent downward at 45°



The K-keyboard was developed in 1972 in order to be able to type with the hands in the least stressful position, minimising the strain on the wrists, fingers, and shoulders. In this position, the fingers are slightly curled, the wrists are straight, and the palms face inward and are tilted at an angle of approximately 45° to vertical (Kroemer, 1972).

The K-keyboard is split into two boards, one for each hand (Figure 2.4). The boards are tilted downward at an adjustable angle between 0° and 90°, to minimise pronation. Most users preferred an angle of 45°. Note that at 0° the device effectively becomes a split keyboard. The hinges are tilted forward at a fixed angle of 25° to minimise wrist extension. Keys are offset to match the length of the fingers. There is one large curved space bar for the thumb. This arrangement allows the user to hold their hands in a more comfortable position: with wrists straight and no unnecessary pronation. This is opposed to the QWERTY keyboard which has the wrists extended, adducted, and fully pronated.

Experiments done on the K-keyboard show some improvement over the QWERTY keyboard. While there is no significant difference in typing speed, subjects made fewer errors (7.7% vs. 12.7% for QWERTY). In the experiments, subjects were told to type until they were unable to do so. The subjects who used the QWERTY keyboard tended to stop because of physical pain. Users of the K-keyboard tended to stop because they could no longer concentrate.

Keyboard Summary

The research carried out on the above keyboards has generated a mass of data on the options for keyboard design. Many of these have been adopted into national or international standards (International Organization for Standardization, 1994). The rest are just good ideas to keep in mind when designing a keyboard.

In our analysis of keyboards we have uncovered nine important factors for efficient keyboard design.

1. **Hand position** The least stressful hand position for typing is with the thumbs facing up and inward at around 45°–60°, palms facing together, and fingers slightly curled (Kroemer, 1972). Reducing extension and adduction of the wrists has been shown to reduce pain levels up to 48% in people who suffer from hand or wrist pain (Tittiranonda et al., 1996).

2. **Keyboard layout** According to ISO9995-3 (International Organization for Standardization, 1994), the QWERTY is the international standard keyboard with the DSK being the standard alternative layout. Changes in the layout have little effect on the speed of the device, but minimising finger travel tends to reduce work and the number of errors (Potosnak, 1988).

3. **Keyboard slope** (Figure 2.5) Most users prefer a keyboard sloped towards them at an angle of at least 15° (Potosnak, 1988). The preferred slope is related to the user's hand length and stature. While the slope of the keyboard affects how much the user likes it, the slope does not affect the users performance.

Figure 2.5: The preferred slope of a keyboard

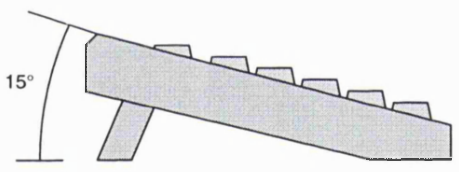
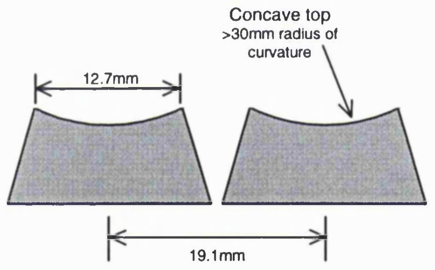


Figure 2.6: Diagram of a key showing preferred sizes and spacing



4. **Number of keys** There is not enough data to know the upper and lower limits for the number of keys on a keyboard. The only guidelines are the obvious. Too few keys slows typing by requiring multiple keystrokes to make some characters. Too many keys slows typing by making it harder to find a specific key (Potosnak, 1988).

5. **Key size and shape** Keys should be rectangular so they fit well together without large gaps between them. A square shape is the most preferred design. To guide the finger to the centre of the key, the tops should be concave, with a radius of at least 30mm (Ilg, 1987). While there is no data for the upper limit of key size, the lower reasonable limit is 12.7mm square for the key tops, with 19.1mm between centres (Potosnak, 1988) (Figure 2.6) The typing speed is dependent on the key size. A square key with a width of 20mm is between 50% and 100% faster than a key width of 5mm (Sears et al., 1993).

6. **Key force and travel** Keys should require somewhere between 20cN (0.7oz) and 70cN (2.5oz) of pressure to activate. The finger should have to travel 4mm to activate the key (Ilg, 1987).

7. Tactile feedback Some sort of tactile feedback is preferred. This can be done by some kind of change in the resistance force of the key once it has been pressed (Potosnak, 1988). Experienced typists, however, may not need tactile feedback or key travel (Guggenbuehl & Krueger, 1991).

8. Auditory feedback Auditory feedback is not as good as tactile, but if the keys are silent and there is no tactile feedback, auditory feedback helps. Some users find it annoying and there should be the option to turn it off (Potosnak, 1988).

9. Key colour Keys which perform a similar function should have the same colour. This aids in finding the keys on a large keyboard. The keys should also have a matte surface, to reduce glare (Potosnak, 1988). The colour of the character on the key should highly contrast the colour of the key itself (Ilg, 1987).

Despite its problems, the QWERTY remains a fast and relatively efficient method for casual as well as intensive desktop text entry. Alternative keyboards, in general, tend to be over-ergonomically designed. That is to say, their design goals of maximising touch typing efficiency has the side effect of being even more difficult to use for untrained typists. On a standard keyboard, operators can develop their own typing style uniquely suited to themselves. This is more difficult to do on some ergonomic keyboards which force the typist into one particular style. The biggest market for a alternative keyboards is formally trained touch typists who perform intensive text entry tasks, requiring a more efficient keyboard.

While quite appropriate for desktop use, the keyboard loses its advantages when shrunk down to a more portable size. The poor ergonomics, which are just noticeable with casual use on a desktop, become quite obvious and limiting on a mobile system. Ergonomic keyboards do not help since they tend to solve the standard keyboard's problems by being bigger. A split keyboard requires extra space for the empty middle and a tilt keyboard takes up even more three dimensional space. While conventional keyboards make poor mobile text input devices, recent advances in touchscreen technology and miniaturisation has produced a new style of keyboard which is more appropriate for a mobile environment: the soft keyboard.

2.2.2 Soft Keyboards

A soft keyboard is not really a keyboard at all, it is just an interesting application of touchscreen technology. A touchscreen can be set up to mimic a keyboard by placing the keys on the screen, which can be operated as if it were a normal keyboard. Since the keyboard exists only in software, the size, shape and even the layout can be adjustable on the fly.

A soft keyboard on a desktop computer or tablet computer has the most potential for flexibility. The keyboard can be enlarged to the size of a standard one, or even larger, allowing the user to interact almost as efficiently as a real keyboard. A 24.6cm soft keyboard has an input rate of 20.3wpm for novice users and 32.5wpm for expert users (Sears et al., 1993). The novice speed is around what one would expect for a standard keyboard, but the expert speed is considerably less. The soft keyboard does has one distinct advantage over a standard keyboard: it can be interactively changed to fit the user's personal preferences. Given enough screen space, there is nothing which would prevent using it as a split keyboard, or even a chord keyboard (see Section 2.2.3). This is a flexibility impossible on any "hard" keyboard.

The smaller displays of notepad-size computers prove to be quite limiting to the flexibility of the soft keyboard. There is not enough space to resize it to a convenient size or shape. The best one could hope for is to be able to get rid of the keyboard when not using it. Input speeds for a 6.8cm soft keyboard are 9.9wpm for a novice and 21.1 for an expert (Sears et al., 1993), significantly less than the speed on the large keyboard, but the room for improvement is much larger. On the smaller keyboard an expert will type 113% faster, as compared to the large keyboard where the expert only types 60% faster. The wide proliferation of mobile computers begs the question of what percentage of users can possibly be experts. The benefits of expert use may not be applicable to the majority of users.

The limitations in typing speed due to the cramped space of a smaller keyboard can be offset by using a stylus to tap on the keys instead of typing with the fingers. A theoretical analysis of upper and lower bounds to the input speed yields a range of 8.9wpm to 30.1wpm for any reasonably sized soft keyboard (Soukoreff & MacKenzie, 1995). Experimental results show that a novice user should expect around 21.1wpm (MacKenzie et al., 1997). This means that, as a keyboard size decreases, typing becomes more difficult, but tapping with a stylus remains just as effective. While this has no benefits for a desktop computer, a small handheld computer would benefit greatly.

There are two major problems of a soft keyboard which have not been addressed so far. The first is the lack of tactile feedback. The lack of feedback requires the user be constantly looking at the screen to know if they hit the correct key. This can be compensated somewhat by audible feedback, which can be performed by a beep from the computer whenever a key is hit. However some users find this rather annoying and it should not be depended upon (Potosnak, 1988).

The other major problem is that a soft keyboard takes up valuable screen real-estate. This is especially a problem for miniaturised computers where the keyboard must take up most of the screen to be usable, even with a stylus. As a result, the soft keyboard really cannot be used as a primary text input device for a mobile computer. However, it can make a useful secondary, backup text input for when the primary input (e.g. handwriting recognition) fails.

The soft keyboard solves the size problem by squeezing the keys into as small a space as possible. This exacerbates the problems of the poor ergonomics of the standard keyboard to the point where normal typing is impossible. Typing with a stylus does improve the interaction, but there is still the problem of the keyboard taking up much of the screen space. This leaves us with a text input device which can be used efficiently with a mobile computer, but may not be appropriate for significant amounts of text input. We still need a primary text input device which is *designed* to be small, not shrunk down as an afterthought. In the following sections we will discuss input devices which do not suffer the same size limitations of keyboards.

2.2.3 Chord Keyboards

All the keyboard alternatives discussed above are just modified versions of the standard keyboard. A character is made by pressing one key, or one key in combination with one (or more) shift keys. This allows any number of characters, as long as there is room on the keyboard. A chord keyboard takes a different approach. There is one key for each finger. Multiple keys are pressed simultaneously to create characters, in the same way that a chord is made on a piano. Pressing combinations of keys in this way is called *chording*.

Chord keyboards were first used by the US Post Office in the 1960's for entering numbers for mail sorting (Potosnak, 1988). Most early research on chord keyboards concentrated on limited applications, such as entering numeric data. In the 1980's chord keyboards were reevaluated and applied towards a general text keyboard. The first thing which needed to be solved before this could happen was the limited number of characters. A one handed chord keyboard has only five keys. This translates to 31 possible combinations. This is enough for all the letters, with room for a few more characters, like `<Space>` and `<Return>`. There are numerous ways to increase the number of characters beyond just 31. The following are just some of them:

Two handed chording A ten key chord keyboard has 1023 possible characters. This more than enough for general text input.

Thumb keys One or more extra keys can be added in reach of the thumb. Sixteen more characters are added per thumb key. More combinations are possible if the thumb keys can be pressed simultaneously. This is the most common solution.

Sticky shift keys A sticky shift, when pressed once, acts on the next one chord. When double-pressed (like double-clicking a mouse) it acts on all chords until the shift is hit again. For each shift, the number of possible characters doubles.

Multiple state keys Instead of an on/off key like most keyboards, it is possible to have a three or more state key. A three state keyboard uses keys which can be pushed up, down, or not at all. This gives 243 combinations for one hand.

Additional finger keys It is possible to have more than one key per finger, such as an extra row, above or below the base row. This is effectively the same as using multiple state keys.

By using one or more of these combinations it is possible to create all the same characters that can be made on a standard keyboard. With this problem removed, it is possible to use a chord keyboard as a general text input device.

The biggest advantage of chord keyboards is that they can be made significantly smaller than a standard keyboard. Each finger presses only one key, so that key can be placed in the keyboard in the most efficient position. In practice, most chord keyboards are around the size of the hand. The palm rests on an empty base, while the fingers press keys located radially around it (Figure 2.7). This setup permits a full range of text input, but at a fraction of the size, with no real loss of comfort.

The chord keyboard suffers from a similar problem to the split keyboard. Arranging each half of a split keyboard for each hand leaves an empty gap in the middle (Figure 2.3). The size of a chord keyboard is not limited by the size of the keys, since half a dozen or so small keys cannot take up much space. The problem is space between the keys. In the chord keyboard shown in Figure 2.7, most of the device is

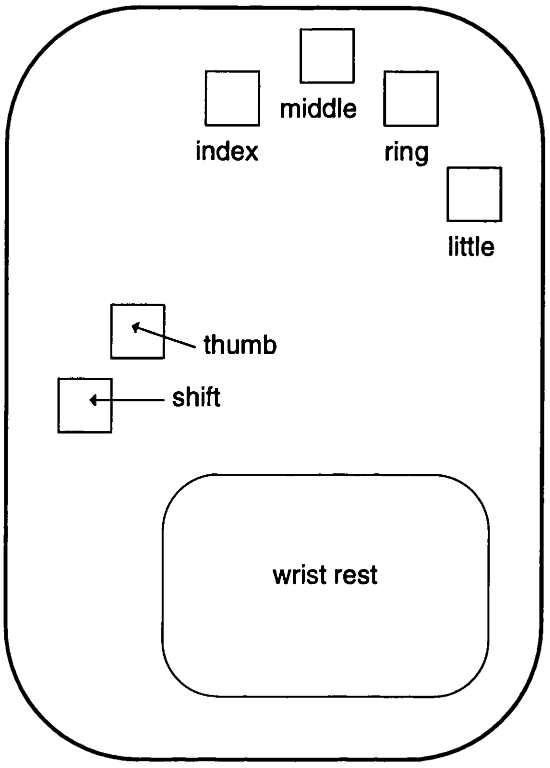


Figure 2.7: A generalised chord keyboard

covered by the hand or the fingers, making the device unnecessarily big. One solution to this problem is to put the keys on a small box which is held or strapped to the palm of the hand. This style of chord keyboard takes up much less space, but it can also hinder the user's interactions when performing real-world tasks. Extra time must be spent putting the device down, or, if strapped in, it might get in the way of the user's actions.

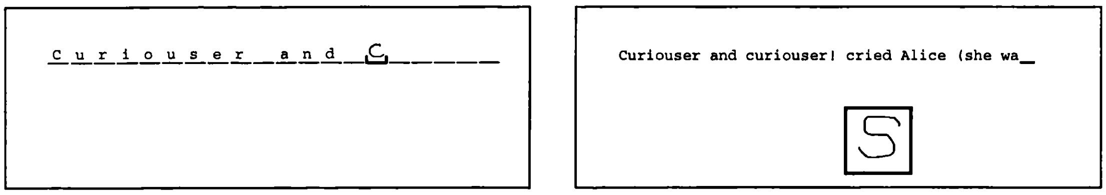
Another disadvantage of chord keyboards is that the fastest typists will *always* type faster on a standard keyboard. The reason for this is key overlap. On a standard keyboard, one often presses more than one key at a time. The key which is pressed first is entered first, but another key is in the process of being pressed. That means up to ten characters can simultaneously be in the process of being made. A one handed chord keyboard can make only one character at a time. There is no overlap. For the novice typist, this is not a problem. Novices can chord faster on a chord keyboard with less training. After twenty hours of training a one-handed chord keyboard user averages around 29wpm, while a QWERTY user averages around 20wpm. After 35 hours of use a chord keyboard levels off around 36wpm (Gopher & Raij, 1988).

A further disadvantage is that it is not possible to type without training on a chord keyboard, although it is possible is on a standard one. On the other hand, learning to type on a chord keyboard is easier because the chord shapes can have some physical correspondence to the letter being created. For example, holding out only the thumb and little finger makes a Y shape. If the chord for Y is made with the thumb and little finger, it becomes much easier to remember. The chords for an entire character set can usually be learned within an hour (Gopher & Raij, 1988).

One advantage chord keyboards have over alternative layouts, like the DSK, is that learning to type on a chord keyboard does not have any effect on the ability to type on a standard one. A person can switch back and forth without any problem. Touch-typing on a standard keyboard is difficult because the large number of keys and movements make it easy to miss a key. This problem is partly alleviated by most alternative keyboards. With a chord keyboard each finger uses only one key, since almost no movement is involved, it is impossible to miss a key. This is especially useful for blind, or otherwise disabled users. The benefits of reduced motion are reflected in the smaller error rates for chord keyboards.

2.2.4 Handwriting Recognition

Handwriting recognition is a particularly attractive method for text input, given the extremely wide proficiency of writing in the population. In the ideal case a handwriting recognition system would be as easy to use as writing on paper. Anyone who can write could pick up the device for the first time and bypass all the time normally spent learning to use the interface. Input speeds are comparable to novice QWERTY speeds, with printing speeds ranging from 12wpm to 23wpm. Speeds for cursive writing are higher ranging from 16wpm to over 30wpm (Soukoreff & MacKenzie, 1995), but character recognition, even by a human, is much more difficult.


C u r i o u s e r a n d C
Curiouser and curiouser! cried Alice (she wa_
S

(a) Text input by writing sequentially on the screen (b) Text input by writing each character in a special block

Figure 2.8: Two methods for handwriting recognition

Most handwriting recognition systems use one of two methods for input. The first is writing on the screen itself. In this method, input area is set up as a series of blanks, which are filled in sequentially, one character per blank (Figure 2.8(a)), much like filling out a form on paper. This method requires the stylus to be positioned over each character. While this may not take up much space, it may cause problems since the hand might cover up parts of the screen. It also has the disadvantages of requiring constant visual supervision. If the stylus is slightly offset from where the user thinks it is, a character could be incorrectly recognised or printed in the wrong place. Finally, the sequential method of writing translates poorly to a heads-up display, since it would require a full sized pad to write on, in addition to the display.

The second method for handwriting recognition is writing in place. This consists of writing on a special block outside the main part of the screen (Figure 2.8(b)). Each character is written in the block, one on top of the next. The character is inserted on the screen at the location of the cursor, just as it is done with a keyboard. The cursor can be positioned by using the stylus as the graphic input device as well. This simplifies the recognition by adding a constraint on the position . It also avoids the problem of hand occluding the display while writing. There is the added benefit of causing less fatigue by minimising hand motions (Goldberg & Richardson, 1993). As long as the stylus starts out in the correct position, visual supervision is unnecessary. In fact, the handwriting recognition block need not even be on the screen at all. Writing in place would translate well to a heads-up display, where characters could be written on very small pad outside of the field of vision and displayed on a monitor in front of an eye.

In addition to the two methods for handwriting recognition, many systems also include a soft keyboard which can be displayed to allow an alternative to writing. This is especially useful when entering less frequently used punctuation or other hard-to-recognise characters. While a soft keyboard allows faster text entry than handwriting, it is not sufficient for primary use because of the possibility of fatigue with long-term use (MacKenzie et al., 1997).

At the time of writing, handwriting recognition is far from perfect. The general agreement is that to be widely accepted, a handwriting recognition system would have to have 97% or higher accuracy (MacKenzie & Zhang, 1997). For isolated printed characters, human recognition is 96.8%, just short of what we would require from a computer. One advantage a handwriting recognition system has over a human reader in that it has temporal information as well. For example, the computer would know the difference between two very closely spaced v's and a w because it would have seen the stylus lift from the tablet in between characters. This could help a handwriting recognition system achieve the desired levels of accuracy.

The current state of the art for handwriting recognition yields an accuracy between 87% and 93% (MacKenzie & Chang, 1997). This falls short of the 97% needed to be widely accepted. In addition, these figures were more effected by the user's adaptations to the quirks of device, rather than the device's recognition techniques.

The underlying problem with handwriting recognition is that Roman characters are not well suited for computer recognition. Next two sections will discuss two handwriting recognition systems which bypass this problem by reshaping the alphabet to a much simpler form.

Unistrokes

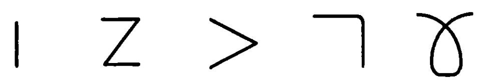


Figure 2.9: The five basic strokes in the unistroke alphabet

Unistrokes were designed to provide a fast, easy to recognise alternative to the standard Roman character set (Goldberg & Richardson, 1993). The characters are generated from a basic set of 5 different strokes, each one made by up to 3 simple motions (Figure 2.9). Each of the 5 strokes has 4 possible orientations and can be drawn from 2 different directions. This yields 40 possible characters that are different enough to be easily differentiated by a computer. The entire printable ASCII character set can be covered by using shift keys, or some of the other methods mentioned in Section 2.2.3 for increasing the character range for chord keyboards. Another key design factor is that unistrokes are designed specifically to use the writing in place method for character entry. This further facilitates writing for the user and character differentiation for the device.

Unistrokes can be entered quite quickly, with a novice input speed of 34wpm. It is expected that entry rates can go as high as 41wpm for expert users. The novice rate is faster than normal printed text (12–23wpm) and is around the same speed as the upper end of the cursive entry rate (30+wpm). The unistroke character set can be learned within 10 minutes of use, but it takes a week of practice to achieve the novice input rate of 34wpm. No error rates were mentioned in the literature.

Unistrokes were intended for "power" users who would use a stylus input enough to benefit from the extra time spent learning the system. Since the handheld market was risky enough to begin with, the manufacturers of these systems had little inclination to cater to a small section of an already specialised market. While it was acknowledged that unistrokes were a good idea, there was little desire to try it.

Graffiti

Graffiti was introduced as an attempt to bring the unistroke idea of a simplified alphabet to a mass market. Instead of simplifying all the letters to the extent where text more resembles Cuniform than English, the letters of the alphabet are simplified just far enough to make them easily differentiated by a computer, but remain recognisable to a human. All but five of Graffiti's letters have a strong resemblance to the Roman Character, and even the remaining five still retain an obvious relationship.

The benefit of Graffiti's approach is that it is very easy to learn. The initial accuracy for the system is 86%. After five minutes of practice the accuracy goes up to 97%, which meets the above-mentioned requirements for an acceptable handwriting recognition system. With continued use recognition levels may reach up to 99%. The accuracy after one week without using the system at all is 97%, implying that the Graffiti alphabet is retained in long-term memory (MacKenzie & Zhang, 1997). Text entry rates are claimed by the manufacturer to be around 30wpm (US Robotics, 1996).

Graffiti was developed by Palm Computing as a commercial text entry system for handheld computers. While the Graffiti software is available for several different systems, it comes standard with Palm Computing's PalmPilot personal organiser. The PalmPilot currently enjoys 66% of the handheld market (McCall, 1997) which shows that Graffiti has been rather successful at its intended job. However it is generally thought that Graffiti must be a short term solution to allow handheld computers to get off the ground while effective general handwriting recognition systems can be developed.

2.2.5 Glove-Based Text Entry

One of the more novel text input methods is text input via a glove-based device. These devices use one or both hands to input text by either sensing the motion involved with gestures or by detecting contact between the fingers and other fingers or a special pad. These devices tend to be quite portable because the glove is worn instead of held and consequently is lightweight and takes up very little space.

Hand Gestures

Several systems have been developed to recognise sign language. The first such device was the Digital Data Entry Glove (Grimes, 1983), which was developed to recognise the letters in American Sign Language to provide a more portable and comfortable text input device. This glove used three kinds of sensors: flex sensors on the joints to measure finger flexion, inertial sensors to detect orientation, and contact sensors to detect finger adduction and contact between parts of the hand. The basic theory was sound, but it turned out to be impractical since the contact sensors required excessively precise finger positions in order to be recognised. In addition, the recognition was hard-coded into the device, preventing any adaptation to the user. While a commercial device was never built, this glove did pave the way for the DataGlove (Zimmerman & Lanier, 1987) which was partly influenced by this design.

A second system based on both Grimes (1983) and the DataGlove used sign language recognition for computer mediated communication (Kramer et al., 1991). The primary intention was to use the computer to help facilitate communication by deaf people by translating the hand motions of sign language into characters, which could then be spoken by a synthesised voice. However, there is nothing to prevent this system from being used purely for text input. The glove in this system, the CyberGlove, improved upon the DataGlove's optical fibre joint sensors by using strain gauges which give more accurate joint angles over a wider range. Strain gauges on the joints of each finger and the wrist measure the flexion angles. Additional strain gauges are positioned between the fingers, and on the thumb and wrist to measure adduction, abduction and opposition. This system uses an adaptive pattern recognition system to recognise the sign language characters and has been released commercially as the GesturePlus^{TM}. An experienced user can reach input speeds of over 40wpm (Virtual Technologies Inc, 1997). Unfortunately, this requires top-of-the-line equipment at a cost of over US$10,000 (Burdea & Coiffet, 1994). Despite its high portability, the high cost of this system prevents its use as an everyday text input system.

The GloveTalk uses a DataGlove to convert the gestures of a specialised sign language into speech (Fels & Hinton, 1993). The number of words is limited, with 66 root words and 5 possible suffixes ('-s', '-ly', etc.), but these can be created in near-real time by one hand. Since each motion is a word which must be recognised by the computer before it can be spoken, it is possible to do away with the vocal aspect of the system and use it for text input only. The major problems with such a system is low vocabulary and high cost, which limit its usability as a text input device. A later version of this device, the GloveTalkII, uses two gloves and a foot-peddle to create individual formants which can be combined to speak at a third to nearly half of the normal rate. While the speed is fast for text input, the learning time is over 100 hours, which makes it unlikely to be a widely usable text input device (Fels & Hinton, 1995).

Contact Gestures

Contact gloves are a somewhat cheaper and computationally simpler alternative to gesture recognition. These devices sense contact between fingers and either the hand, other fingers, or a special tablet. Touching contacts completes a circuit between the two sensors, which is detected by the computer. The gestures which make these contacts can be converted into characters or act as function keys.

Hartwig (1978) describes a contact glove used to enter numbers. The glove has contacts on each finger. These are touched against a special tablet, which is divided into 3 sections. Each section can produce one of five characters, depending on the finger which touches it. The first section contains the numbers `1–5`, the second section contains the numbers `6–9` and `0`, and the third section contains the symbols $+$ $-$ $\times$ $\div$ and $=$. This style of contact glove is equivalent to a keyboard which produces a different character depending on which finger is used to press the key. There is no research indicating the performance of such devices. However, it is clear that hitting the tablet in the correct place requires visual supervision. This limits the position of the tablet to where it can be easily viewed and also prevents the use of a heads-up display. As a consequence this system would be less mobile than a glove-only system.

Another contact glove, the Pinch glove makes gestures by touching fingers against each other (Holands, 1996). This is more mobile than Hartwig's glove, since it does not require a tablet. Unfortunately, it can only be used for function input, since there are too few combinations to enter a decent fraction of the 100 printable ASCII characters. Using two gloves, or increasing the number of contacts on one might allow enough gestures to enter the alphabet, but it could suffer similar problems to the Digital Data Entry Glove, where the finger combinations were too complex to be easily made.

In summary, glove-based text entry is extremely well suited for use in a mobile environment. The biggest drawback is the difficulty is recognising enough separate gestures to allow useful text input. Differentiation between the large number of gestures necessary for text input on a contact glove requires excessive precision, limiting their usefulness. Contact with a separate tablet allows easier gestures, but at a loss of mobility. Gloves like the CyberGlove, which sense the full orientation of the hand, can be used to recognise existing sign language for text input, but the prohibitive cost of these devices precludes their everyday use.

2.2.6 Voice

In some ways voice is the ideal text input device. A microphone takes up practically no space, has negligible weight, and is completely hands-free. It can be used just as easily in almost any environment and conditions. It is also much faster than most other text input devices, with an input speed on the order of 150wpm (Dragon Systems, 1998). However, the technology is based on pattern recognition, which limits its usefulness.

Speech recognition systems perform a number of steps in converting an individual utterance into a word (Scahill et al., 1996). First, the utterance is divided into individual sound units called *phonemes*. The phonemes are then decorrelated to extract certain features. This simplified model of the utterance is compared to a known vocabulary, from which the most appropriate match is selected.

The last step of this process shows the two main sources of trouble for speech recognition systems: vocabulary and accuracy (Hunt, 1997). The vocabulary size needed to use a speech recognition system as a general text input is between 5000 and 60,000 words. One existing commercial system, the Dragon NaturallySpeaking$^{\text{TM}}$ Deluxe, can maintain a 30,000–50,000 word active vocabulary, and a total vocabulary of 230,000 words (Dragon Systems, 1998). The need to change vocabularies and the potentially high memory use are relatively minor problems brought about by the size of the active vocabulary. The main problem is the difficulty of recognition: the more words there are to choose from, the more difficult it is to differentiate between them.

A high accuracy is essential for a speech recognition system. Not only does the system need to correctly recognise the words, but it also needs to reject invalid input (coughs, "um"s, etc.). Consequently, a system must have a rejection threshold. This is the point at which, if the match is not good enough, the utterance is ignored. This leads to three different kinds of recognition errors. The first is a *substitution error*, which is made when a valid utterance is incorrectly interpreted as a different word. A second error is an *insertion error*, which is made when an invalid utterance is mistakenly identified as valid input. The last error is a *deletion error*, which is made when the quality of a valid utterance is below the threshold, and it is considered invalid (Johnston, 1996).

Setting the rejection threshold too low will cause more false acceptances (substitution and rejection errors), allowing potentially harmful events to occur, e.g. confusing "(cough) read file" for "delete file". Alternatively, setting the threshold too high will cause more false rejections (deletion errors), annoying the user either by ignoring their input or by constantly asking them to repeat themselves. Consequently, for speech recognition systems, high accuracy is not enough. The systems must also have a good method for safely handling errors.

Another problem with speech recognition system is the issue of privacy. Privacy is not much of an issue in a working environment. In a work environment, a mobile computer would be mostly used for input tasks such as inventory or damage control, or communication tasks, such as accessing an information database to assist in performing some work. In either case, voice control is useful, and allows the user to work hands free. Vocal text input is ideal for these situations.

The problem arises in personal computing. The nature of mobile computing means the computers are just as likely to be used in a public environment as not. In general, one is unlikely to want to write sensitive work-related or personal documents in public. Even if the speech recognition was performed flawlessly, people would still want some private way of interacting with the computer. For a computer which is likely to be used in public, the option of non-vocal input is necessary, even if it is only used as a backup.

2.2.7 Summary of Text Input Devices

The QWERTY keyboard is extremely popular as a text input device due to the momentum built up from over a hundred years of use. However, the QWERTY's design is not ergonomically sound. The hands are held close together with the wrists extended and bent outward. This has been linked to cases of RSI in full sized keyboards. If the QWERTY is shrunk, the strained conditions are made even worse, giving more potential for injury. The poor ergonomics of the QWERTY and the rising popularity of mobile computers has caused a surge of alternative methods for text entry.

Ergonomically designed keyboards tend to solve the QWERTY's problems by repositioning the keys for maximum efficiency when touch typing. This is usually done by separating the keys in some manner, which has the side effect of making the keyboard bigger. While this might be a great benefit for people working in office conditions, it makes the problem of mobile computing even worse.

Part of the problem of the poor ergonomics of a shrunken QWERTY can be solved by using a soft keyboard. These devices can be adjusted on the fly to fit the ergonomic needs of the user. However, this is rarely done in practice, and instead the more popular solution is to use a stylus to tap the keys, one at a time. This reduces the strain of touch typing in such a small area, but text entry speeds are significantly reduced. These facts lead to the conclusion that conventional keyboards are undesirable for mobile computer inputs. Soft keyboards, on the other hand, are adaptable enough to make them a useful backup text input.

Chord keyboards have a comparatively small number of keys, allowing touch-typing in confined areas without reduced comfort. A chord keyboard is not as fast as a desktop keyboard for an expert, but it can achieve quite reasonable speeds for novice users, and with less training than a keyboard. The 36wpm expert speed on a chord keyboard is roughly the same as the higher end of expected speeds on most handwriting systems. Chording does have the advantage of being potentially less fatiguing than handwriting systems since there is minimal movement and no stylus needs to be held. The biggest hurdle to overcome with chord keyboards is that the keymap must be learned before it can be used, even though the alphabet can usually be learned within an hour. But, if we can learn anything from unistrokes it is that the market is very hesitant to embrace such a technology.

Modern handwriting recognition systems are not flexible enough to analyse general handwriting. Instead the most effective systems require the user to learn a simplified alphabet, like Graffiti. Writing in place at the end of the workspace removes the need for moving the wrist and minimises the finger work. While this is slower than a desktop keyboard, it is less fatiguing, and remains one of the best options for a mobile interface.

Glove-based gesture systems are also highly portable, but they suffer from excessive cost or poor accuracy. Contact gloves are affordable enough for everyday use, but they are might not be accurate enough to recognise the large number of gestures needed for text entry. Gloves which can sense the full orientation of the hand can learn to recognise enough gestures for text entry, but these are too expensive for casual use.

Speech recognition systems provide an extremely portable, hands-free text input method. As of this writing the widespread use is limited primarily by the vocabulary size and accuracy of these systems. These are purely technological constraints which may be solved in the near future. However, even with near-perfect accuracy, recognition errors are inevitable, thus necessitating a safe method for avoiding potentially damaging misinterpretations. While speech recognition is a useful text input method, there are situations in which vocal input is inconvenient or undesirable. Consequently, it is a good idea to back up voice input with a silent text input alternative.

As it stands, a handwriting system using a simplified alphabet, speech recognition, a contact glove with a tablet, or a chord keyboard, are the text inputs most suited for use with a mobile computer. The handwriting system, speech interface, and chord keyboard are effective on a heads-up style system while conventional handwriting and the contact glove are more effective on a notepad-style system. The next step in designing a mobile input is to look at the options for graphic input devices and determine which of them can work effectively in a mobile environment.

2.3 An Overview of Pointer Input

Numerous pointer input devices have been developed over the last few decades as a result of the trend in computers to a more graphically oriented interface. Most of the more recent developments with pointers take advantage of the advancements in computing power and miniaturisation. But the most recent spurt in novel pointers has grown from the need for a device which works effectively in a mobile environment.

Many of the design criteria for text input devices also apply to graphic input devices. For example, the ideas of minimising work and using tactile or auditory feedback make as much sense for pointing as the do for typing. Consequently, in this section, we will be concentrating more on the factors which limit portability, and the performances of the devices.

With text input devices, performance was judged quantitatively, primarily by a device's input speed. It is necessary to use a similar quantitative method for comparing the performance of graphic input devices. In the early 1950's Fitts developed a method for measuring the performance of human motion. While this was originally applied to ergonomics and kinematics, Fitts' law has been adopted into the field of Human Computer Interaction as a tool for comparing graphic input devices.

2.3.1 Fitts' Law

Fitts' law is an application of information theory to motor coordination which provides a metric for comparing various graphic input devices in simple pointing tasks (MacKenzie, 1992). The model is based on Shannon's Theorem 17, which describes the information capacity C of a channel with a finite bandwidth B in terms of signal power S and noise power N:

$$C = B \log_2 \frac{S+N}{N} \tag{2.1}$$

The bandwidth B is measured in Hz and the capacity C is measured in bits per second.

Fitts applied this law to human motor coordination by finding the appropriate analogues for the terms. Channel capacity C becomes the *index of performance*, IP. The inverse of the bandwidth ($1/B$) becomes the *movement time*, MT, which is the time it takes to perform a specific motion. The strength of the signal S becomes the *amplitude* of the motion A. The noise N in the signal becomes the *width* of the destination of the motion. The yields the modified equation:

$$IP = \frac{\log_2 \frac{2A}{W}}{MT} \tag{2.2}$$

where Fitts' logarithmic term of $\frac{2A}{W}$ replaces Shannon's logarithmic term. The 2 is an arbitrary constant to ensure that IP is positive when the motion starts outside the destination region.

The term in the numerator is the log of the ratio of the size of the motion to the size of region in which it ends. This is called the *index of difficulty* (ID):

$$ID = log_2 \frac{2A}{W} \tag{2.3}$$

Using this term, Fitts' law can be rewritten as:

$$IP = \frac{ID}{MT} \tag{2.4}$$

Since ID and MT are known terms, this equation can be solved directly. However the results are somewhat lacking as it does not provide a term for reaction time, which turns out to be necessary.

Equation 2.4 can be rewritten, solving for MT, as:

$$MT = \frac{ID}{IP} \tag{2.5}$$

which can, in turn, be solved by performing a linear least squares fit on the known variables, ID and MT:

$$MT = a + bID \tag{2.6}$$

where a is the reaction time and IP = $\frac{1}{b}$.

By using Equation 2.6, the index of performance of a graphic input device can be found. IP, as we recall, is measured in bits/s. Applying this to an input device tells us how fast information can be sent from user to computer. This yields a useful method for comparing effectiveness of various graphic input devices.

Note that the $\frac{2A}{W}$ term in ID is unitless, and is unchanged under any scaling. This means that the time it takes to perform a motion (MT) is independent of the scale of the device. This is limited to situations involving the same Index of Performance. If the system is scaled up to the point where new muscles are involved in the action, a new IP is involved. This is why tapping on a soft keyboard with a stylus is equally fast, regardless of the size of the keyboard. A small soft keyboard would require motions from only the wrist, while a large one would require motion mostly at the elbow. These actions have differing IPs (Balakrishnan & MacKenzie, 1997) and thus would take different amounts of time to perform.

Unfortunately there are some problems of consistency from experiment to experiment when using Fitts' Law. Different experimental setups can give widely ranging values for the same device. For example, in different experiments, the mouse has been found to have an IP as low as 2.6 and as high as 10.4 (MacKenzie, 1992). Fortunately, the ratio of IPs is more or less constant from experiment to experiment. For example, in the experiments above the joystick was found to have IPs of 1.2 and 4.5 respectively. This gives joystick to mouse IP ratios of 0.46 and 0.43, which are very close. As a consequence of this, ratios of IPs should be used to compare the performances of devices checked in different experimental setups. In this thesis we will be using a "normalised" version of Fitts' law in which the performance is calculated by the IP of the device divided by the median value of the IP for the mouse, as calculated in the same experimental setup. In other words:

$$\widehat{IP}_x = \frac{IP_x}{\overline{IP}_{\text{mouse}}} \tag{2.7}$$

The mouse was chosen because of its widespread use and relatively high performance. This method is used in Table 2.1 to rate the performances of various common graphic input devices.

It is obvious that a consistent standard for rating graphic input devices is necessary. At the time of this writing, the International Organisation of Standardization is working on ISO 9241-9: "Requirements for non-keyboard input devices". This is still in the committee draft stage, but it is expected to propose a method based on Fitts' Law for comparing graphic input devices (MacKenzie & Oniszczak, 1998). In this proposal, the Index of Performance (IP) is renamed *Throughput* (TP), and a set of consistent rules for measuring it are laid out. Unfortunately, nothing official has been published. As a consequence it is only included here as a reference for future research. All comparisons of graphic input devices in this thesis will be based solely on Fitts' law as described above.

		Experiment				
		MacKenzie & Oniszczak (1998)	I. Scott MacKenzie & Buxton (1991)	Epps (1986)	Card et al. (1978)	Mithal & Douglas (1996)
Device	stylus		1.09			
	mouse	0.80–1.20	1.00	1.00	1.00	1.00
	trackball	0.53–0.93	0.73	1.15		
	trackpoint					0.71
	touchpad			0.62–0.88		
	joystick			0.42–0.46	0.43	
	trackpad	0.26–0.29				

Table 2.1: A comparison of the normalised performance of six graphic input devices. The performance value is the ratio of the device's IP to the median IP calculated for the mouse in the same experiment.

2.3.2 Arrow Keys

The earliest personal computers were almost entirely text-based and had little graphic capability. Arrow keys located on the keyboard were used for pointer control. There are various methods for interaction with arrow keys, the most common of which is *step keys*. These move the cursor up or down one line and back or forward one character. In a graphic environment step keys move the pointer one or more pixels, the size of the step often changeable by either another set of keys, or as a function of the duration of the key press. *Jump keys* are designed for a more specific type of environment. Jump keys move the cursor to one of a series of predefined areas on the screen. This is often used in hypertext applications to jump between links. It has been shown that jump keys are faster and more preferable than using a mouse in a text-only hypertext system (Greenstein & Arnaut, 1988). This has not been proven for complicated text and graphic windows-based environments.

The simplicity of arrow keys makes them a very versatile graphic input method. Arrow keys can be made very small to fit various sized devices, while still retaining the same level of usability. While arrow keys are not terribly fast nor efficient for general graphic interaction, they do make a decent backup pointer control. There is no Fitts Law data for the performance of arrow keys.

2.3.3 Joystick

A joystick is a vertical handle mounted on a base, usually with one or more buttons used for selection. A *displacement joystick* uses potentiometers or similar technology to measure the displacement of the handle from the normal position. This is translated to a magnitude and direction. Springs are often used to move the handle back into its normal position. An *isometric joystick* uses strain gauges to measure the force on the handle. The handle itself does not move. A *switch activated joystick* or *joyswitch* uses a set of switches, evenly spaced around the handle, when the handle is moved in one of the directions, one or more switches are activated determining the direction, but not the magnitude of the motion.

The popularity of video games in the late 70's and early 80's brought joysticks to a mainstream audience. Joysticks were occasionally used on home computers, usually for games since there were very few other graphical applications. Joysticks work best for navigation and low-precision pointing (Greenstein & Arnaut, 1988) which, while excellent for games and visualisation, compare poorly to mice when performing higher precision tasks such as drawing. As a consequence, joysticks for personal computers remains a primarily niche market.

Standard joysticks are not easily used in a mobile environment because the shaft needs a large enough base to provide leverage. If the base is too light compared with the forces required to operate the joystick, the base will simply move with the joystick, preventing any motion. One solution to the miniaturisation problem which has become popular in recent years is the microjoystick, one common version of which is called the trackpoint. This device is a finger-controlled isometric joystick shrunk down to the size and shape of a pencil eraser. This is embedded in a board and operated with the thumb or index finger. Its small size makes it more difficult to use and beginners may have a very hard time using it. Part of the difficulty stems from the high gain of the device. Since it is so small, the forces applied to the device need to be amplified by a large amount to be turned into pointer motion. This does not just amplify the motion, but also the natural vibrations, or *tremors*, in the finger, which are too small to be seen using most other input devices (Mithal & Douglas, 1996). Despite these problems, the trackpoint has a large advantage in that it takes up only around a square centimeter of work space. As a consequence, the trackpoint has recently become widely available as a graphic input for notebook computers. Note that the IP for the trackpoint is different from the IP of a normal-sized joystick since different muscles are used (Table 2.1).

2.3.4 Trackball

A *trackball* is a small ball placed inside a fixed housing. The ball is free to rotate in the socket and is usually manipulated by the thumb, index finger, or the whole hand, depending on the size of the device. Rotating the ball causes the pointer to move in the direction of rotation. Switches located near the trackball are used for selection.

The most commonly used trackballs allow all the work to be done by the fingers. This tends to be more comfortable than devices such as the mouse or full-sized touchpads because the hand does not move. Tactile feedback is provided by the motion of the trackball itself. The fingers can feel the speed and direction of the rotation, giving them better control of the pointer than visual supervision alone. Trackballs work best for pointing and selection as well as manipulating symbols (Greenstein & Arnaut, 1988). This makes them appropriate for operating a menu and desktop style GUI.

Trackballs are well suited to a mobile environment. They can be made very small, allowing them to be embedded in a handheld computer sized base. One drawback is the relation between the selection switches and the trackball. If the same finger is used to move the ball and press the buttons, it is possible that these actions might interfere, since the same muscles are used for both (MacKenzie & Oniszczak, 1998). This is especially a problem with click-and-drag tasks, where holding the button down might get in the way of moving the ball.

2.3.5 The Mouse

A *mouse* is a small, hand-held device which is dragged along a surface. Moving the mouse causes the pointer on the screen to perform a corresponding motion. Selection is performed by one or more buttons at the end of the mouse. There are two basic kinds of mice: optical and mechanical. A *mechanical mouse* has a small ball on the underside. Dragging the mouse across a surface rotates the ball, determining the motion of the pointer. The moving parts in a mechanical mouse make it prone to errors by picking up lint or other small particles. An *optical mouse* requires a special mouse pad imprinted with a fine grid. The mouse counts the number of lines it passes horizontally and vertically, usually by means of an LED and optical sensor. The rate at which it passes lines on the grid determines the pointer motion. An optical mouse has no moving parts and is much less prone to errors (Greenstein & Arnaut, 1988).

Much research has been done on the ergonomics of the mouse. A mouse must be textured to avoid the hand slipping (Abernethy & Hodes, 1987). The mouse must also have buttons which require enough activation force that the fingers can easily rest on them without accidently pressing them. The mouse should also be broader at the end with the buttons to accommodate finger spread (Greenstein & Arnaut, 1988).

A mouse cannot be made effectively smaller than it is. The device itself can be shrunk, but the space needed to use it cannot without a loss in performance. Excessive amplification of a mouse can lead to a tremor problem similar to that found in the trackpoint (see Section 2.3.3). As a consequence the mouse will remain usable as desktop input device only, and cannot be made suitable for a mobile environment.

2.3.6 Trackpad

Trackpads have evolved from graphic tablets, which have been around since the 70's. These were never very widespread until they were applied to portable computers, where they found their niche. A trackpad is a small, touch sensitive region a few square inches in area, usually located just below the keyboard. It works by measuring the change in capacitance caused by the user's finger on a grid of electrodes . The finger's touch can be detected at each node, which works out to 250 points per inch resolution (Andrews, 1994). Other input methods exist, such as conductive devices, or infrared or acoustic (Greenstein & Arnaut, 1988). The former uses two conductive layers made of electrode grids, which work by touching when pressure is applied. This can be made with the highest resolution and detects any pressure, not just a finger. Infrared and acoustic devices tend to be low resolution and are not very common for mobile devices.

Selection with touchpads is commonly done by one of two methods. The first is to have an external button which is pressed with another finger, or by moving the pointing finger from the trackpad. This method tends to be slower and suffers similar problems to the trackball of muscle interference. The second method is often called *lift and tap*. After the finger moves the pointer over the proper location, the finger is quickly lifted and returned to the pad. This is faster than using an external button (MacKenzie & Oniszczak, 1998). Both methods suffer from *jitter*. Lifting the finger from the pad can be interpreted as a slight motion, meaning that the selected point is not exactly where the user wants it to be. If selection is performed by using another finger on a separate button, the jitter problem is avoided. Another problem touchpads suffer from is the lack of tactile feedback. This can be compensated for in software by an audible or visual indicator, so the user knows selection has been occurred. Alternative selection methods exist, but are not as common. One method uses pressure deviations to differentiate between light touches (dragging the pointer) and hard touches (selection) (Greenstein & Arnaut, 1988). This can be adapted to provide a tactile "click" when pressed, thus providing tactile feedback. The tactile version has been shown have 25% higher IP than the lift-and-tap method (MacKenzie & Oniszczak, 1998).

Trackpads have the advantage of having no moving parts. This makes them less prone to collecting dirt, easier to clean, and more adaptable to the hostile environments one is likely to encounter with a mobile computer (Greenstein & Arnaut, 1988).

2.3.7 Touch Screens

A touch screen is just a transparent trackpad overlaid on top of a display. This is a very common input method for tablet and handheld computers. Both touch screens and trackpads tend to be both just as easy to use, with certain advantages and disadvantages for each. Trackpads have the advantage over touchscreens that the finger or hand does not obscure the display, allowing interaction with no visual supervision. On the other hand, separating the display and input pad takes up valuable workspace, and a device with integrated input and output is more intuitive to use. CRT-based monitors have trouble with drift, which would make a touch screen very difficult to use (Greenstein & Arnaut, 1988). However, this is not a problem with mobile systems since they tend to use liquid crystal displays, which do not suffer from drift.

2.3.8 Stylus

The problem of occluding the display of a touch screen is magnified by the small size of mobile displays. One solution to this problem is to use a stylus instead of a finger. The stylus tends not to occlude the display as much, and gives a sharper point for more accurate pointing. Using a stylus is less work than moving a finger or hand, making interacting faster and less tiring (Greenstein & Arnaut, 1988). Using a stylus also allows one to merge the graphic and text input into one device simplifying interaction (see Section 2.2.4). A stylus can also be effectively used on a trackpad with a heads-up display. This tends to lead to the conclusion that, in a mobile environment, it is more efficient to operate a touch screen or a trackpad with a stylus than with a finger.

2.3.9 Gesture, Eye Tracking, and Voice

There are other, less common, graphic input techniques which can easily be adapted for use in a mobile environment. In this section we will focus on gesture, eye tracking and voice as graphic input. These are not very popular at the moment, usually due to expense or poor usability, but advances in technology may make them cheap and usable enough to provide a viable input method.

Gesture

Gesture input is usually done via glove or body-based motion trackers. This generates 3D position data, which are recognised as gestures which manipulate 3D objects. The nature of the input devices makes gesture input very mobile. In addition, glove-based devices which can be used as graphic input can also be used to interpret gestures as text input (See Section 2.2.5).

Most body-based or glove based motion trackers are too expensive to be used for the 2D GUIs common on modern mobile computers. Until a virtual reality GUI becomes feasible for mobile computers, using these devices for graphic input is overkill. However, a simplified glove system may be more suitable to 2D output. Instead of full 6 degree of freedom (DoF) tracking with wrist and finger joint recognition, one could only measure the motion of one or two fingers or just the wrist. This would make a glove interface much more suited for a conventional 2D environment.

Eye Tracking

Most eye tracking uses sensors to recognise and track features on the surface of the eyes to determine the viewing direction. Selection is performed either by a separate button or dwell time (Greenstein & Arnaut, 1988). Selection by *dwell time* is done by leaving the pointer in roughly the same place for a specific short period of time. This can have problems in a windows-style or hypertext GUI. The user cannot look at a hot area such as an icon, button, or menus for too long without activating it. In a windows or hypertext environment, there are usually many such hot areas all over the display, This means that the user can only glance at these objects to avoid risk of accidental selection. This is sometimes referred to as the "Midas Touch" problem (Gips & Oliveri, 1996). Selection with dwell time also prevents click-and-drag and double-click options.

Eye tracking also tends to have problems with locating small targets because of involuntary eye motion. In general, because of the potentially small size and low encumbrance, eye tracking is well suited to a mobile environment, but since it is less accurate and more expensive than the alternatives it would only suffice when no other graphic input is an option.

Voice

Voice input is very common with wearable computers because of the small amount of unintrusive hardware required. Speech recognition is well adapted to be usable as a text input, but is not nearly as well suited to general graphic manipulation (Hunt, 1997). It tends to work best for menu selection, or other constrained tasks. Drawing and similar activities are rather difficult (a picture is worth a thousand words), making speech-based input graphic devices rather inefficient.

2.3.10 Bioelectric-Controlled Input Devices

A bioelectrically controlled device would be especially convenient as a graphic interface for a mobile system. One advantage is that the sensors can be remote from the motions they detect. For example, finger motions can be sensed by electrodes on the forearm (Hiraiwa et al., 1993). This, plus the small size of the sensors, allows such a system to be discrete, even possible to be hidden underneath clothing. Another benefit is that one can learn to interact with the device through biofeedback. By this process, the computer effectively becomes an extension of the user's body. In order to understand how these devices can work, we must go into detail on the physiology, methods for measuring, and safety issues involved with bioelectric signals.

The most common bioelectric signals which can used for input devices are the electromyogram (EMG), electroencephalogram (EEG), electro-oculogram (EOG) and Galvanic Skin Response (GSR). These are not the only options, but just the most common. In this section we will describe these signals, focusing particularly on their usability for controlling a graphic input device. Special attention will be paid to Electromyography.

Electromyography

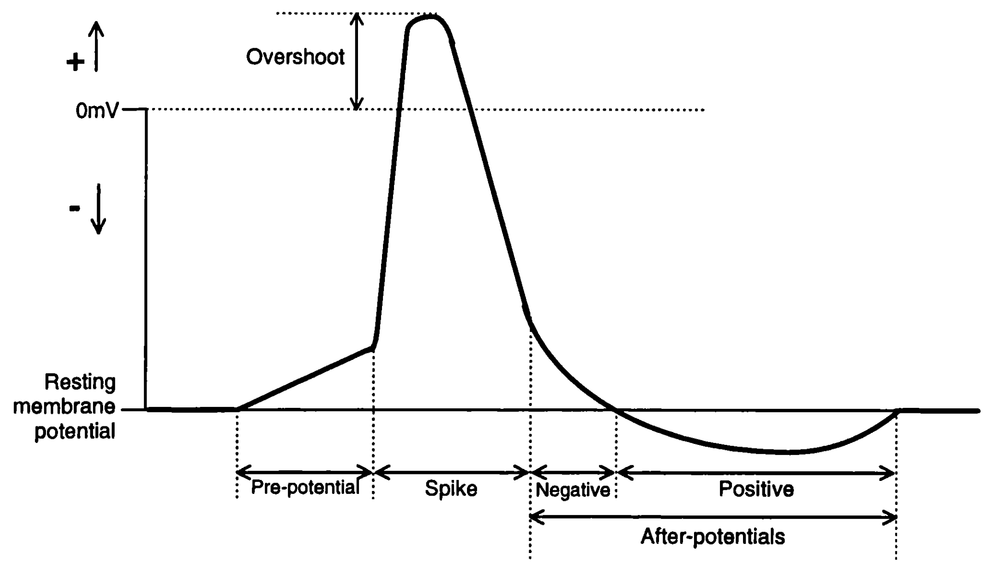


Figure 2.10: Diagram of an action potential and its important features. Based on a diagram in Geddes (1972)

Electromyography is the reading of action potentials generated by muscle use. When a single muscle fibre is excited, a wave of charge propagates down the cell membrane, causing the muscle to contract. When the potential of a point on the fibre is measured, it yields a curve of the form seen in Figure 2.10.

The action potential starts at a small negative voltage called the *resting potential*. In the *pre-potential* period, the voltage slowly increases until it shoots up during the *spike*. The maximum potential of the spike is usually a small positive voltage called the *overshoot*. The period after the spike and before the membrane potential returns to its resting potential is called the *after-potentials* which can have two parts. The *negative* part of the after-potential is the slow decrease from the end of the spike to the resting potential. Sometimes the after-potential falls below the resting potential for a short period. This is called the *positive*.

A muscle is made up of many fibres and its EMG is generated from the sum of the action potentials of all the excited fibres. Light muscle use excites only a small number of fibres, yielding a few clear action potentials. A fully contracted muscle uses many fibres, generating several overlapping action potentials, yielding a seemingly random, cluttered, complex waveform (Figure 2.11).

Reading signals Electromyograms can be read either directly via depth electrodes or through the skin via surface electrodes. Depth electrodes are inserted subdermally and come into direct contact with the muscle. These have a clear, well defined EMG, requiring less amplification and filtering. However, the health and safety issues involved with the daily application of needle electrodes preclude their use as a casual computer input device. Surface electrodes are placed on the skin directly above a muscle. The EMG is weaker and noisier than from depth electrodes due to signal damping from the skin. It is also harder to read the EMG of deeper muscles with surface electrodes. The problems with surface electrodes

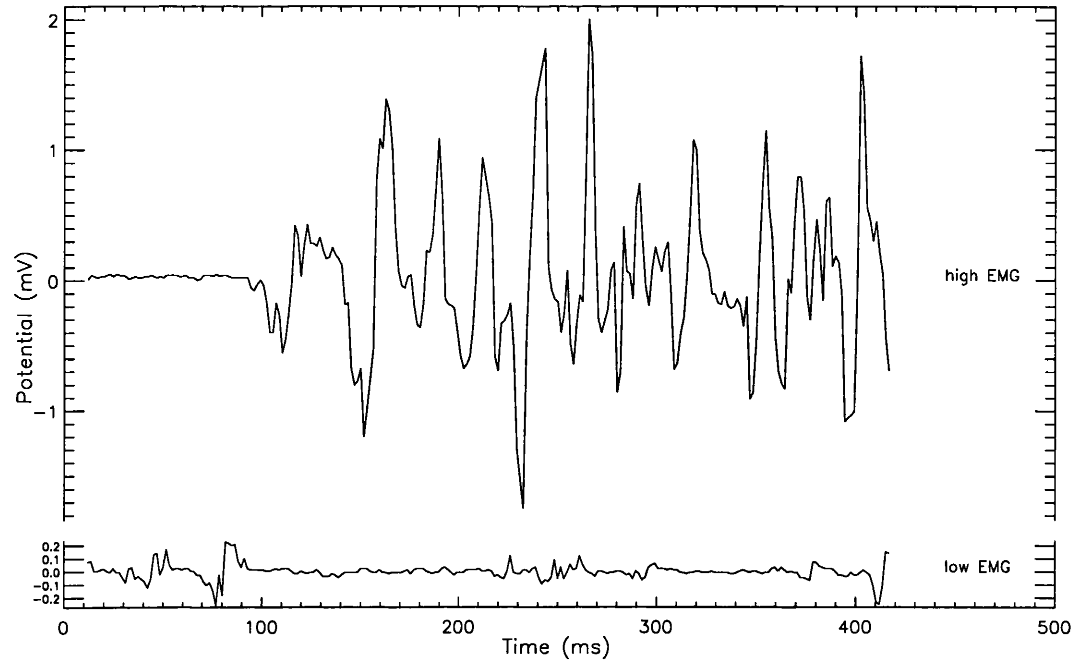


Figure 2.11: Graphs of the EMGs for low muscle use and high muscle use

are made up for by their ease of application and use on a daily basis, giving them a vast advantage over depth electrodes.

Surface electrodes come in a variety of sizes and styles. In general, they consist of 3 parts. First is the electrode housing. The edges of the bottom of the housing are usually lined with adhesive to keep it affixed to the skin. The housing is either made from the electrode plate, or the electrode plate is fastened to the housing. The plate is usually made of a non-biologically active metal such as silver, gold, surgical steel, or tin.

The second part is the electrolyte, which can take various forms. Usually it will be a paste or gel. Liquid electrolytes are applied between the skin and housing upon use. A semi-solid gel electrolyte is permanently fixed to the electrode housing. Most commercially available electrolytes are made of a hypoallergenic saline solution and should not produce any allergic reactions (3M Health Care Customer Helpline, 1997).

Dry electrodes do not contain an electrolyte, the electrode plate is put in direct contact with the skin. Dry electrodes are useful in situations where preparing the skin is undesirable, or in environments where a gel would freeze or liquefy. Dry electrodes, however, tend to be more expensive.

The last part is the connector. This is usually a knob connecting to the electrode plate which emerges from the top side of the housing. A clip is usually connected here. In some smaller electrodes, the connector is a permanently fixed wire.

The size is an important factor in the electrode. A larger contact area for the electrode yields a stronger signal. However, a smaller electrode will be easier to fix over a specific muscle, giving a more precise signal. Also smaller electrodes are easier to apply and wear. The effective contact area can be increased by roughening the skin under the electrode. This will increase the surface area of skin exposed, giving a stronger signal (Geddes, 1972). In general the size of electrode used must be small enough to fit over the muscle while being large enough to provide a decent signal.

Safety is another factor to consider when using electrodes. The condition of the skin affects the quality of the bioelectric signal. The skin should be cleaned before using. Cleaning off dead skin will enhance the signal, especially if an abrasive soap is used, as roughening the skin will improve the connection.

Bacteria and fungi which are brushed off by normal interaction with the environment will grow under the electrode. The density of these organisms vary on different parts of the body ranging from as low as 105/cm^2 on the forearms to 2.4 million per cm^2 in the armpit (Geddes, 1972). Significant amounts of these organisms may appear after long periods of electrode use. While most of these are harmless, some can lead to infection or disease. As a consequence, the skin should be washed after use, especially if the electrode is on for a long period of time. Attaching an electrode to damaged skin should be avoided. If it must be done, special care must be taken to clean the skin before and especially after use. A normal gel electrode can be safely used on unbroken skin for periods up to 24 hours without risk (3M Health Care Customer Helpline, 1997). This period can be increased up to two weeks when using special electrodes on prepared skin.

In order to read the EMG, the electrode must be connected to an amplifier, which in turn is connected to a power supply. The primary safety factor for the amplifier is isolation from the power source. While this is especially important for use with devices connected to power mains, it is still an issue with battery-powered computers. According to British health and safety regulations, the maximum current allowed to reach the subject is 500μA (British Standards, 1993). This is usually done by optically isolating the amplifier.

Electroencephalography

Electroencephalography is the measurement of the electric potentials generated by neurons in the brain. This is much more complicated than electromyography and not entirely understood. Discernible waveforms can be created by conscious thought as well as by sensory input. EEGs are low frequency (up to 50Hz, but usually less than 20Hz) and low amplitude (usually between 2 and 100μV, and as high as 400μV) (Thompson & Patterson, 1974). The low amplitudes require better amplifiers to read, and make the system more susceptible to noise. The low frequencies reduce the potential speed of the input device.

Measuring EEGs is performed by placing several electrodes in various regions of the scalp, usually over hairy areas. In order to make a solid contact though the hair, an electrolyte gel is applied. This has the unfortunate side effect of being somewhat messy and uncomfortable for the subject. Some EEG systems avoid this problem by placing the electrodes only on the forehead, where standard surface electrodes can be used.

Electro-oculography

Electro-oculography is the measurement of the difference in potential between the cornea (front) and retina (back) in the eye. This can be used to measure the intensity of light seen by the eye, or the orientation (gaze direction) of the eye. The latter is useful for eye tracking, as it provides a convenient alternative to those mentioned in Section 2.3.9.

Determining gaze direction is fairly simple and straightforward compared to controlling a pointer from an EEG or EMG. Electrodes are placed, above, below, to the left and right of the eye. The difference in voltage from left to right indicates the horizontal orientation. The voltage difference between the electrodes above and below the eye indicates the vertical orientation. The signals are on the order of $20\mu V$ per degree of arc and have a frequency under 30Hz (Gips & Oliveri, 1996).

Galvanic Skin Response

Galvanic Skin Response (also known as Electrodermal Response or EDR) is the measurement of the variations in the resistance of the skin and usually generated by varying emotional states. The cycle periods of the GSR are on the order of seconds, significantly slower than either EEG or EMG. The low frequencies and high correlation to emotional stress levels make GSR most appropriate for use for lie detection or biofeedback techniques for stress reduction. GSR is very dependent on the condition of the skin. The signals are harder to detect on areas of the skin with sweat, cuts, or scars as they tend to interfere with the signals. EEG and EMG signals are actually improved by these defects as it improves current flow through the skin (Geddes, 1972).

Bioelectric Computer Interfaces

There are a few experimental and commercial bioelectric interfaces in existence today. The commercial systems tend not discuss their exact methods, so only experimental ones will be detailed here. The following summary is an overview of some common methods used to convert EEG, EOG, and EMG data into pointer motion.

Wolpaw and McFarland have developed one and two dimensional pointers using the EEG[1]. The one dimensional pointer works with a fair degree of accuracy. Electrodes over the sensimotor cortex in each hemisphere measure the EEG and filter the signals to isolate the mu waves. Mu waves are in the 8–12Hz band and are affected by conscious thought, usually related to motor control (Lusted & Knapp, 1996). The pointer's motion in controlled by the difference in the amplitude of the mu waves of the right and left hemispheres. The subject learns to control the pointer by thinking about random movements (like running or floating). The subject watches the pointer and learns which thoughts control what motions. The two dimensional pointer is more complicated and is still being worked on, but it is based on the same principles. The 2D pointer is about 70% accurate in controlling the rough direction for a trained user. Controlling the specific direction, or the speed of the motion is too difficult and needs improvement.

An EOG-based eye tracking system called EagleEyes has been developed at Boston College and is designed for use by disabled people (Gips & Oliveri, 1996). The tracking was tested with a setup in which the subject would enter text by selecting letters on a soft keyboard. Selection was performed by dwell time. The time it took to learn the device ranged from 15 minutes to months. The accuracy is described as "fair" and control of the device is limited due to problems with dwell time as mentioned in Section 2.3.9.

Several methods exist for EMG control of prosthetic devices. The methods used for controlling a prosthesis can be easily adapted to either a gesture input or pointer control. To get an overview of the

[1]Browne, M. W. "How Brain Waves Can Fly a Plane", *The New York Times*. 7 March, 1997.

various methods we will review three different situations: first is a method for control of a prosthetic arm, second is a method for controlling a prosthetic or virtual hand, and the last is a method for simplifying EMG signals for computer recognition.

Kermani & Badie (1990) used the integral of the absolute value (IAV) of the EMG from the biceps and triceps to control elbow and wrist rotations of a prosthesis for above-elbow amputees. The space defined by the IAVs for the biceps (IAVB) and triceps (IAVT) is segmented into areas which control certain actions. For low values of both IAVB and IAVT, no motion occurs. For medium-to high IAVB and low IAVT, elbow flexion occurs. For medium-to high IAVT and low IAVB, elbow extension occurs. These are effectively the same motions which would occur naturally. With roughly equal and low to medium IATB and IAVT, wrist pronation occurs. For medium to high IATB and IAVT, wrist supination occurs. These last two motions are not related to the actual motions these muscles would perform, and thus would have to be learned. Noise which could cause unwanted movements is eliminated by a rule-based strategy. There are two disadvantages to this method. First is that the wrist motions need to be learned. The second is that only one motion can be performed at a time, despite the independence of the elbow and wrist. These are not major problems when using this method for a prosthetic device, but it translates poorly to a method for computer interaction.

Hiraiwa et al. (1993) use a neural network to control the fingers on a prosthetic hand. One electrode pair was placed on the anterior side of the forearm, above the flexor digitorum superficialis. Another pair was placed on the posterior side, above the extensor digitorum. Each EMG was transformed into a power spectrum every 200ms. The spectra were then rebinned down to 10 points each . The 10 values from each EMG were fed into the neural network, which produced two joint angles for each finger. Training was performed by placing a DataGlove over the existing hand, which would be used to determine joint angles. During the training, the user randomly moves their fingers, mirroring the actions with the missing hand. The training lasts for 150 seconds. The RMS error averaged at around 25°, which provides sufficient accuracy to manipulate objects. This method has also been suggested as a gesture input for virtual reality systems.

Chang et al. (1996) developed a method for recognising EMGs in real time by analysing various features. The first feature is the zero crossing rate (ZCR) which is the number of times the EMG switches from positive to negative over a fixed time period. This is roughly equivalent to the frequency, but is easier to calculate. The start of a muscle contraction is identified by the ZCR passing a threshold of about 30–40% of the average ZCR.

The second feature is the cepstral coefficients, which are the inverse fourier transform of the logarithm of the signal's power spectrum. These can be estimated using an nth order recursive function which generates n coefficients. A large n is computationally expensive, and small n is less precise. It was found that $n = 4$ was a decent compromise. This generates a vector of 4 cepstral coefficients which identifies the motion.

Before using the system, the user must create a set of these vectors, one for each desired motion. These base vectors are then compared to the feature vectors generated during the system's use using the modified maximum likelihood distance (MMLD). The minimum MMLD determines which motion was performed. In a test of five movements of the head, this method yields a recognition rate averaging 95% with a delay time of under 170ms. However, the recognition only tells what the motion is, not the amplitude. In order to make a more accurate pointer, the number of motions must be increased, which would increase the recognition time.

Summary of bioelectric devices The EEG is commonly used in biofeedback systems for clinical purposes (Carroll, 1984), but usage as a computer input is limited due to the low accuracy. Eye trackers based on the EOG are currently feasible, and in some cases are more convenient than video-based systems. EMG input methods are also feasible and come in a variety of styles. Both the rule-based system and the feature recognition system fail to provide a continuous graphic input since they give a direction, but no magnitude. The neural network system is the only one which provides an continuous set of values. The processing speed for each method varies. The rule based method is the fastest since it is the least complicated. The neural network and feature recognition systems take roughly the same amount of time with 4 and 6 updates per second. The update rate is to too slow for a useful input device, but, since these systems were created 5 and 2 years ago, one can expect that current computer speeds should be fast enough to use at least the neural network system, if not both systems, without significant delay times.

2.3.11 Summary of Graphic Input Devices

Using arrow keys is the simplest method for pointer control. This may be slow, but it allows easy pointer control at the pixel resolution and is guaranteed to always work, regardless of the environment . The performance of arrow keys can be improved by using them as jump keys in specialised applications. In addition, they are easy to implement. Any computer which has room for a few small buttons can use them. This makes them well suited for use as a backup pointer control for a mobile system.

Joysticks are most useful in environments which require navigation or low precision pointing, but they are not very portable. The trackpoint is a much more portable derivation of the joystick, but the high amplification of the device makes it difficult to master.

Partly due to its obvious usage, high performance, and successful marketing, the mouse has dominated the graphic input market and has became the *de facto* standard interface for a desktop environment. Unfortunately the mouse does not translate well to a mobile environment. Shrinking the mouse requires amplifying its motions, potentially causing problems similar to the trackpoint.

Trackballs can be made very small, making them quite popular for use with mobile computers. While these devices work well for pointing and selection tasks, they have trouble with tasks which require use of the selection button and the trackball at the same time, such as clicking and dragging. Trackpads are slightly larger than the trackball, but are still quite portable and thus popular in many mobile systems. These devices suffer similar problems with click-and-drag to the trackball when used with a separate selection button. When used in the lift-and-tap method, click-and-drag becomes quite easy, but fine selection becomes difficult.

Touchscreens are very intuitive and easy to use, but when shrunk to a more mobile size, the finger can occlude much of the workspace, making it difficult to use. Using a stylus on a touch screen not only solves the occlusion problem, but can be manipulated faster and with less work. Stylus-based systems also have the added benefit of being able to integrate text and graphic input, making them quite popular for handheld systems.

Gesture, eye tracking, and voice interfaces are well suited for use in a mobile environment. The high end glove and body tracking systems are too complex for a 2D environment, but a simplified glove used for 2D input might be adequate for a mobile system. The primary trouble with eye tracking is the poor resolution due to involuntary eye motions. Selection by dwell time also has a number of problems, and it is often easier to just use a hand operated button instead. Voice makes a very poor graphic input. A vocal system is limited to acting like arrow keys for basic cursor control or acting like function keys.

Bioelectric measurements such as EMG and EOG have much potential as a graphic interface in a mobile environment. Modern computers are fast enough to handle the computational complexity of analysing these signals. The hardware required to measure the bioelectric signals can be made very small and the electrode connections are lightweight, safe, and barely noticeable to the user. The motions required to operate such an interface are normal body motions. No stylus or board needs to be held. This makes bioelectric input particularly well suited to mobile computing.

2.4 The State of the Art of Mobile Computing

The portability of a mobile computer is effected by four basic factors. First is the ease of the user interface. This includes such issues as the size of the keyboard, accuracy of the handwriting recognition or voice recognition, sensitivity of the trackball, etc. Also included are the issues of screen visibility and software design, but they are beyond the scope of this thesis.

Second is the encumbrance of the device when it's not being used. This not only includes the size and weight of the device, but where it is kept as well.

The third factor is the time it takes to bring the computer out of a passive, away state to an active, useful one. This does not mean it must be turned on and booted up, just the amount of time the user needs to bring the computer from a safe position where the user can perform other tasks (where the encumbrance is important), to a usable one, where they can operate the device (where the effectiveness of the user interface is important). For a desktop computer this is simply the time it takes to sit down and type. A laptop would need to be picked up or taken out and then opened up.

The final factor is the complement of the third factor. It is the time it takes to safely put the computer away (temporarily away – not necessarily turned off) so the user can interact with the real world in an unimpeded manner. These factors provide a methodical and qualitative basis for determining if one system is more portable than another. In using this method, we can judge how the input devices limit the portability of a system and thereby find out how to avoid those limitations.

There are three main types of mobile computers: the Portable Computer (i.e. laptops and other miniaturised computers), the Handheld Computer(which includes the Personal Digital Assistant (PDA) and similar devices), and the Wearable Computer. In this section we will discuss each type and how their portability factors compare. The types of portable and handheld computers discussed are here are based on the standard definitions used by Dataquest, a marketing research company (Charlton, 1997).

2.4.1 Portable Computers

A *transportable computer* (also *lunchbox computer*) is the largest portable computer, being self-contained and designed to be easily moved. These computers are usually fairly large, usually weighing around 18–20 pounds. The large size of these machines means that they usually have a full, or near-full sized keyboard and monitor.

A *laptop computer* is also a self-contained unit, but is somewhat smaller than a transportable computer, being usually less than 15 pounds. The main common factor of laptops is the clamshell design. This means that, when closed it is in a passive state in the form of a rectangular box. When opened the computer enters the active state, with the monitor embedded in the top half and the keyboard and graphic input in the lower half.

A *notebook computer* is smaller version of the laptop often around the size of an A4 sheet of paper and weighing under 8 pounds. A *tablet computer* is around the same size as a notebook, but uses a stylus instead of a keyboard, and is used in the manner of an electronic clipboard .

The next smaller size is the *subnotebook* or *ultraportable computer.* This is a smaller version of a notebook, often without a built-in floppy drive, weighing 4 pounds or less. A *notepad computer* is a subnotebook which uses a stylus input instead of a keyboard .

These computers evolved from the miniaturisation of a standard PC. The size has been decreased by various methods, often by doing away with the less frequently used components, or accessing them only though PCMCIA cards or communications ports. Another method for decreasing size is to use older, well understood technology which can be made smaller and less power-consuming. This often leaves the user with the tradeoff of size and portability to power and flexibility.

Except for tablets and notepads, all the portables need to be placed on a surface such as a table or a lap in order to used. In addition, they all tend to have the same input devices. Text is performed almost exclusively by a QWERTY-style keyboard. Usually this is a miniature keyboard, with a reduced number of auxiliary keys such as function keys or a numeric pad. Some computers use an expandable keyboard, which is a standard keyboard which opens up to 11.4 inches across, but fits into a 9.7 inches when closed[2]. Graphic input usually has a variety of options ranging from separate mice for the larger computers to trackballs, trackpads, and trackpoints for most of the others. These computers are fairly slow to take out. The machine must be taken out from wherever it is stored, placed on a surface, and then opened up. Putting them away is requires the opposite action. Storing a portable computer of this size is usually done by keeping it in a small bag hung over the shoulder.

Tablet and notepad computers, by definition, are limited to stylus input devices. The pen is usually just a hard, pointed piece of plastic used to point to a touch-sensitive display. The simplicity of the stylus is fortunate, since its small size means it is easily lost. These computers are fairly fast to use and put away: The stylus must be taken out of the main housing and the cover opened (assuming there is one). Putting them away and storing them is roughly the same as for the non-stylus computer above.

[2]Glitman, R. & McLaughlin, L."IBM and Digital: Two Cool Ones for the Road". *PC World.* 25 April, 1995.

2.4.2 Handheld Computers

Handheld computers are small, lightweight, battery-powered computers which are specifically designed to be used while held in the hand. These computers are the fastest growing segment of the mobile computer market. In 1997 worldwide unit sales of handhelds was almost 12% of the mobile computer market. Unit sales are expected to increase by an average of 40% per year, as compared to 22% for other mobile computers. By the year 2001 handheld computers should have over 20% of the unit sales in the global mobile computer market (Charlton, 1997).

There are two kinds of handheld computer. The first is the *expandable organiser*, which is a sort of cross between a PC and a calculator. These are primarily used for schedule planning, note taking, and other personal information management. These are controlled by a proprietary operating system. The expandable nature of these devices allows them to plug into external devices, often proprietary as well. These can be used for additional memory or for connecting a full sized keyboard, but are most often used for communications, such a cellular modems for sending faxes and other information, or connecting to a desktop PC or network to update the information on both machines. These devices tend to be around 3" $\times$ 6" $\times^3/_4$" and weigh less than a pound.

The *standard handheld* is slightly larger than an organiser and is also often used for personal management and communication. The major difference is that this usually uses standardised software (like Windows CE) and communications protocols (like PCMCIA cards) to allow communication between devices as well as with desktop PCs. This is the fastest growing area for handheld computers, with an expected average annual growth of over 55% in worldwide unit sales over the next 3 years (Charlton, 1997).

The input devices for handheld computers vary from device to device. Text input is usually performed either by handwriting recognition or a miniature keyboard. Voice is possible, but not very common. Graphic input is either touchscreen, stylus, or by arrow and function keys. The miniature keyboard devices are easy to take out and use, they just need to be taken out of the pocket and opened up. The stylus-operated handhelds need to be removed from the pocket, the cover removed (if there is one) and the stylus taken out. Both actions take about the same amount of time and effort. Storage when not used is fairly simple, since the device can be kept in a pocket.

2.4.3 Wearable Computers

The same technological advances which have made handheld computers possible have also given rise to wearable computers. Instead of being designed for a compact form which can fit in the hand, a wearable computer is designed to be worn on the body in the same manner as clothing or accessories. The distribution of weight around the body allows the computer to be larger, and thus more powerful, than a handheld computer, but not less convenient to carry. While still a few years away from commercial acceptance, wearable computing is growing in popularity with researchers and niche markets. Interest in the field has reached the point where research has started to become organised on the international level, with the first International Symposium on Wearable Computers being held in 1997.

While all wearable computers are unique, they also follow certain trends. Almost all wearable computers are built with Commercial Off The Shelf (COTS) technology, and are based around an Intel CPU. The bulk of the hardware, including the CPU, memory, batteries, ports and other various connectors, is often concentrated in a compact form which can be worn on a belt on in a backpack. The display is usually either a monocle or stereo heads-up display. Input devices vary from system to system, but speech recognition is the most common, usually with the option of backup text and graphic input devices (Bass, 1996a). Miniaturised keyboards, chord keyboards, and stylus inputs are the most common text inputs. Pointer control is most frequently done with trackballs, trackpads, and joysticks. These are often mounted on the CPU housing or on a separate body-mounted board.

Wearable computers tend to fall into one of two categories: *general purpose* or *special purpose.* General purpose machines are designed to have a variety of uses. These often run standardised desktop operating systems and software applications, and are designed to be customisable by the user. Most commercial systems are in this category. Via's Wearable$^{\text{TM}}$ and Rockwell's Trekker$^{\text{TM}}$ are good examples of general purpose wearables. The default setup for both machines is to use the Windows 95 operating system with vocal text input, and a trackpad for pointer control. There are connections for keyboards, mice, video, and assorted serial, parallel, and PCMCIA ports. These connections allow users to choose their own input devices, in case they are not satisfied with the default configuration.

Special purpose wearable computers tend to use specialised software or input devices in order to perform one or more well-defined tasks. This class of wearable computer tends to be made mostly by universities, with Carnegie Mellon University (CMU) and Massachusetts Institute of Technology (MIT) being the two main centres for research. The CMU research tends to concentrate on the use of wearable computing for maintenance and assisting navigation, while the research at MIT tends to deal more with personal information management (PIM).

A vehicle maintenance is a common application for a wearable computer (Bass, 1996b). This includes the tasks of inspection, troubleshooting, and repairs. Normally these tasks require a manual, a checklist, and schematics, which require turning pages, entering data, etc. This may not be easy to perform at all times, especially when in the constrained positions likely in vehicle maintenance. Using a computer for these tasks allows the user to scroll through the instructions or schematics without needing to turn pages. Items on a checklist can be selected without needing both hands and data can be entered while performing the task instead of afterwards. Speech is a useful input method, since it leaves both hands free. However, there are occasional problems in determining whether the user is speaking to a coworker or the computer. An alternative to speech is CMU's input dial and pressure switch. This can be operated with one hand and can be used to scroll data or select items from a list, but its use cannot easily be extended beyond this application to general graphic input control.

CMU's Metronaut is an example of a wearable computer designed to combine navigation and PIM tasks (Jastrzembski, 1997). The system uses a bar code reader to obtain information from objects in the environment, such as signs or flyers. A pager is used to communicate with a computer network to negotiate PIM tasks. In this system, the user's position is determined by scanning bar codes placed on landmarks in the environment. The computer can combine the position information with the user's schedule to provide directions to the user's next meeting.

While speech is useful for text input, it is insufficient for graphic control. One solution to this is to do away with the graphic interface and use audio as the sole input and output of the computer by using a *Speech User Interface* (SUI). An SUI uses speech commands to control the computer, while replies are in the form of generated speech or non-speech auditory cues called *earcons*. This user interface can be used for telephony systems, such as British Telecom's Stap (Miah et al., 1998), or for remote telephone access of a computer database such as Sun's SpeechActs (Yankelovich, 1994). An SUI can also be used to directly access a computer, making it a potentially efficient interface for a wearable computer.

An audio-only interface is not as efficient as a GUI for browsing data. For example, reciting a large list of messages places a large cognitive load on the user. To solve this problem, the computer must be able to parse and reduce the data into a manageable size. For audio messages this can be accomplished by keyword spotting or speaker identification (Roy et al., 1997). Text messages can be parsed by commonly existing search techniques. In addition to being able to parse the data the computer must be able to understand the complex natural language commands necessary to specify the search criteria. The question, "Which messages are about today's meeting?" is a typical command the system would need to be able to understand and provide a reasonable response without overwhelming the user.

Existing SUIs can perform such tasks as scheduling or providing directions, but the range of tasks is not as large as the wide variety of existing GUI applications. This is partly because GUIs have been around much longer and have had time to develop more applications. It is also partly due to the fact that visual scanning is much more efficient for interpreting large amounts of data. The use of an SUI as a general purpose operating system is still a number of years away.

The main problem with wearable computers is the lack of decent input devices. Currently either voice and board-based devices are used. Voice, while useful for pure text input, is ineffectual for graphic interaction, requiring a separate pointing device which negates the hands-free advantage of voice input. Using a SUI instead of a GUI bypasses the pointer problem, but the interface will not work with many existing applications. Board-based devices include just about all non-vocal devices: keyboards, stylus inputs, and most types of graphic pointers. To be usable in a mobile environment, a board-based input device needs to be either held or mounted on the body.

Anything which needs to be held is going reduce the portability of the system. A hand-held device needs be stored somewhere, and consequently it must be then taken out in order to be used. This slows the time to start up or put away the system. In addition a hand-held device often requires one hand to hold the device and the other to operate it. This limits the actions one can take while using it. The device must also be safely put away when not being used, often in a pocket, or hooked to a belt. An example of this is a stylus-based computer. The pad must be taken out, then the stylus removed from it. Then stylus is then held with the other hand while the person writes. No other tasks can be done, nor can other objects be comfortably held while the person is operating the computer this way. When finished, the stylus must be replaced in the board and the whole thing put in a pocket. This limits the usefulness of a stylus-based system to nomadic computing.

Arm-mounted devices are located either on the non-dominant arm, or below the wrist of the dominant arm. Devices on the arm, like hand-held systems, require two hands (or one hand and one arm) for a one handed task, which is wasteful. This system is very quick to start up and put away, but the "free" hand can do little more than hold small objects. A board mounted below the wrist of the dominant hand frees the other arm for various tasks, but limits the actions of the dominant hand when not in use.

Body-mounted devices are often attached to the torso or a belt. Like arm mounted devices, these are very quick to start and stop using. Also, they do not limit one's actions either while using or not using the device. The major problem with these is ergonomic. Is the device in easy reach? How difficult is it to hold the arm or hand in the same position for long periods? Is the position of the device adjustable on the fly? All these need to be answered to determine if the device is comfortable enough to use for the long periods of time one is likely to want from a mobile computer. It becomes clear that arm and body mounted boards can be used for continuous mobile computing, but at the cost of imposing constraints on the user's actions or their comfort.

The above discussion gives an idea of what one would want from an input for a wearable computer. The input would have be very fast to start using, preferably the just time it takes to grab a device or press a button. The time it takes to put away should also be minimised, making the switch from real-world interactions to computer interactions as seamless as possible. The device should be as body-efficient as possible. This means it should leave as much of the body as it can to perform real-world tasks. Ideally the device would be totally passive and not noticeable when not in use, but could be called into use or put away at any time without much effort, while still allowing simultaneous interaction with the real world. Lastly the device should be ergonomically sound. It should not cause discomfort or fatigue after extended use. Given that board-based devices tend to be limited in these aspects, it would be a good idea to look for alternative inputs which do not require boards.

2.5 Summary

Several types of graphic and text inputs devices were reviewed in this chapter in the search for a continuous mobile interface for a wearable computer. Most input devices use boards, which limits their portability. Voice was discounted because of the poor pointer control and the need for backup input devices in situations where speech is not an option. Video-based eye tracking was discounted because of problems with involuntary eye motion and selection. This leaves glove-mounted and bioelectric-based input devices as the most suitable solutions.

In the next chapter we introduce two new input devices which meet our requirements for a mobile interface. The first is a text input device made from a chord keyboard mounted on a glove. The second is a pointer controlled by the bioelectric signals generated by wrist motion. These are designed to be lightweight, easy to use and wear, and sufficient for use with a wearable computer.

Chapter 3

Wearable Text and Graphic Input Devices

3.1 Introduction

The purpose of this chapter is to introduce two new computer input devices specifically designed for interaction in a mobile environment. The text input device is called the Chording Glove, a glove-mounted chord keyboard which requires nothing more than a solid surface to tap the fingers against. The graphic input device is called the Biofeedback Pointer. This uses bioamplifiers to detect and recognise muscle movements in the lower arm in order to control a pointer. In this chapter we will describe both devices in detail, including their hardware, software and their usage.

3.2 The Chording Glove

While keyboards are highly efficient when used on a desktop, they have been found to be quite difficult to use with a mobile computer. Chord keyboards can be made much smaller than a standard keyboard, with no loss of performance. The main restricting factor of chord keyboards is that, despite the small number of keys, the board must be nearly hand-sized to allow the user to chord comfortably. Soft keyboards operated with a stylus have been found to be the most usable keyboard for mobile system, since they can be made smaller than chord keyboards. The biggest problem is that the keyboard can take up a large amount of the display. Handwriting recognition, with a soft keyboard as a backup has been found to be much more acceptable and is currently is common use on many handheld systems. However, a boardless text interface is a necessity in order to use a continuous mobile computer. Contact gloves can be used as a boardless text interface by translating gestures into characters. While the system can be used for continuous mobility, the gestures must be made with a high level of accuracy in order to be correctly recognised.

Another boardless interface can be made by reducing a chord keyboard to its most basic parts. The board can be thrown away and the chord keys mounted on the fingers themselves. This is exactly what the Chording Glove does. This combines the benefits of a chord keyboard with the portability of a contact glove. A chord keyboard can be used for general text input, like a standard keyboard. It can be learned quickly and retained in long term memory. A contact glove is uncumbersome, since it is just another part of the users clothing. It is quick to activate and use and to stop and put away.

3.2.1 The Hardware

The Chording Glove has three basic parts. These are the Finger Sensors, the Shift Buttons, and the Function Keys (Figure 3.1).

Finger sensors

The finger sensors are small pressure sensitive triggers sewn to the glove at each fingertip. These sensors are designed to activate when pressed against any solid surface.

The thumb is intended to have two sensors, one at the tip and one on the side. The sensors are in series, so pressing either activates it. Different positions on the thumb are used to chord depending on the orientation of the hand. When the hand is against a flat surface, the side is used. When the hand is chording against a curved surface (like a coffee mug) the tip is used. The prototype does not include the second thumb sensor because the current sensor design is too large to accommodate both sensors. The

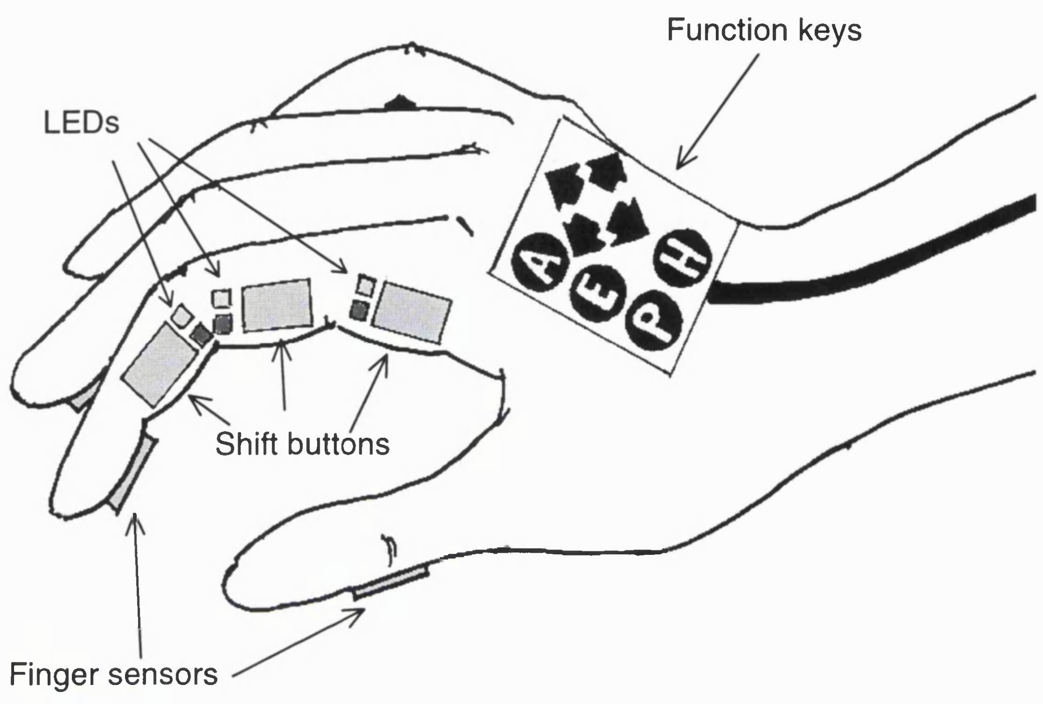


Figure 3.1: Side view of the Chording Glove

sensor on the side of the hand is used because chording against a flat surface is more common.

Several sensor designs were tested before settling on the Hard Copper Plate finger sensor (see Appendix C). The Hard Copper Plate design is a digital button (Figure3.2(a)). Two plastic rectangular plates are connected by a spring at each end. Thin sheets of copper are glued to each piece of plastic backing. Wires are soldered to the copper plates. Additional solder acts as a conductive extension of the copper plates. The entire button is encased in insulating tape to prevent damage and facilitate connecting to the glove.

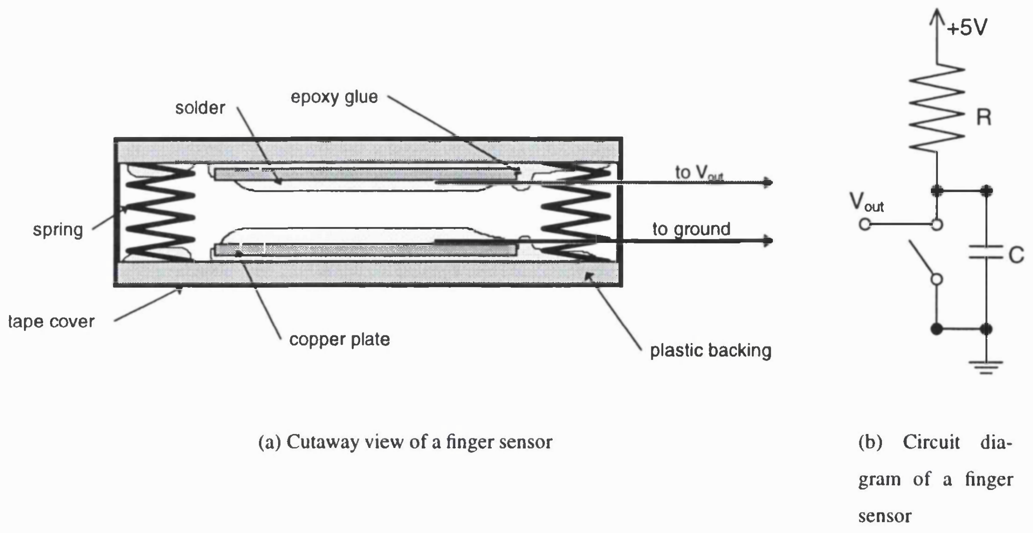
One plate is grounded and the other is connected to 5 volts through a resistor. Compressing the sensor, makes the plates touch, grounding both. Bounce noise is removed by a 48Hz low pass filter (Figure3.2(b)).

Shift buttons

The shift buttons are used to change each shift state: Caps, Number, and Control. When pressed, the shift operates on the next character[1]. When double pressed the shift operates on all characters until pressed again. There is a green LED next to each shift button which lights when the shift is On. A red LED lights when the shift is Locked.

Two designs were tested for the shift buttons (see Appendix C). The first was Taped Aluminium Foil Plates, but these were found to break too easily and the wires would constantly be pulled loose because they could not be soldered in place. The aluminium foil was replaced by copper foil, which was found to

[1]Unless the next chord made is `<Space>`, `<BackSpace>`, or `<Return>`. If one of these is made, the shift stays on until another chord is made.



(a) Cutaway view of a finger sensor

(b) Circuit diagram of a finger sensor

Figure 3.2: The finger sensor

be sufficiently rugged and could be soldered to the wires to keep them in place.

A shift button is made of two small square pieces of copper foil, separated by a thick piece of paper with a hole in the middle. One piece of foil is grounded and the other is connected to 5 volts through a resistor. The foil is flexible enough to connect when pressed, and will return to its original position when released. when the foil pieces connect, they are both grounded, activating the sensor. Bounce noise is removed by a 48Hz low pass filter. This has the same circuit diagram as the finger sensor (Figure3.2(b)).

Function Keys

The function keys are seldom used buttons which are located on the back of the glove. These are intended to be pressed by the other hand. These buttons perform the following operations:

`<Pause>` - This causes all the finger sensors to be ignored until this key is pressed again. This is to allow the hand to perform other actions without accidental chording.

`<Escape>` - This has the same effect as on a normal keyboard.

`<Help>` - A single press of the `<Help>` key will call an application sensitive help function and a double press displays the chord keymap. The keymap is displayed until `<Help>` is pressed again. This allows the user to be able to look up chords at any time.

`<AutoCaps Toggle>` - This turns on and off the AutoCaps feature. The AutoCaps feature automatically capitalises the first letter of a sentence. This is to save effort of using the shift at the start of each sentence. The Caps shift turns on automatically whenever a sentence ending character is chorded: `<Period>` (`.`), `<Exclamation Point>` (`!`), and `<Question Mark>` (`?`). There is an LED next to this button which indicates if it is on or not.

Arrow Keys - The four keys: `<Up>`, `<Down>`, `<Left>`, and `<Right>` provide basic pointer control, in the same way as on a normal keyboard.

The function keys have the same design as the shift buttons. They are made of foil separated by thick paper with holes for each key. One layer of foil is grounded and the other is connected to 5 volts through a resistor. Bounce noise is removed by a 48Hz low pass filter. The circuit diagram is the same as the finger sensors (Figure3.2(b)).

A more detailed description of the Chording Glove's hardware can be found in Appendix C.

3.2.2 Chord Recognition

There are a variety of ways to programme the chord recognition. Some chord keyboards only have one manner of chord recognition. The recognition starts when the first finger is pressed and stops when the last finger is released. All the fingers pressed in this period are used to generate the chord (Roberts, 1995).

The Chording Glove has a different method of chord recognition. The computer reads the state of the Chording Glove every clock tick (approximately 55ms). If a chord changes or a timeout occurs, a chord is generated in one of three possible ways, *press and hold*, *press and release*, and *change*.

Press and Hold The first case is to make the chord and keep it pressed. When the first finger is pressed, the computer waits 5 ticks. If no fingers are released within those 5 ticks, the computer registers the chord generated from the pressed fingers and outputs it to the keyboard buffer. If the chord remains unchanged for 10 ticks, it is repeated every 2 ticks until it is released.

Press and Release The second case is to make the chord and release it before 5 ticks have elapsed. As above, when the first finger is pressed, the computer waits 5 ticks. If a finger is released before that period the chord generated from all the pressed fingers (before one was removed) is sent to the keyboard buffer. The computer then waits another 5 ticks. If all the fingers are released in this period, the computer stops waiting and is ready to start a new chord. Otherwise, if there are fingers still pressed after 5 ticks, that chord is sent to the keyboard buffer. If this new chord is held, it is repeated as above: first after 10 ticks and then after every 2 ticks.

Change The third case is to change the fingers after a chord is recognised. When the first finger is either added or removed, the computer waits 5 ticks, ignoring any finger changes in that period (as in the Press and Release case). At the end of this time, if any fingers are still pressed, the chord is sent to the keyboard buffer.

The Chording Glove's method for recognising chords has a slight advantage. Instead of needing to release all the fingers after each chord, a new chord can be made by changing the necessary fingers. This can be used to full advantage by the keymap by making frequent digrams and trigrams out of similar chords. This is very similar to having a one character overlap on a standard keyboard. Allowing this "overlap" can potentially speed up the text input rate.

3.2.3 The Chord Keymap

One of the most difficult parts of creating a new keyboard is determining the layout of the keys. This problem is amplified for a chord keyboard, because the user needs to memorise the key positions before they can effectively use the device. The characters associated with all the chords is called the keymap. When generating a new keymap there are five criteria that should be kept in mind.

1. Easy to learn – A good chord keymap can be learned for basic use in less than an hour. It is very important that the chords are easy to memorise, because the user cannot use the keyboard without knowing the all the chords. The time it takes to learn will be proportional to the difficulty in getting people to use the keyboard.

2. Easy to remember – Once learned, a good keymap should stay in long-term memory, even if not used for significant periods of time. There are several ways to help learn the keymap and keep it memorised. Some examples are:

1. Chords can be related to keyboards already learned. For example, a chord can be based on a motion usually used in touch typing, such as using the thumb to press space. This can also be detrimental as it is possible to have the two keyboards interfere with each other, making it difficult to switch between the two keyboards.
2. Chords can be related to sign languages learned. This is similar to being related to keyboards. The advantage is that if the user already knows the sign language, there are less new chords that must be learned.
3. Chords can follow a logical pattern. If chords which are usually typed together (common digrams or trigrams) follow a simple pattern, they are easier to remember. Repeated, rhythmical motion tend to say in memory longer than random, disjointed motion.

3. Minimal work – Because a chord keyboard does not require any hand motion, all motion is concentrated in the fingers. This could cause strain in weak fingers. In order to reduce strain on the user, the finger work should be biased towards the strongest fingers. This reduces muscle strain which could lead to injury.

4. Low error – There is a fine line between where similar chords help memorisation and where they cause confusion. A good rule of thumb to use is that the character which is simpler in shape should have a simpler chord. For example, if the chords for `U` and `W` are related, it would be better to have `W` use more fingers than `U`. Similarly, since `<Comma>` is just a `<Period>` with an appendage, if the chords are similar the `<Comma>` should use more fingers. Another good rule is to have the less frequently typed characters be the less comfortable chords. This reduces strain while reducing errors.

Fast typing – Fast typing is often related to productivity. A user will not want to switch to a new keyboard if their productivity will drop considerably. Chords can be made faster to enter by making trigrams and digrams created by fluid motions. As mentioned above this also aids in memorisation.

Once the keymap is designed, it must go through several iterations of testing and revising before it reaches its final form. Once the final form is settled, it is necessary to test it, both theoretically and empirically.

Developing a Keymap

The first step in creating a new chord keymap is to find which characters are most frequently used. It is possible to find tables of letter probabilities in books on cryptography (Seberry & Pieprzyk, 1988; Denning, 1982; Konheim, 1981). Unfortunately these tend to only give the relative frequency of letters. To make a keymap, all possible typed characters are needed. This includes numbers, punctuation, and control characters. The easiest way to discover this is to collect a large sample of text and count the frequency of characters.

The text which is analysed should contain samples of all possible input which would be typed on a keyboard. This includes normal English, computer languages, operating system commands, data, etc in roughly the proportions that they would be expected to be typed.

The next step in generating a keymap is to divide up the characters in groups. Since there are more typable characters than chord combinations, some way must be found to increase the number of chords. The methods for increasing the number of chords are described in Section 2.2.3. There are two basic concepts behind all these methods. The first concept is to add extra keys. This can be accomplished by using two hands, or by having multiple keys for each finger. The second concept is to add shifts. This can be done by adding extra keys for the thumb, or using sticky shift keys.

Using shift keys has the advantage of dividing the keymap into smaller groups. If divided properly, the smaller groups could be easier to memorise than one large group. The chord groups should be defined using three criteria:

1. All the characters must have a high chance of being typed together. This minimises the need to switch between groups.

2. The characters typed with no shifts on must have the highest frequency. The characters typed with two shifts on must have the lowest frequency.

3. Each group of characters must be obvious and coherent.

The third step in generating the chord keymap is to rate the 31 combinations by comfort. This was done in Seibel (1962). Seibel measured the time it takes to make a chord once a subject is told to make the chord. This is called the Discrimination Reaction Time (DRT). The chance of error of each chord was measured and was found to be roughly proportional to the DRT. When the chords are sorted by the DRT, this gives a list of the most to least comfortable chords.

The next step is to line up the characters on the sorted chord list, with the most frequent characters made by the most comfortable chords. This is now the initial keymap. After this is done, the chords are modified to take into account mnemonics to help remember the chords. This is done is several ways:

1. The chord can resemble the character. The finger combination can have some obvious relation to the shape of the character typed.

2. A sequence of chords can have some meaning and be easy to make. For example a common sequence like `t` $\rightarrow$ `h` $\rightarrow$ `e` can be made a simple sequence of chords, which are easy to make together.

3. One chord can be based on another chord. For example, similar characters, like `<Comma>` and `<Period>`, could have similar chords, with only one or two finger changes.
4. A shifted chord can be based on an unshifted chord. For example the chord for `<Exclamation Point>` could be the shifted version of the chord for `e`.
5. Sequential characters (i.e. numbers) can be made by following a simple pattern.

This is not a comprehensive list, for there are other logical ways to map which can assist in memorisation. Other mnemonics would have to depend on the shift groups and the physical layout of the chord keyboard.

In summary, the process for creating a chord keymap is:

1. Determine character frequency
2. Separate the characters into logical groups
3. Rate chords by comfort
4. Assign characters to chords
5. Revise using mnemonics

Following each of these steps in order will give a good keymap for a specific chord keyboard. There is no one best keymap for a chord keyboard, for many solutions may work equally well. In order to find the quality of the keymap created by following these steps, the keymap must be tested.

Testing the keymap

There are two ways to test the keymap: theoretically and empirically. Of the five criteria mentioned at the start of this section (Page 60), three can be tested empirically: relative finger work, speed and error. We would want a keymap to use the strongest fingers most and the weakest fingers least. One way to determine he relative finger use is to add up all the frequencies of characters made using each finger. Normalising the results will show which fingers are used most and which are used least. The thumb is the strongest finger and can handle larger amounts of work than the other fingers. The thumb would hopefully be used most. The index finger is the next strongest, after the thumb, and should be used almost as much as the thumb. The little finger is the weakest and should be used least. This test gives results which make it quick and easy to compare different keymaps. There are limits to this test, in that it will only determine which fingers are used most, not the ease of the finger combinations. Thus the results of this test must be taken with a grain of salt.

Another theoretical test for the keymap is to find out the speed of the keymap. The theoretical input speed can be approximated by averaging Seibel's DRTs, and weighting each DRT by the frequency of the associated character. This is a heuristic method for estimating the potential chording speed, and although the model is somewhat physiologically flawed, it does give a rough idea of speeds that might be expected. The theoretical error can be calculated the same way using Seibel's errors.

In Seibel (1962), the DRTs were calculated by measuring the time it took for a subject to make a chord once asked to do so. The chords were displayed randomly and were not associated with any characters. When using chords for text entry, the following character is known in advance, giving the subject a slight time advantage, as knowing the next character in advance allows the user to plan the next chord before the previous one is finished. This implies that it is possible to achieve faster speeds than the theoretical input speed. Further study must be carried out to show the validity of this model.

A common empirical method for testing the keymap is to have several people learn it and measure their progress (Gopher & Raij, 1988; Kirschenbaum et al., 1986). This would give the time it takes to learn the keymap, which is very important in determining how likely people are to use the keyboard. In addition, input speed and rate of increase can be measured. This determines how easy it is to use the keymap.

Developing the Chording Glove Keymap

By using guidelines described above, a new keymap can be created. The following is a step-by-step description of how the chord keymap was made for the Chording Glove.

In order to clarify the finger patterns used in the chords, we will be using a graphical notation to specify the chord. In this notation, the thumb is on the left and lower than the rest of the fingers. A filled dot indicates that the finger is pressed, and an empty dot indicates the finger is not used in the chord. For example, the notation $_{\circ}\bullet\ \circ\ \bullet\circ$ refers to a chord made by the index and ring fingers. This notation will be used throughout the rest of this thesis.

Determine character frequency The probabilities of all the characters in the ASCII character set were calculated by counting their appearances in a large amount of text in various formats. Details of texts and probability results can be found in Appendix B.

Divide the characters into logical groups There are 100 typable ASCII characters. This requires at least two shift keys. This yields 124 possible chord combinations, which is more than enough. This allows a few characters to be in more than one group.

The 100 ASCII characters divide well into four groups defined as follows:

Lower Case - this is the default group, which is entered with no shifts on.
`a b c d e f g h i j k l m n o p q r s t u v w x y z . , <Space> <BackSpace> <Return>`

Upper Case - these tend to be the next most common, and should be entered with just the first shift key (*caps*) on.
`A B C D E F G H I J K L M N O P Q R S T U V W X Y Z . , <Space> <BackSpace> <Return>`

Numbers - this group consists of numbers and maths-related punctuation. This is the next most common group, so it is entered with just the second shift (*num*) on.
`0 1 2 3 4 5 6 7 8 9 + - = * / ( ) ^ ~ . , <Tab> <Space> <BackSpace> <Return>`

Punctuation - This is all the rest of the characters. Since these are the least frequent, they are made by using both shifts (*caps* + *num*).
``! ? _ " ' ` < > @ ^ { } # % ~ | $ \ <Tab> : ; <Space> <BackSpace> <Return>``

Each mode can be locked on if the user needs to enter a several characters from the same group in one stretch. Since `<Space>`, `<BackSpace>` and `<Return>` are likely to be used with any group, it stands to reason that they should be used in all groups.

Table 3.1: The evolution of the Chording Glove keymap. Any character in **bold** has been changed from the previous step.

Chord	1st Guess		Iteration 1			Iteration 2			Iteration 3			Iteration 4			Final Keymap		
	letter	num.	punctuation			shape			sequences			similarity					
○ ○ ○ ●○	**E**	**3**	E	3	**!**	E	3	!	E	3	!	E	3	!	E	3	!
○ ○ ● ○○	**T**	**2**	T	2		T	2		**H**	2		H	2		H	2	**@**
● ○ ○ ○○	<Space>		<Space>			<Space>			<Space>			<Space>			<Space>		
○ ● ○ ○○	**O**	**1**	O	1	|	**R**	1	**’**	**T**	1	’	T	1	’	T	1	’
○ ○ ○ ○●	**A**	**4**	A	4	**&**	**I**	4	**?**	I	4	?	I	4	?	I	4	?
● ○ ○ ●○	**N**	**7**	N	7		**O**	7	|	O	7	|	O	7	|	O	7	|
○ ● ● ○○	**R**		R			**N**		**”**	N		”	N	**(**	”	N	(	”
○ ○ ● ●○	**I**		I		**?**	**A**		**&**	**G**		**>**	G	**)**	>	G	)	>
● ● ○ ○○	**S**	**5**	S	5	**$**	S	5	$	**A**	5	**&**	A	5	&	A	5	&
○ ● ● ●○	**H**		**L**	**=**	**<**	**M**	**-**	_	M	-	_	M	-	_	M	-	_
● ○ ● ○○	**L**	**6**	**H**	6		H	6		**R**	6		**S**	6	**$**	S	6	$
● ● ● ●○	**D**	**0**	**G**	0	**>**	**C**	0	**%**	C	0	%	C	0	%	C	0	%
● ● ● ○○	**C**	**9**	C	9	**%**	**G**	9	**>**	**S**	9	**$**	**R**	9	**‘**	R	9	‘
● ○ ○ ○●	**U**	**8**	U	8		**Y**	8		Y	8		Y	8		Y	8	
○ ○ ○ ●●	<BackSpace>		<BackSpace>			<BackSpace>			<BackSpace>			<BackSpace>			<BackSpace>		
○ ● ○ ●○	**P**		P	**+**	**#**	P	+	#	P	+	#	P	+	#	P	+	#
○ ● ● ●●	**M**		M	**-**	_	**L**	**=**	**<**	L	=	<	L	=	<	L	=	<
● ○ ● ●○	**G**		**D**	**/**	\	D	/	\	D	/	\	D	/	\	D	/	\
○ ○ ● ●●	<Period>		.	.	**:**	.	.	:	.	.	:	.	.	:	.	.	:
● ● ● ●●	<Return>		<Return>			<Return>			<Return>			<Return>			<Return>		
○ ● ○ ○●	**F**		F			**U**	**[**	**{**	U	[	{	U	[	{	U	[	{
● ○ ○ ●●	**Y**		Y			**F**			F			F			F	**<Tab>**	**<Tab>**
● ● ○ ●○	**B**		B			B			B			B			B	*	*
● ○ ● ●●	**W**		W			W			W			**,**	**,**	**;**	,	,	;
● ● ○ ○●	<Comma>		,	,	**;**	,	,	;	,	,	;	**W**	**]**	**}**	W	]	}
○ ○ ● ○●	**K**		K	^	^	K	^	^	K	^	^	K	^	^	K	^	^
● ● ● ○●	**V**		V			V			V			**X**	*	*	**J**		
● ○ ● ○●	**X**		X	*	*	X	*	*	X	*	*	**Z**			Z		
○ ● ○ ●●	**Z**		Z			Z			Z			**V**			V	<Escape>	<Escape>
● ● ○ ●●	**J**		J			J			J			J			**X**	*	*
○ ● ● ○●	**Q**		Q			Q			Q			Q			Q	**?**	**?**

The mapping of the capital letters is exactly the same as the lower case letters, including `<Period>` and `<Comma>`. `<Period>` and `<Comma>` have the same mapping in the numerical characters, but are replaced by `<Colon>` and `<Semicolon>` in punctuation mode. Each mode has 31 or fewer characters, with the most common characters made without any shifts and the least common made with two. This satisfies the three criteria for grouping characters.

Rate chords by comfort Seibel's (1962) list of DRTs was used to rate the chords. The chords in table 3.1 are ordered using his results, from most comfortable, to least comfortable.

Assigning characters to chords Assigning characters to chords can be done in four simple steps.

The First Guess: Blind Association The first step is allocating chords for `<Space>`, `<BackSpace>`, and `<Return>`. The following chords were chosen for them:

●○ ○ ○○	`<Space>`	This is the same way `<Space>` is typed on a keyboard, with just the thumb.
○○ ○ ●●	`<BackSpace>`	The `<BackSpace>` is often on the rightmost side of the keyboard. Furthermore, having `<BackSpace>` as the last two fingers contrasts well with `<Space>` space being made with the first finger.
●● ● ●●	`<Return>`	Using all fingers simultaneously is easy to remember and has a sense of finality which is good to associate with `<Return>`.

The easiest way to start is to blindly assign the characters to the chords, aligning the highest probabilities with the lowest DRTs. This is first done for letters (both lower and upper case are the same). The initial keymap can be seen in the two columns under 'First Guess' in Table 3.1.

Numbers follow a special pattern to facilitate memorisation and use. This was generated by using an increasing pattern of the fingers and the thumb as shown in Table 3.2. Not only is this logical, but all the chords are in the easier-to-make half of the DRT list.

Table 3.2: The chord patterns for numbers

	Index	**Middle**	**Ring**	**Little**
No thumb	1	2	3	4
Thumb	5	6	7	8
	9			
	0			
	`<Return>`			

After the initial keymap is made, it is repeatedly revised using the methods described above in an iterative process in which each step brings it closer and closer to a workable keymap. Four steps are carried out in between the initial guess and the final keymap. Table 3.1 shows the changes in Chording Glove's keymap for each of these iterations.

The First Iteration: Punctuation Since the characters in the shifted states are not as common as the unshifted characters (letters), assigning them should not be weighted by the probability but by their relation to the unshifted characters. Table 3.3 details the mnemonic relationships used in creating the keymap.

Table 3.3: Mnemonic relations between shifted and unshifted characters

Character	Punctuation	Mnemonic	Comments
A	&	**A**nd	
B	*	**B**y	*as in "multiply 4 **b**y 5"*
X	*		*similar appearance*
C	%	Per**C**ent	
D	/	**D**ivision sign	
/	\		*similar appearance*
E	!	**E**xclamation point	
G	>	**G**reater than	
I	?	**I**nquiry	
Q	?	**Q**uestion	
K	^	**K**arat	
L	=	Equa**l**	
	<	**L**ess than	
M	-	**M**inus	
-	_		*similar appearance*
O	|	**O**r	*used in programming languages like C*
P	+	**P**lus	
	#	**P**ound sign	*also similar in appearance to +*
S	$		*similar appearance*
.	:		*similar appearance*
,	;		*similar appearance*

The Second Iteration: Shape A chord which resembles the character it makes is easier to remember. The characters in the keymap are switched around to allow as many shape relations as possible. The letters `I`, `M`, `N`, and `Y` are based on the letters in American Sign Language (Figure 3.3). Other letters are based on the hand shape resembling the chord, such as `C`, `F` ,`L`, `R`, `U`, `<Quote>`, and `<Apostrophe>` (Figure 3.4).

The Third Iteration: Sequences In the next iteration chords are chosen to facilitate creating common digrams or trigrams. This is done by making minimal finger changes between letters which are likely to be used in sequence. For example, `t` $\dashrightarrow$ `h` $\dashrightarrow$ `e` is made by the index finger followed by the middle and then the ring fingers. It is the most common trigram and it is one of the easiest chord sequences to make. Table 3.4 shows the chords for some common sequences.

The Fourth Iteration: Similarity It facilitates memorisation if similar characters have similar chords. For example, `<Comma>` is a `<Period>` with an added appendage. The chords should reflect this, by having the chord for `<Comma>` be the same as for `<Period>`, but with the thumb added. Table 3.5 shows the complete list of chords which are based on other chords.

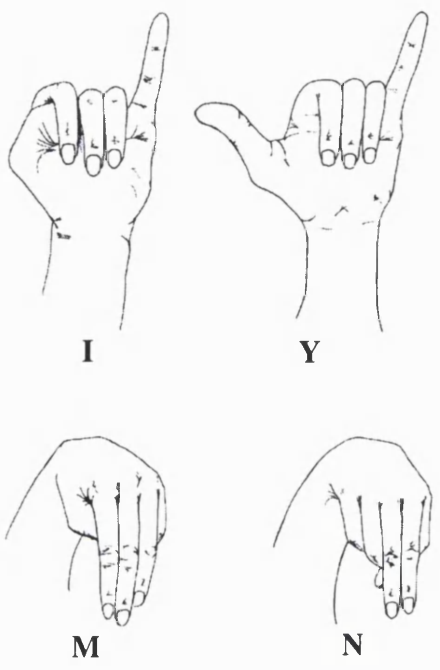


Figure 3.3: The hand positions in American Sign Language for the letters I, M, N, and Y

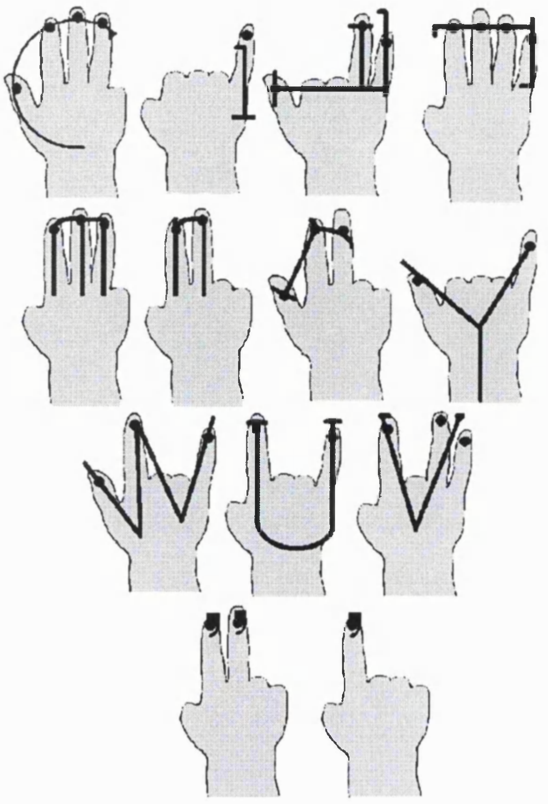


Figure 3.4: The chords which resemble the characters they make. Note: the fingers not shown are not bent forward, they merely do not touch the surface. They are removed from the diagram for clarity.

Table 3.4: Common trigrams and their chords

t → h → e		i → n → g		q → u	
t	○● ○ ○○	i	○○ ○ ○●	q	○● ● ○●
h	○○ ● ○○	n	○● ● ○○	u	○● ○ ○●
e	○○ ○ ●○	g	○○ ● ●○		

Table 3.5: Chords related to other chords

U	○● ○ ○●	S	●○ ● ○○	.	○○ ● ●●	'	○● ○ ○○
W	●● ○ ○●	Z	●○ ● ○●	,	●○ ● ●●	"	○● ● ○○
V	○● ○ ●●					`	●● ● ○○
[	○● ○ ○●	{	○● ○ ○●	(	○● ● ○○		
]	●● ○ ○●	}	●● ○ ○●	)	○○ ● ●○		

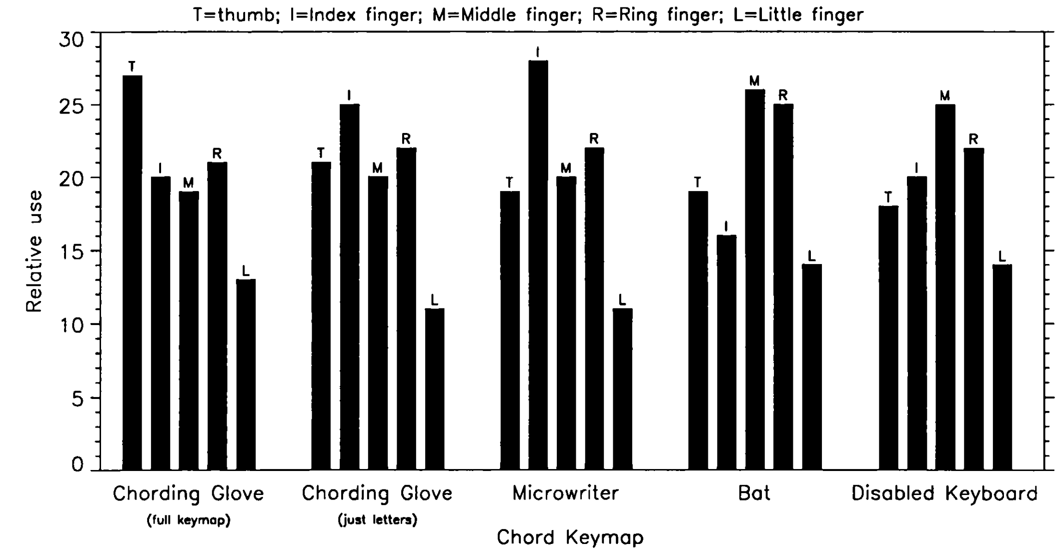
The Final Keymap The final step in generating the keymap is to switch any last few characters which are still in an inconvenient position and then place all the left-over characters. First X and J were switched because, in practice it was found that the chord ●● ○ ●● was easier to make than ●● ● ○●, and thus was associated with the more common letter. Next, @ and ~ are assigned to the easiest punctuation chords still available. Then secondary chords were found for * and ? because there were no shifted chords already assigned to `B` and `Q` and it seemed logically appropriate. Lastly the control characters <`Tab`> and <`Escape`> were arbitrarily placed in the easiest shifted chords still available.

Theoretical Testing As mentioned above, the keymap can be tested by determining the relative finger usage. The normalised sums of frequencies for each finger are shown in Table 3.6 and displayed as a bar graph in Figure 3.5. Each keymap is normalised to that the sum of the relative use of each finger is 100%. This is to enable a comparison of the relative work for each finger. Because of the differences between the methods for entering special characters on the various chord keyboards, only the chords for the letters were used. Using the full keymap will yield slightly different results.

Table 3.6: Normalised finger work distribution

Keymap used	Thumb	Index Finger	Middle Finger	Ring Finger	Little Finger
Chording Glove (full keymap)	27%	20%	19%	21%	13%
Chording Glove (just letters)	21%	25%	20%	22%	11%
Microwriter (Roberts, 1995)	19%	28%	20%	22%	11%
Infogrip's Bat (Microsoft Corporation, 1995)	19%	16%	26%	25%	14%
Disabled Keyboard (Kirschenbaum et al., 1986)	18%	20%	25%	22%	14%

Figure 3.5: The relative work performed by each finger for different keymaps



The results are favourable: the index finger performs a majority of the work. This is good because the index finger is relatively strong. The little finger is weak and, as designed, does the least work. The other fingers roughly share the rest of the work equally. However, since <`Space`> makes up approximately 15% of all characters typed, and each of the keymaps use just the thumb for <`Space`>, the work values for thumb would be noticeably increased if <Space> were included. This is obvious by comparing the results for the Chording Glove's full keymap to the one for just the letters.

The other methods mentioned above for testing the keymap are finding the theoretical input speed and error rate. These values for various chord keymaps are listed in Table 3.7. Like the situation with work distribution, comparing keymaps is difficult because they do not entirely share the same input mechanisms. Therefore only the chords for letters are used. Again, the full keymap yields slightly different results.

The last three entries are used for a comparison only. *Best possible* show the results from the fastest DRTs associated with the most frequent letters, and the lowest errors associated with the most frequent letters. Note that these are not the same keymap, just the best possible values for these attributes. *Best random* is generated from the average value of the fastest 26 DRTs and lowest errors, which would tend to be the values of a randomly generated keymap. *Worst possible* is results from the slowest DRTs and highest errors associated with the most frequent letters.

Table 3.7: Theoretical Input Speed and Error Rate

Keymap used	**Speed**		**Errors**
	in ms	in WPM	
Chording Glove (full keymap)	305	39.3	7.1%
Chording Glove (just letters)	306	39.1	7.0%
Microwriter (Roberts, 1995)	308	38.9	8.1%
Infogrip's Bat (Microsoft Corporation, 1995)	313	38.3	7.3%
Disabled Keyboard (Kirschenbaum et al., 1986)	305	39.4	6.7%
Best possible	302	39.7	5.3%
Best random	313	38.3	7.8%
Worst possible	336	35.7	15.2%

The results show that the Chording Glove achieves its design specifications of having a fast input speed and low error rate. It also compares favourably with all but the Disabled Keyboard, which rates only slightly better. Note the relatively small range of speeds for the various keymaps. The difference between 'Best possible' and 'Worst possible' is only 4wpm. Like with the alternative layouts on standard keyboards, there is not much difference in input speed. The big differences are in finger work and error rates, and can be seen in the graphs in Figure 3.5, and in the 'Errors' column in Table 3.7.

Alternative Keymaps

The Chording Glove's keymap is designed for general English text input. This is not the only situation in which the Chording Glove can be used. Entering text in a foreign language is the most obvious reason for changing the keymap, but there are others. Changing the keymap on a standard keyboard would require either changing or ignoring the labels on the keys, neither of which is particularly desirable. If a user does create a customised layout on their own computer, interference between the layouts would cause their performance on anyone else's computer to be worse than if they never learned their new layout. Since the chord recognition and display of the chord list are both done in software, changing the Chording Glove's keymap can be as easy as loading a new file. Consequently, choosing a new keymap for use with another language, specialised application, or even for reasons of personal preference, becomes a trivial task, effectively equivalent to adding an unlimited number of customisable shift states. The portable nature of the Chording Glove avoids the problems of using a non-customised device. Since the Chording Glove can be taken with the user, it can be just plugged into the new machine and used normally.

While creating a new keymap is not trivial, it is hoped that the method provided here will greatly simplify the process, allowing specialised keymaps to be easily created for any desired purpose. One example could be to make a keymap intended to write Fortran programs. Since the language is not case-sensitive, one could switch the caps and punctuation modes. This way, entering common punctuation like `$` and `'` would be easier since they would require only one shift, while switching to capital letters, a less common task, would be harder, requiring two shifts. Any set of 124 characters (plus another 124 control characters) can be created in order to maximise the efficiency of the user's text entry.

3.2.4 Uses for the Chording Glove

The combination of high mobility and the chording-style input makes the Chording Glove particularly suited to certain tasks. One advantage of the Chording Glove is the fact that, as a chord keyboard, no visual supervision is necessary. This means that the Chording Glove can be used to enter text in situations where one must pay attention to outside events, such as taking notes, driving, or walking in difficult terrain. The display can take the form of either a heads-up display, or even an audio display, to provide feedback to the user without interrupting their other activities.

The one-handed nature of the Chording Glove frees the other hand to perform separate or parallel tasks. For example, the other hand could be used for real-world tasks, such as operating light machinery, carrying objects, etc. For parallel tasks, the other hand could operate a graphic input device to allow simultaneous text entry and graphical interaction. Performing separate tasks in parallel can have improved performance over performing the tasks separately. Experiments have shown this to work for two graphic inputs devices in parallel (Buxton & Myers, 1986; Kabbash et al., 1994), but it may work for graphic and text tasks as well.

Using only one hand has the disadvantage of a slower text input rate as compared to a full-size desktop keyboard. This means the Chording Glove should be used in situations where a full-size desktop keyboard is inconvenient, such as in a mobile environment. Alternatively, it could be used in situations where speed is not an issue, such as in casual text entry tasks like interacting with the operating system, searching or reading text, and possibly basic text composition. This issue is discussed in more detail in Section 6.2.3.

The Chording Glove has the potential to avoid some of the problems linked to RSI. Touch typing on standard keyboard requires the hands to be held in an unnatural position for long periods of time. There is no 'standard' hand orientation for the Chording Glove. The user can position their hand in any manner they find comfortable, as long as they have a surface to chord against. This position can be easily changed if it becomes uncomfortable after a while.

Suggestions for specific applications for the Chording Glove can be found in Section 3.4.

3.3 The Biofeedback Pointer

In addition to a method for entering text in a mobile environment, there must be a method for pointing and selection. Again, this must be highly economical in its use of space, and be accurate enough for day-to-day interactions - such as those associated with word processing, browsing and data input. Desktop interfaces such as the mouse use up too much space to be of any use. Pointers designed for portable computers such as trackballs, trackpads and trackpoints are board-mounted and while they take up very little space, they still require holding or manipulating something. Glove-mounted virtual reality-style graphic input devices take up no space and do not require anything to be held, but they tend to be too complicated and expensive for everyday use as a simple two-dimensional pointer.

The speed and power of modern computers makes it possible to measure and analyse bioelectric data in real time. The extra hardware required to do this is little more than a few bioamplifiers and a medium to fast analogue to digital converter. The user tapes electrodes and amplifiers to their forearm, which are connected via a junction on the belt to the computer. This is light weight and uncumbersome. By using this device the user can control a pointer simply by moving their hand. Selection can be accomplished by a set of buttons on the side of the index finger which can be pressed by the thumb. These are provided by the Chording Glove. When in text mode the buttons act as shifts. In pointer mode, the buttons act as selection buttons.

3.3.1 Measuring Bioelectric Signals

EEG, GSR, and EMG were all considered as possible bases for the Biofeedback Pointer. The advantages and disadvantages of each were explored in detail and are described below.

An EEG is a collection of low frequency and low amplitude waveforms generated by electrical activity in the brain. Low amplitudes require better amplifiers and noise filtering. The low frequencies can limit the rate of information transmission, reducing the speed of an input device. The potential discomfort caused by the liquid gel required to make the ohmic connection though the hair makes measuring EEGs even more difficult. GSRs, on the other hand, are large enough to be fairly easy to read under most conditions. However, GSR is extremely low frequency, which places limits on its usability as an input device. The GSR is sensitive to imperfections (cuts, scars, etc) and perspiration in the skin, degrading the potential performance of the device. These same imperfections cause no problems with sensing EEGs and EMGs, and in some case imperfections actually *improve* the signals (Geddes, 1972).

An input device based on EMGs needs to be able to detect skin voltages on the order of microvolts (Carroll, 1984). The frequencies in question are, for the most part, less than 500Hz, fast enough to not limit the input speed, while slow enough to read with a 1kHz sample rate. EMGs can be measured with easy-to-apply surface electrodes which are clean and pose no health or safety risk to the user. Consequently, EEGs and GSR were ruled out as potential input devices in favour of EMGs, which were easier to read, control, and measure.

3.3.2 Which Muscles to Use?

Before continuing, readers may find it useful to review some basic anatomical terminology. The following terms will be frequently used in the following sections. The *sagittal*, or *median* plane of the body is the plane of symmetry, i.e. between left and right halves of the body. *Medial* is the direction toward the median plane. The little finger is medial to the thumb (with the palm facing forwards). *Lateral* is the opposite direction, outward from the median plane. The *anterior* or *ventral* side of the forearm is the front, when the palm is facing forward. The *posterior* or *dorsal* side is the back of the arm. *Distal* is the direction away from the root of a limb. The hand is distal to the elbow. *Proximal* is toward the root of a limb, the opposite of distal. A full description of all the physiological terms used in this thesis is available in Appendix A.

Selecting a Muscle Group

There are many muscle groups which can be used to control a computer pointer, but some groups are easier and more appropriate to use than others. There are four main factors to keep in mind when selecting a muscle group. First, the location of the electrodes should be in an area to which they are easy for the user to attach. One would prefer not to need a mirror or another person's assistance for attaching the electrodes. Another factor is that they should be in a socially convenient place. The head and the arm remain as the most workable zones, as they are usually left bare, or with a minimum of covering. A third factor is the size of the muscles and their EMG signals. The neck and face muscles tend to be smaller in size. This requires smaller electrodes to prevent the electrodes from physically interfering with motion. The arm muscles are longer, flatter, and have larger EMGs. This makes them both easier to measure and easier to attach electrodes to.

The fourth and last factor is the type of motion the muscle controls. The arm has joints in the shoulder, elbow, wrist and fingers. All have EMGs large enough to measure. However, the types of motions controlled are quite different. Collecting EMGs for some of the finger motions require electrodes in rather inconvenient locations, making the fingers a less than ideal control group. The shoulder has three degrees of freedom: adduction/abduction, rotation in the sagittal plane, and rotation about the axis of the arm. The elbow has one degree of freedom, making it useless as a two dimensional pointer (Figure 3.6). The wrist has the three degrees of freedom (Figure 3.7): up and down (extension/flexion), left and right (adduction/abduction), and clockwise and anti-clockwise rotation (supination/pronation). Only two degrees of freedom are necessary for a two-dimensional pointer, but having more degrees than necessary is an added bonus which will be addressed in more detail in Section 6.2.3.

Another advantage the wrist has over the other arm muscles is that the relaxed state is in the middle of each degree of freedom. That is to say, the wrist can move from a relaxed state to the left, right, up, or down, or rotate clockwise or anti-clockwise. Few other joints can do this. For example, the elbow, in a relaxed state is straight, when used, it can only move in one direction. This makes it more difficult to create a metaphor to control the pointer. The easier the metaphor, the easier it is to visualise the motion, and, consequently, to learn it. Following this logic, the best results are most likely to come from the linear wrist motions (extension/flexion and adduction/abduction). It is easy to visualise the relationship

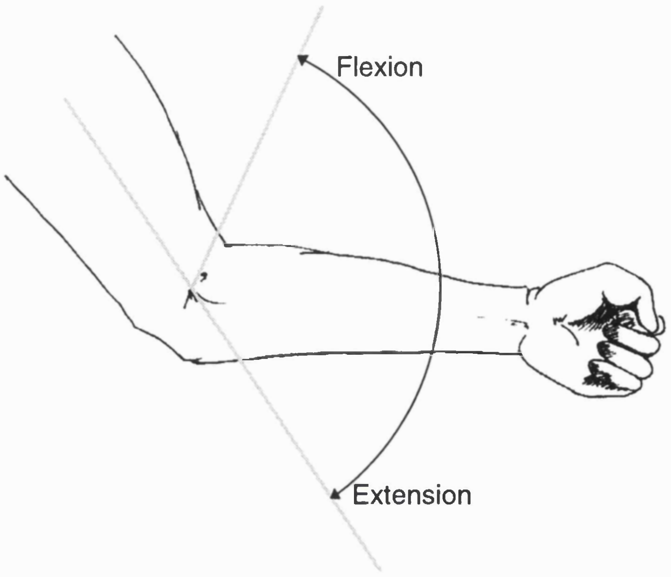


Figure 3.6: The one degree of freedom of the elbow

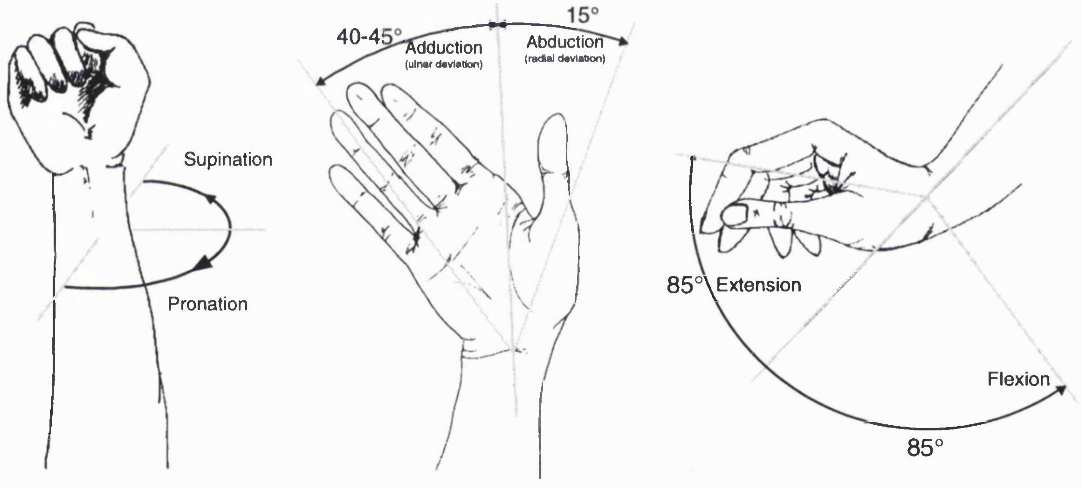


Figure 3.7: The three degrees of freedom of the wrist

between extending the wrist and moving a pointer upwards. Rotations have a less obvious relationship, which would impede learning.

An added bonus of the using the wrist is that the forearm has the least microorganism population density of any part of the body (Geddes, 1972). This means that it is safer to keep electrodes on for longer periods of time, which is important when considering the extended periods for which the device is likely to be used as a computer input device.

A final argument for using the wrist is its relatively high bandwidth in Fitts' Law tasks. The index finger, used by itself, has an IP of 3.0 bits/s. The wrist and forearm have a roughly equal IP of 4.1 bits/s. Slightly higher in bandwidth is the combination of thumb and index finger, with an IP of 4.5 bits/s (Balakrishnan & MacKenzie, 1997).

For these reasons the wrist was chosen as the muscle group to control the pointer for all the experiments (further information about using other muscle groups to control the pointer can be found in Section 6.2.2)

Selecting the Appropriate Wrist Muscles

There are six separate wrist motions: *supination* - rotating the palm to face forward, *pronation* - rotating the palm to face backwards, *flexion* - moving the wrist downward, *extension* - moving the wrist upward, *adduction* or *ulnar deviation* - sideways motion in the direction of the little finger, and *abduction* or *radial deviation* - sideways motion in the direction of the thumb. These motions are controlled by 19 muscles (Figure 3.8). The motions and their controlling muscles are detailed in Table 3.8.

Of these 19 muscles, 14 control the linear motions. Of those, only 6 are superficial and not obscured, and consequently easily measurable. Of the remaining six, the following were chosen: flexor carpi ulnaris (FCU), flexor carpi radialis (FCR), extensor carpi ulnaris (ECU), and extensor digitorum (ED). The extensor digitorum was chosen instead of the extensor carpi radialis longus and brevis pair because it is more superficial and has a stronger EMG. Table 3.9 summarises the motions which these four muscles control.

This choice of muscles gives an orthogonal group which can be used to determine the motion. There are four sensors. One sensor on the FCR is activated upon flexion or radial deviation. Another sensor on the ECU is activated upon extension or ulnar deviation. So far, this gives a one dimensional pointer which can go up or down. A third sensor on the FCU is senses either flexion or ulnar deviation. If the FCR and FCU sensors both detect a signal, the wrist is flexing, i.e. pointing down. If the ECU and FCU are both active, the wrist is moving in the ulnar direction, i.e. to the right (for the right hand). The FCR activated by itself yields radial deviation, i.e. leftward motion. ECU by itself means the wrist is extending, i.e. moving up. These three sensors by themselves are theoretically enough to uniquely determine the wrist's orientation. A fourth sensor was used to improve the accuracy. This sensor was placed on the ED, which shows extension, or upward movement of the wrist.

3.3.3 Hardware

Attaching the Electrodes

The placement of the electrodes is very important, since the Biofeedback Pointer works better with stronger signals. The electrodes need to be placed on the largest part of the muscle in order to get the clearest signal. In this section we describe the recommended positions for the electrodes and how to find them. The level of detail in this section tends to give the impression that the task of applying electrodes is somewhat daunting. In practice the task is simpler than it sounds, and can be performed rather quickly after a little practice. Precise details are given in this section to maximise consistency and repeatability for the tests in Chapter 5 and to avoid any confusion by the reader.

It helps to stretch the skin taut before attaching on the electrode. This is to prevent the skin from stretching later and pulling the electrode loose. This often occurs when the electrode is placed when the elbow is bent. When the elbow is straightened, the skin stretches and the electrode may come off. All the

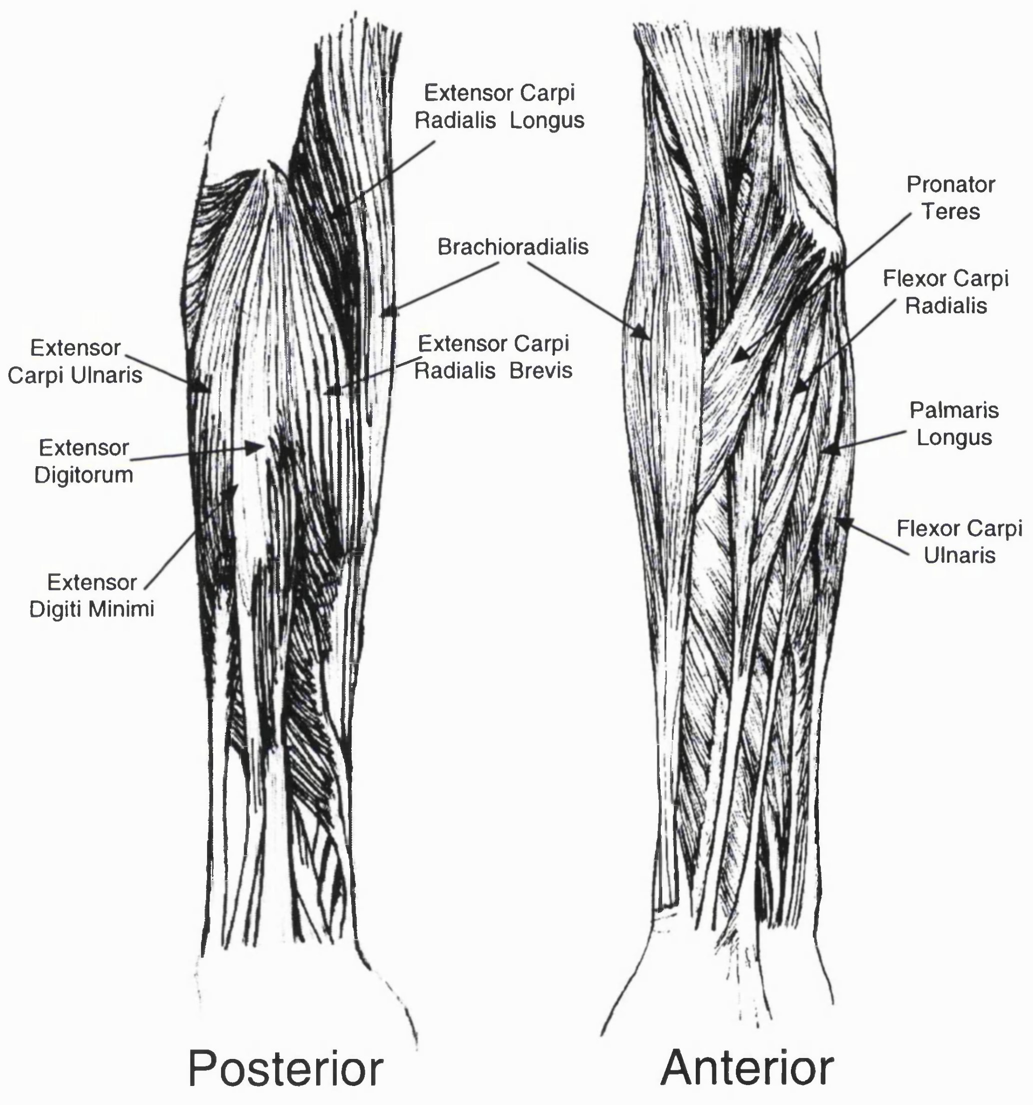


Figure 3.8: The superficial muscles in the lower right arm which control wrist motion. These are based on diagrams found in Moore (1985)

Table 3.8: The muscles controlling movement of the wrist. Since muscles are listed by the motions they perform, some muscles are listed more than once. The primary actions controlled by the muscle are in **boldface**. Motions are for the wrist unless otherwise specified. This is summarised from the muscle descriptions in Chapter 4 of Palastanga et al. (1990).

Motion	Muscle	Movements controlled	Level	Notes
Pronation	Pronator Teres	**pronation**, weak elbow flexion	superficia	
	Pronator Quadratus	**pronation**	deep	
	Brachioradialis	**elbow flexion**, pronation, supination	superficia	Used to return from extreme pronation or supination
Supination	Supinator	**supination**	deep	
	Biceps Brachii	**elbow flexion**, shoulder flexion, supination	superficia	With supinator for strong twisting/pulling motions, e.g. using a corkscrew.
	Brachioradialis	**elbow flexion**, supination, pronation	superficia	Used to return from extreme pronation or supination
Flexion	Flexor Carpi Ulnaris	**flexion, adduction**, finger extension	superficia	with PL and FCR
	Flexor Carpi Radialis	**flexion, abduction**, pronation, elbow flexion	superficia	with PL and FCU
	Palmaris Longus	slight metacarpophalangeal flexion, flexion	partly-obscured	weak muscle, absent in 10% of people
	Flexor Digitorum Superficialis	**metacarpophalangeal flexion, proximal interphalangeal flexion**, flexion	partly-obscured	
	Flexor Digitorum Profundus	**distal interphalangeal flexion**, proximal interphalangeal flexion, metacarpophalangeal flexion, flexion	deep	
	Flexor Pollicis Longus	**thumb interphalangeal flexion**, thumb metacarpophalangeal flexion, flexion	partly-obscured	
Extension	Extensor Carpi Radialis Longus	**extension, abduction**, elbow flexion	partly-obscured	with ECRB and ECU
	Extensor Carpi Radialis Brevis	**extension, abduction**	superficia	with ECRL and ECU
	Extensor Carpi Ulnaris	**extension, adduction**	superficia	with ECRL and ECRB
	Extensor Digitorum	**metacarpophalangeal extension**, interphalangeal extension, extension	superficia	
	Extensor Indicis	**assists ED**, extension	deep	
	Extensor Digiti Minimi	**little finger metacarpophalangeal extension**, little finger abduction, extension	partly-obscured	
	Extensor Pollicis Longus	**thumb extension**, abduction, extension	deep	
	Extensor Pollicis Brevis	**thumb metacarpophalangeal, carpometacarpal extension**, abduction, extension	deep	
Adduction (ulnar deviation)	Flexor Carpi Ulnaris	**Adduction, flexion**, finger extension	superficia	with ECU
	Extensor Carpi Ulnaris	**Adduction, extension**	superficia	with FCU
Abduction (radial deviation)	Flexor Carpi Radialis	**Abduction, flexion**, pronation, elbow flexion	superficia	with ECRL and ECRB
	Extensor Carpi Radialis Longus	**Abduction, extension**, elbow flexion	partly-obscured	with ECRB and FCR
	Extensor Carpi Radialis Brevis	**abduction, extension**	superficia	with ECRL and FCR
	Extensor Pollicis Brevis	**thumb metacarpophalangeal, carpometacarpal extension**, extension, abduction	deep	
	Extensor Pollicis Longus	**thumb extension**, abduction, extension	deep	

Table 3.9: The muscles used by the Biofeedback Pointer and the motions they control

	ED	ECU	FCU	FCR
Radial deviation				■
Flexion			■	■
Ulnar deviation		■	■	
Extension	■	■		

electrode pairs should be placed on along the arms axis, with the end with the lead facing proximally and the far end facing distally.

The electrodes used are 3M Red Dot^{TM} pediatric monitoring electrodes. Any electrodes will work, but these were chosen because they are small, cheap, and easy to order and apply. The electrodes are slightly too big for the amplifiers. This necessitates cutting the last half-centimetre off one side of the electrode. This can be easily done with scissors or a sharp razor. Since only some of the sticky tape is cut off, it has no effect on the quality of the electrode. The reference electrode is on its own and should not be cut.

It is often helpful to write the number of the amplifier on one of its electrodes. This greatly facilitates matching up the electrodes to the amplifiers if the amplifiers are removed and later re-attached. It is also a useful tool if attempting to diagnose any problems with the hardware. Since each of the amplifiers is numbered on the underside, the number cannot be seen when the amplifier is worn. Numbering the electrode makes the identity of each amplifier clear at a glance.

Before placing the electrodes, the hardware should be plugged into the computer and turned on. The software should be running, and EMG measurements turned on (see Appendix H). This will display the EMG from each electrode on the screen. This is useful when placing the electrodes, as the display will immediately show how strong the signal is.

Acromium Process (Reference Electrode) The easiest electrode to place is the reference electrode. This can be placed on any bony part of the body. The reason for this is that the skeleton acts as a common ground. Since there is no fat or muscle underneath the reference electrode the voltage will be constant. The acromium process in the right shoulder is the most convenient location for this electrode. The acromium process can be easily found as it is the larger and more medial of the two bony projections on the shoulder. The other bony projection, the coracoid process can also be used for the reference electrode. The acromium process was arbitrarily chosen for the sake of consistency.

Flexor Carpi Radialis The first electrode pair is placed over the Flexor Carpi Radialis. Hold out the right arm, with the anterior side facing up. At the wrist the FCR is the lateral-most tendon, just medial to the position where the pulse can be read. Follow this muscle proximally and medially along the arm to about an inch or two distal to the elbow. The muscle is easier to detect if the wrist is flexed. The electrode should be placed here, on the widest part of the muscle. This position should be easy to find by flexing and relaxing the wrist.

Extensor Digitorum The second electrode pair is placed over the Extensor Digitorum. Reach around to the dorsal side of the arm while keeping the anterior side up. Place your fingers about halfway down the arm and then extend and relax the wrist. You should be able to feel the muscle moving. When you can feel the muscle, place the electrodes there.

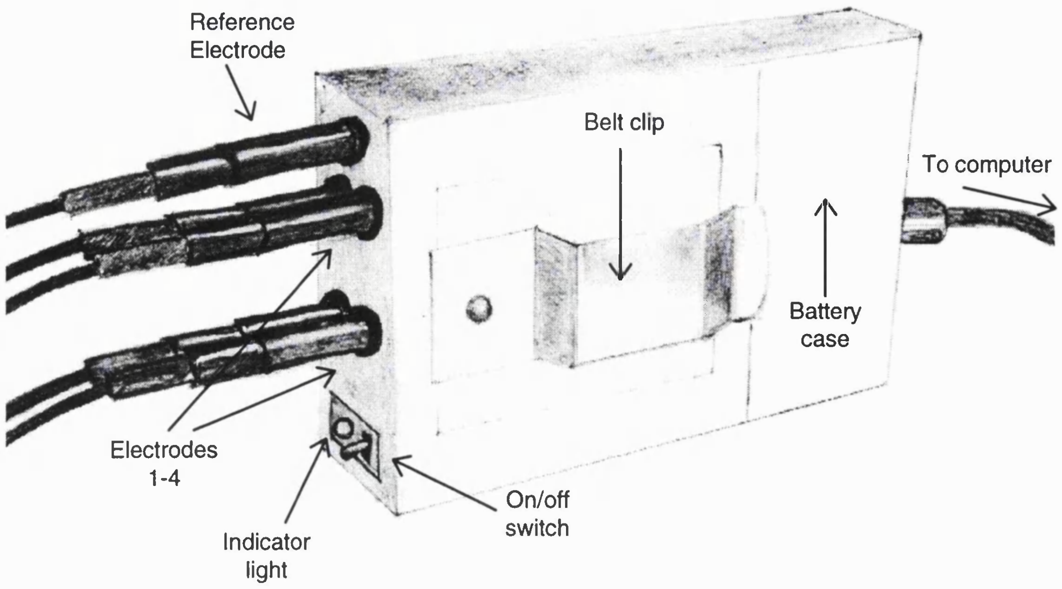
Flexor Carpi Ulnaris The third electrode pair is placed over the Flexor Carpi Ulnaris. Keep the arm anterior side up. Feel on the left side of the arm about an inch or two distal to the elbow. Adduct and relax the arm. Place the electrodes at the point where you feel the strongest muscle movements.

Extensor Carpi Ulnaris The fourth and last electrode pair is placed over the Extensor Carpi Ulnaris. Turn the arm dorsal side up. Feel approximately midway down the forearm, slightly left of centre. Abduct and relax the wrist. You should feel the muscle contracting under the skin. Place the electrode at the widest part of the muscle.

Data Collection

The majority of the Biofeedback Pointer's hardware is contained within a small belt-mounted box (Figure 3.9). This contains the power supply and circuitry to isolate the bioamplifiers from the computer. The remainder of the hardware consists of the bioamplifiers and an analogue to digital converter (ADC). A schematic of the data collection hardware is shown in Figure 3.10. Each electrode pair is plugged directly into a Remote Physiological Pre-Amplifier. The direct connection minimises potential background noise which can be generated by long leads. The physiological amplifier is made up of five parts, all contained in a solid epoxy resin housing. First, the current from each electrode is limited to $500\mu A$. This is a safety precaution necessary in order to prevent a power surge from reaching the user (British Standards, 1993). The EMG signals are fed into the first amplifier with a gain of 100. Low frequencies of 16Hz or less

Figure 3.9: The main hardware of the Biofeedback Pointer



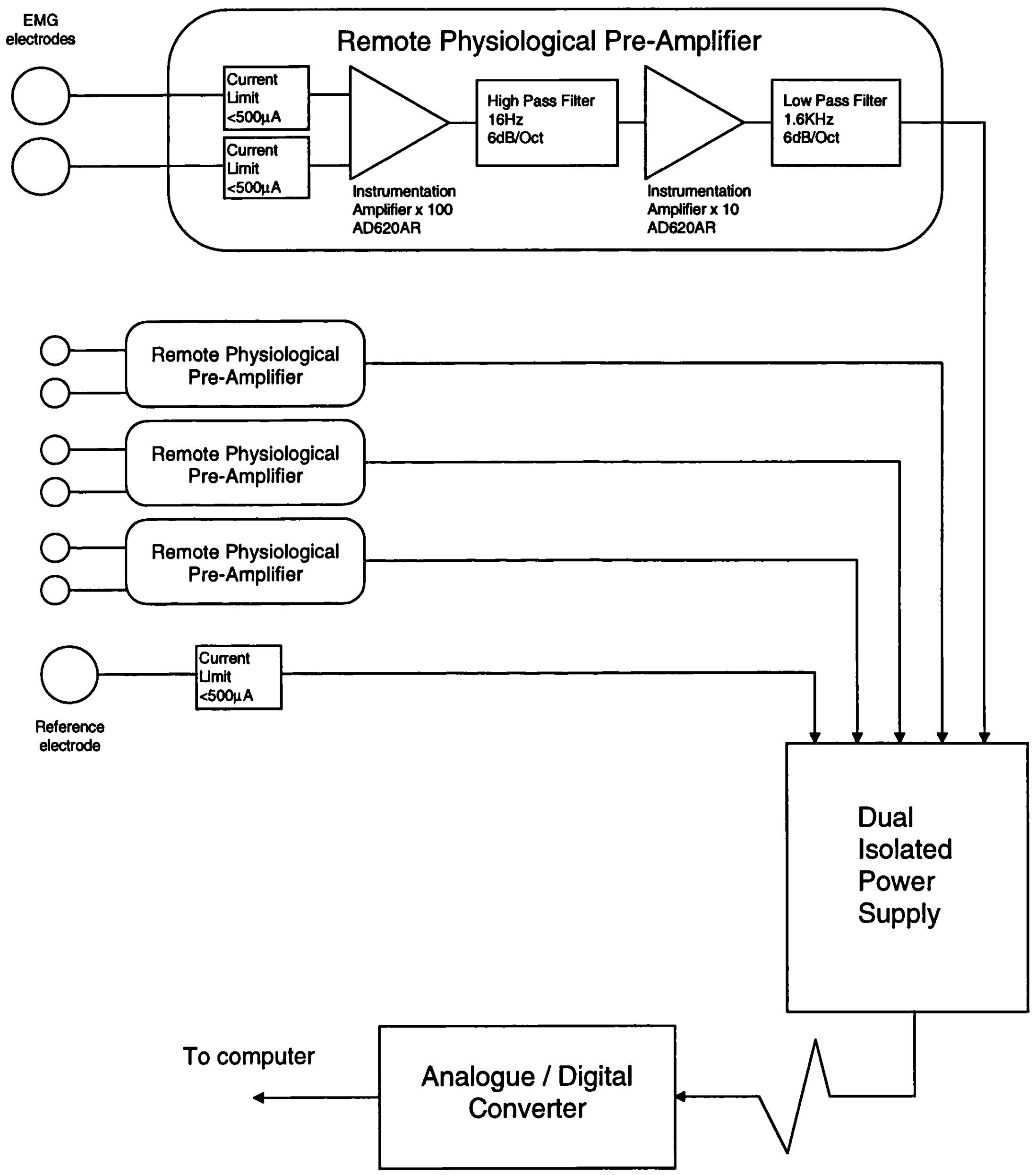

EMG electrodes
Remote Physiological Pre-Amplifier
Current Limit <500µA
Current Limit <500µA
High Pass Filter 16Hz 6dB/Oct
Low Pass Filter 1.6KHz 6dB/Oct
Instrumentation Amplifier x 100 AD620AR
Instrumentation Amplifier x 10 AD620AR
Remote Physiological Pre-Amplifier
Remote Physiological Pre-Amplifier
Remote Physiological Pre-Amplifier
Current Limit <500µA
Reference electrode
Dual Isolated Power Supply
To computer
Analogue / Digital Converter

Figure 3.10: Diagram of the bio-amplifier setup

are then filtered out. Next the signal is sent through the second amplifier with a gain of 10. Lastly, high frequencies over 1.6kHz are filtered out. The specifications for the physiological amplifier are detailed in Appendix G.

All four amplified EMGs and the reference electrode are sent into the box containing the dual isolated power supply (Figure 3.9). The reference electrode is also current limited to 500μA. A single 9V battery powers the amplifiers, plus an LED which blinks when the power is on. The four amplified signals emerge from the box in a long cable which ends in a male 25 pin D-shell connector.

The analogue to digital converter is a PICO ADC-11, which is a eleven channel ADC which is attached to the parallel port. The ADC is powered by the PC and its operation is controlled in software. It measures voltages in the range from 0 to 2.5V and can run at speeds up to 18kHz. The technical specification for the ADC-11 can be found in Appendix G.

Signal Processing

Signal processing can be done by several different methods. Three of those were described in Section 2.3.10. Of all the methods, the neural network system (Hiraiwa et al., 1993) seemed to have the most advantages. This method was adapted to work with the above hardware, using an increased update rate of 16Hz.

The process of data acquisition and analysis is pictured in Figure 3.11 and described as follows. Each of the digitised signals is read every millisecond (1). Every 64ms a Fast Fourier Transform (FFT) (2) is performed on the past 512 data readings. The real and imaginary parts are combined to give a 256 point array (3) which is the square of the absolute value of the frequency spectrum. The spectrum is then linearly rebinned to an 8 point array (4). Each of the four reduced spectra (5) are then fed into the neural network (6), which converts the data into pointer motion (7).

3.3.4 Pattern Recognition

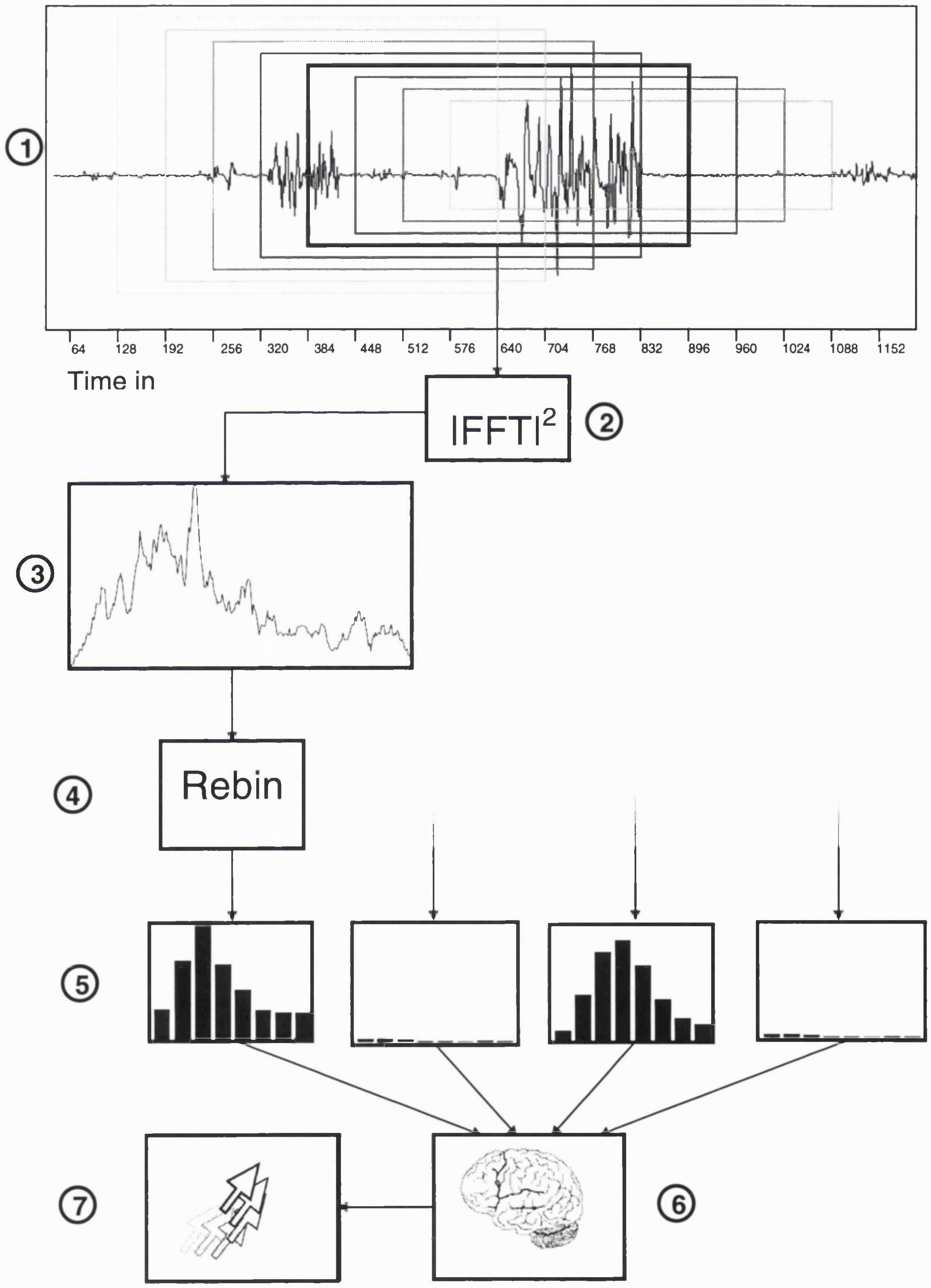
Accurate recognition of the complicated EMG waveforms is essential to the performance of the Biofeedback Pointer. In Section 2.3.10 we described three methods for EMG analysis. Of these three, only the neural network provides the continuous values and a fast enough update rate to be used as a pointing device. For this reason a modified version of Hiraiwa et al.'s (1993) neural network was used to analyse the EMG waveforms.

The Neural Network

Hiraiwa et al.'s (1993) neural network analysed two EMG inputs to provide 10 finger joint angles as outputs. The Biofeedback Pointer has four inputs and x and y coordinates as outputs. By exploiting biofeedback, the Biofeedback Pointer's neural network did not need to perform all the work, allowing for a somewhat larger margin of error than Hiraiwa et al.'s (1993). Consequently, a much simpler neural network could be used for the Biofeedback Pointer. A simple perceptron was found to give the most accurate results.

A *simple perceptron* is a one layer feed-forward neural network (Hertz et al., 1991). This type of neural network takes an array of N input values and produces an array of M output values. One way to

Figure 3.11: The signal processing



implement this network is to multiply the input array by an $N \times M$ matrix of weights. Sometimes the input array is prepended with an additional unit element to accommodate for a uniform offset. This is to allow the output to be non-zero when all the inputs are zero.

The Biofeedback Pointer uses four 8 point arrays as inputs. Combining these arrays with an additional point to act as an offset gives a 33 point input array (a). Since the output is a two-dimensional vector, it is simpler to conceptualise if we use separate arrays for the weights, $\mathbf{w}_x$ and $\mathbf{w}_y$. The dot product of the input array by each of the weight arrays gives the output values of (x, y):

$$x = \mathbf{w}_x \cdot \mathbf{a} \tag{3.1}$$

$$y = \mathbf{w}_y \cdot \mathbf{a} \tag{3.2}$$

Figure 3.12 shows a diagram of this neural network.

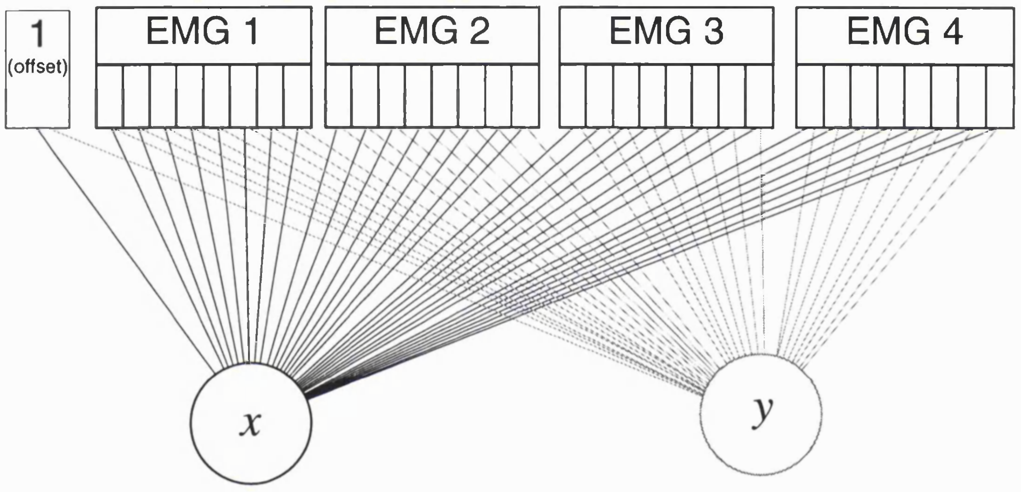


Figure 3.12: The Biofeedback Pointer's simple perceptron neural network

Training

When the pointer is first used, the neural network must be trained. This is done by having the computer move the pointer around the screen while the user follows the motion with their hand. The pointer starts at the centre of the screen and pauses. Each motion begins from the centre, moves out, pauses, then moves back to the centre, where it pauses before beginning the next motion. The motions are: diagonally down and right, diagonally down and left, diagonally up and left, diagonally up and right, down, left, up, and finally right (Figure 3.13). After all the motions, the pointer pauses at the centre of the screen while the weights are calculated.

At each time step the pointer position (x_i, y_i) and the reduced EMG spectra $(\mathbf{a}_i)$ are saved. When the training is complete, we have $N = 378$ sample points saved in the vectors $\mathbf{x}$ and $\mathbf{y}$, and the matrix a, generated by N arrays of EMG spectra, each with 33 values. The weights are then calculated by the following equations:

$$\mathbf{w}_x = \mathsf{R}^{-1}\mathbf{q}_x \tag{3.3}$$

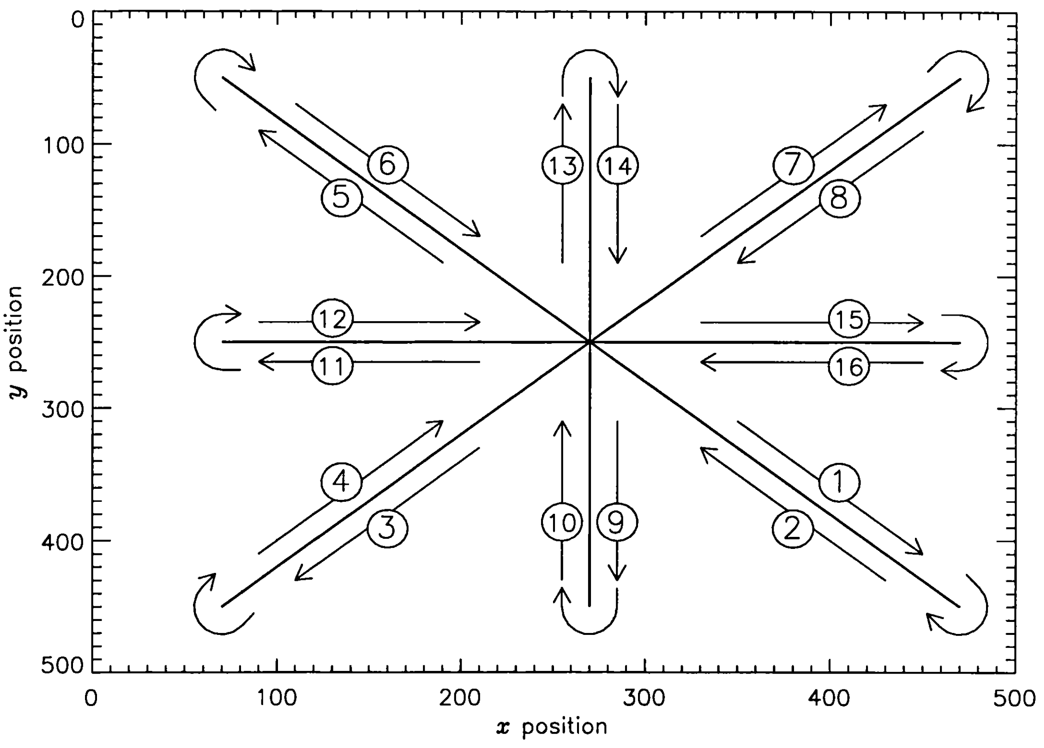


Figure 3.13: The pointer training motion

$$\mathbf{w}_y = \mathsf{R}^{-1}\mathbf{q}_y \tag{3.4}$$

where the variables R, $\mathbf{q}_x$, and $\mathbf{q}_y$ are defined as

$$\mathsf{R} = \mathsf{a}\mathsf{a}^{\mathsf{T}} \tag{3.5}$$

$$\mathbf{q}_x = \mathsf{x}\mathsf{a}^{\mathsf{T}} \tag{3.6}$$

$$\mathbf{q}_y = \mathsf{y}\mathsf{a}^{\mathsf{T}} \tag{3.7}$$

The derivation of these equations can be found in Appendix F.

It is possible that when the pointer is trained, the user's motions will lag behind the motion on the screen. To compensate for this problem with reaction time delay, the weights are calculated eight times, with time offsets from zero to seven steps. This can compensate for reaction time delays up to $^7/_{16}$ seconds. The weights which yield the least error are saved and used in the neural network. The error in this network is defined as the average value of the distance from the real points (x, y) to the calculated points $(\mathsf{x}', \mathsf{y}')$. In other words:

$$\text{error} = \sum_{i=1}^{N} \frac{\sqrt{(\mathbf{w}_x^t \mathbf{a}_i - x_i)^2 + (\mathbf{w}_y^t \mathbf{a}_i - y_i)^2}}{N} \tag{3.8}$$

Using

Once the neural network weights are calculated, it immediately starts translating the EMG input into a two-dimensional position vector ($\mathbf{r}$). The earliest versions of the Biofeedback Pointer used this vector as the position of the pointer on the screen. This was very noisy and hard to control. The biggest deficiency was that the centre of the screen was easy to point to, and less central points were difficult to target. To solve this problem it was decided that the wrist motion would control the velocity of the pointer. The velocity ($\mathbf{v}$) was calculated from the position vector ($\mathbf{r}$) by the piecewise function:

$$\mathbf{v} = \begin{cases} \frac{\mathbf{r}}{s} & r < cs \\ \frac{\mathbf{r}}{s}(3 - \frac{2c}{r}) & r \geq cs \end{cases} \tag{3.9}$$

This continuous function scales the larger motions by a factor of three. In addition, this is controlled by two variables, c and s. s is the overall scaling, which converts the (x, y) position into a reasonable value for velocity. Since r ranged up to 300, scaling it down by a factor of $s = 16$ provided the reasonable range of values. c is the cutoff magnitude. Below this value, the velocity is unchanged, but above it is scaled by a factor of 3. A value of $c = 6$ was settled on by a process of trial-and-error. Below this value the effect of scaling was unnoticeable, and larger values of c required excessively large motions to trigger the scaling.

Using these values allowed the pointer to perform detailed small movements and faster long movements. This type of scaling is common with continuous graphic input devices and is called an *adaptive control-to-display (C/D) ratio* (Foley et al., 1990).

Other Neural Networks

The simple perceptron was the most effective neural network for the Biofeedback Pointer. In addition to the simple perceptron mentioned above, a Back-propagation neural network (BPN) was also tried with a variety of nodes and levels. Both networks took in 32 inputs (4 spectra of 8 points each) and produced 2 outputs (x and y). The actual process of calculating the weights and outputs varied greatly.

A test was designed to compare the networks. In this, a large number of sets of training data was collected. Part of this data was used to train each of the neural networks. The rest of the data was used to determine the average error which was measured as the distance from the actual to the predicted point. All the possible parameters of each network were varied to determine the best values.

The Back-propagation Network The Multi-level BPN was based on a combination of the algorithms found in Freeman & Skapura (1991) and Hertz et al. (1991). In training the BPN, all the weights were set to random values between 0 and 1. For each step in the trial the outputs are calculated and weights adjusted. This is repeated until the weights converge to constant values. The forward and back propagation process is as follows:

1. Propagate the inputs through the network. The sum of all the weighted inputs to the node (h) is scaled by the function:

$$g(h) = \frac{1}{1 + e^{-h}} \tag{3.10}$$

   which is the output of the node.

2. Once the final outputs are found, the error (δ) is calculated by the function:

$$\delta_i = g'(h_i)(t_i - O_i) \tag{3.11}$$
$$= O_i(1 - O_i)(t_i - O_i) \tag{3.12}$$

where t_i is the target value and O_i is the output value.

3. These errors are propagated backwards by:

$$\delta_{i,k-1} = g'(h_{i,k-1}) \sum_j w_{j,i,k}\delta_{j,k} \tag{3.13}$$

This is done for each node in each layer.

4. The weights are then updated by:

$$\Delta w_{j,i,k}(t) = \eta\delta_{i,k}O_{j,k-1} + \alpha\Delta w_{j,i,k}(t-1) \tag{3.14}$$

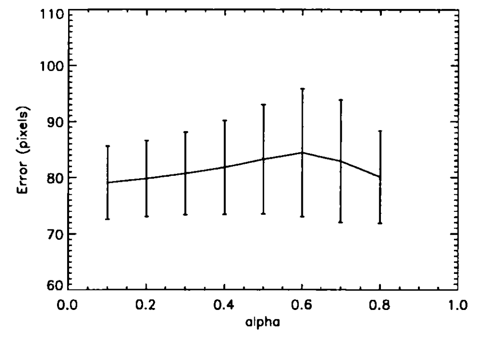
where α is the momentum term and η is the learning rate.

5. This is then repeated until the weights converge.

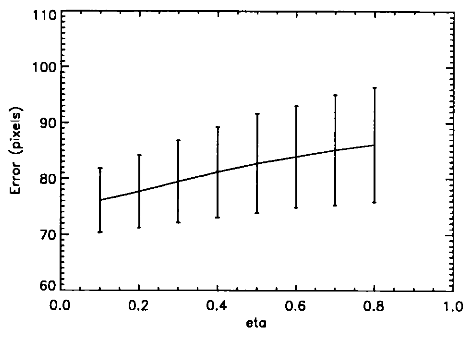
The BPN had three levels. One input layer of 32 nodes, one output layer of 2 nodes and one hidden layer of n nodes. This gave three parameters: α (momentum), η (learning rate), and n (nodes). In the test, α and η were varied from 0.1 to 0.8. The number of nodes (n) ranged from 2 to 40.

Figure 3.14 shows the effects that varying these parameters has on the errors. Smaller values η give less error. Very large or very small values of α give better results. The best number of nodes is 9. The smallest values of α and η with 9 nodes yields an error of 76 ± 23. This error is the distance in pixels from the computed point to desired one. This was calculated using 13 sets of training data. Other BPN setups were tried, but none gave as low errors. The lowest errors on the other BPNs ranged from 90 to 150.

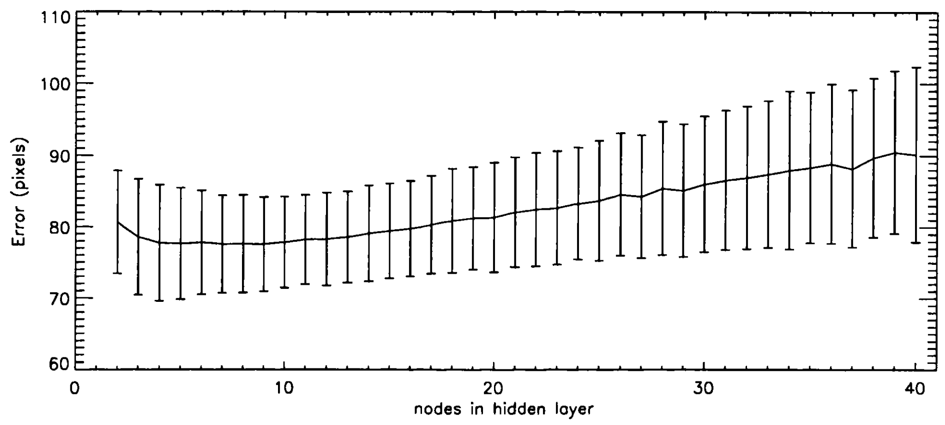
The Simple Perceptron Network The Simple Perceptron neural network collected all the training data before calculating the weights. The best values for the weights were calculated by a multidimensional least squares fit, as described in Appendix F. This did not require any initial guess because an exact solution was found by solving the matrix equations, instead of using an iterative process to estimate the weights. There were no parameters which could be changed for optimisation, making it easier to determine whether this method was effective or not. The average error after one set of training data was 82 ± 17, which is slightly higher than the best BPN, but the difference is not significant on the 5% level. On the other hand, this method achieved this result 13 times faster and with an order of magnitude fewer calculations per step, making it clear that this was the better method.



(a) Variation in α



(b) Variation in η



(c) Variation the number of nodes

Figure 3.14: Error levels of the Backpropagation Neural Network. Errors are given in pixel units *before* scaling (Equation 3.9)

3.3.5 Biofeedback

Once the neural network is trained to recognise the arm motions, its learning process is over. From then on the user's performance with the pointer is improved by biofeedback. *Biofeedback* is a process in which one or more biological signals are measured and converted into a form which can be displayed to the person (visual, audio, etc). Through a process of trial and error, the person soon figures out how to control the biological function.

This process occurs in the Biofeedback Pointer. As users manipulate the pointer, they slowly learn exactly which arm motions correspond to the pointer motions. The muscles used are not exclusive to the trained motions. For example, the extensor digitorum is not only used in extending the wrist, but is also used in extending the fingers. As a consequence, moving one or more fingers can give similar signals to moving the whole wrist. It has been tested empirically that by experimenting with different motions, the user can find that they achieve the same motion by a slight movement of the middle finger that they would otherwise have used their entire hand. This provides a process in which the user progressively learn the easiest method for interacting with the computer through the Biofeedback Pointer.

3.3.6 Uses for the Biofeedback Pointer

In operation, the Biofeedback Pointer is essentially equivalent to a joystick: the forearm is the equivalent to the base of the joystick and the hand is equivalent to the joysticks shaft. As a consequence we would expect that the Biofeedback Pointer would be best suited for similar applications. As stated in Section 2.3.3, the joystick works best for navigation and low precision pointing. It would stand to reason that the Biofeedback Pointer also be most efficient in those situations.

Navigation refers to both two and three dimensional graphic navigation. This includes such applications as video games, exploration of 3D data, operating a virtual trackball, etc. This also includes navigation in a windows-style environment, such as menu selection and scrolling. Low precision pointing consists mostly of selecting objects (point and click), or moving objects (click and drag). The Biofeedback Pointer would be ineffectual at higher precision tasks such as drawing or fine manipulation tasks.

One disadvantage of the Biofeedback Pointer is that it will continuously move, even when not being used. The only way to keep the pointer still is to completely relax the arm, preventing the user from doing anything else with that arm. This is not a problem for most tasks, since the window manager usually does not care where the pointer is when it is not performing any selection. A situation where this is a problem is an environment with pointer-driven keyboard focus. In this environment, all text input is sent to the window which contains the pointer. It would be very difficult to both control the position of the pointer and enter text at the same time. There is a way around this, and that is to be able to turn off the pointer when it is not needed. This issue is discussed in more detail in Section 6.2.3.

3.4 Applications

While the Biofeedback Pointer and Chording Glove can be used individually or with other input devices, the full advantage comes from using the two devices together as an input for a wearable computer. In this system, the Chording Glove is worn on the dominant hand, while the Biofeedback Pointer's bio-amplifiers are attached to the forearm. These are connected to a wearable computer with a heads-up display. In this system, there is no externally visible difference between the active and passive states of computer use. The only difference would be that in the active state the user would see that the display is on. Switching between the active and passive states can be accomplished simply by pressing a button or two. Consequently, this system rates rather well when judged by the four portability factors introduced in Section 2.4.

Using bioelectric-based graphic input devices for people with severe motor disfunction has been accomplished with EMG, EEG, and EOG. Chord keyboards have been shown to be more usable than a standard keyboard for people with motor disabilities like cerebral palsy and, muscular dystrophy and dyslexia (Kirschenbaum et al., 1986). The reason for this is that the minimal motions involved in chording are much easier to perform for people with these kinds of disabilities. The system introduced here can be used, in one form or another, for people of varying levels of disability. Those with minor motor disabilities could use the Biofeedback Pointer and Chording Glove together to interact with a computer in the same manner as an able-bodied person. For those with more severe disabilities who cannot use the Chording

Glove, the Biofeedback Pointer could be used both as a normal pointer, and to select characters from a soft keyboard.

The portable nature of the Chording Glove and Biofeedback Pointer allows them to be used in several kinds of hostile environments. In Space there is very little room for extraneous equipment. This system takes up very little room and can even be integrated into a space suit. Zero-gravity has its own special problems. A standard keyboard cannot be used because of Newton's Law of action and reaction: by typing a keyboard one could slowly launch oneself across the room (Matias et al., 1996). If the Chording Glove is operated by using the body as the chording surface, this problem is avoided completely. Our system also has benefits for military usage. The devices are silent and are very fast to put away, an important factor in potentially life-threatening situations. In underwater situations, where voice input is near-impossible, our system provides a method for text and graphic interaction with minimal motions.

A more everyday application would be text or graphic interaction for fieldwork. An excellent application would be archaeology. A map can be scrolled or zoomed to determine the present location. The location of artefacts can be recorded, along with notes about them. The computer can then be used to look up further information about the artefacts. With this system, all this can be performed in the field, without even needing to look away from the task at hand.

3.5 Summary

The Chording Glove is a text input device designed for efficient use in a mobile environment. Chords are input by pressure sensitive switches on each finger tip. The shift modes are controlled by small sticky-shift buttons located on the side of the index finger in reach of the thumb. Useful, but infrequently used function keys are located on the back of the hand and can be pressed by the opposite hand.

The chord keymap is designed to take full advantage of the glove's intended use by making the most common characters require the least work to create. The keymap is also designed to maximise speed, minimise errors, and be easy to learn. If necessary, the keymap can be easily changed, to allow more efficient text entry in other languages or for specialised applications.

The lack of visual supervision and the requirement of only one hand for text entry are major factors in allowing the Chording Glove to work in a mobile environment. The lower input speed, as compared to a standard keyboard, suggest the Chording Glove should be used only for causal text entry. Fortunately one would not expect to perform more intensive text entry in a mobile situation. In addition to being a design requirement for a mobile environment, the variety of workable hand orientations for text entry, has the potential to reduce some of the factors that have been linked to RSI on a standard keyboard.

The Biofeedback Pointer is designed to control a computer pointer in a mobile environment. Nothing need be held or manipulated, wrist motion alone is used to control a 2D pointer. The wrist was chosen primarily for the ease of applying the electrodes and measuring the EMGs. Added incentives were high IP of wrist motion and the fact that electrodes could be safely kept on the forearm for relatively long periods of time.

The EMGs from four of the major muscles in the forearm are measured by bio-amplifiers, converted to a digital signal, and transformed into power spectra before being analysed by a neural network. These four spectra are then translated into an x and y velocity for the pointer. The neural network is trained by requiring the user to follow the cursor as it moves trough a series of motions. When training, the EMG spectra and the cursor position are fed into the neural network which finds the best set of weights to derive the cursor position from the EMG spectra.

The Biofeedback Pointer's design makes it best suited for tasks requiring navigation or low precision pointing. Examples of these tasks are basic windows-style GUI interaction and exploring graphical data, both of which are likely to be performed by a mobile computer. The continuous motion of the Biofeedback Pointer can cause problems in certain environments or applications. Consequently there must be a method for turning the pointer off when motion is unwanted.

Both the Chording Glove and the Biofeedback Pointer used theoretical tests during their construction to determine if certain aspects of the devices would work. These experiments were very limited in the scope of what they could test. In order to fully understand the devices, they must be empirically tested in operation. Experiments were designed and carried out for the Chording Glove and the Biofeedback Pointer. These are the subjects of the next two chapters.

Chapter 4

The Chording Glove Experiment

4.1 Introduction

This chapter describes an experiment to examine the claims set out in the previous chapter. Specifically, the Chording Glove claims to combine the quick learning time and high performance of a chord keyboard with the portability of a glove interface.

The learning rates of standard chord keyboards (see Section 2.2.3) indicate that the keymap of a chord keyboard can be learned in a short period of time. Mounting the keys on a glove should make no difference in the learning rate. As a consequence, the Chording Glove's keymap should be learned and retained in a comparable manner. In other words, we hypothesise that the 97 character keymap for the Chording Glove should be memorised to the extent that the user can perform continuous text input within 90 minutes. In addition, the keymap should remain in long term memory, allowing users to maintain their chording speed after extended periods of disuse.

After eleven hours of use, a one-handed chord keyboard can be expected to have a text entry speed around 20–24wpm. Sources differ as to the expected text entry speed of a standard keyboard. Gopher & Raij (1988) found it to be about 14wpm after 11 hours of practice , while a touch typing course claims to reach a speed of 20wpm after 12 hours of training (Noyes, 1983). However, one thing is clear, for beginner and moderate users, a one-handed chord keyboard should be as fast or faster than a QWERTY keyboard. As a consequence, we hypothesise that the Chording Glove should be able to reach input speeds of at least 20wpm after 11 hours of chording.

The expected error rate for the Chording Glove was be determined in Chapter 3 by weighting Seibel's (1962) error data against the chord's frequency of use. The result is an error rate of approximately 7%. This is, of course, the steady state error rate, ie the rate after long term use. For comparison, the error rate for the QWERTY is 12.7% (Potosnak, 1988). We hypothesise that the error rate for the Chording Glove, after 11 hours of use, should be somewhat higher than the 7% level. In addition, we hypothesise that the Chording Glove should be able to chord on any solid surface without any significant loss in speed.

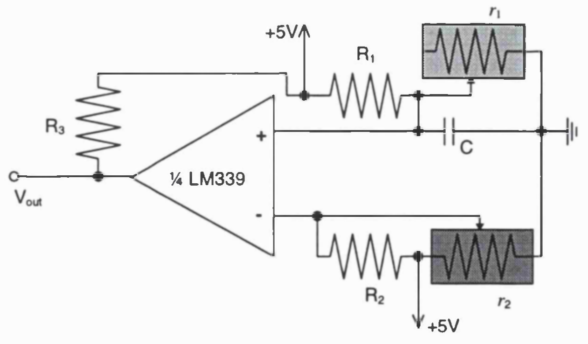
In order to test these hypotheses, a longitudinal experiment was performed. In addition, data were collected to analyse fatigue, muscle strain, preferences, and the subject's opinions on various aspects of the device in order to provide further insight into the Chording Glove's performance.

4.2 Materials

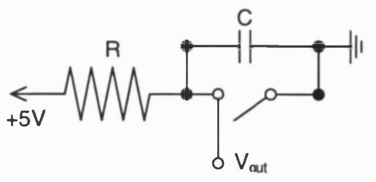
A simplified version of the Chording Glove was used to limit the scope of the experiment. It was decided that the features should be limited to only those necessary for plain, line-by-line text entry with no editing functions.

4.2.1 Shift and Function Key Changes

The `<Help>` key's behaviour was changed slightly for the experiment. The normal behaviour (as mentioned in Section 3.2.1) is to display the keymap when double pressed. The keymap would then disappear when `<Help>` is pressed again. In the experiment, the keymap is displayed while the `<Help>` key is depressed. When released, the keymap disappears. This is done to measure how long the subject needed to view the keymap.



(a) Resistive Foam



(b) Metal Plate

Figure 4.1: Circuit diagrams for the two types of finger sensors

The `<Control>` shift was not needed for the experiment, nor were any of the function keys except the `<Help>` key. To minimise the amount of controls the subjects had to contend with, the `<Control>` shift was remapped in software to produce the same results as pressing the `<Help>` key. Since the `<Help>` key was now redundant and the rest of the function keys were not wanted, the entire set of function keys were removed from the glove to avoid confusion.

The LEDs next to the `<Control>` shift were not removed, nor were they used during the experiment.

4.2.2 Finger Sensors

The first five subjects in the experiment started out using the Resistive Foam finger sensors(Figure 4.1(a)). As the foam was compressed, the resistance (r_1) lowered. This was measured as a change in voltage. Since this was an analogue sensor, the voltage had to be compared against a set cutoff voltage. This voltage was set by a potentiometer (r_2). Each sensor had it's own control potentiometer to allow it to be adjusted to maximum sensitivity.

Once the Chording Glove was used by multiple people over a short period of time, it was found that the sensors could not handle such frequent use. The problem was that r_1 was excessively temperature dependent. The longer the sensor was in use, the more the resistance changed. Eventually the resistance would fall outside the range of r_2, making the sensor useless after just over an hour of chording.

These sensors were replaced with a new design. This was a large, non-flexible button made of two metal plates separated by springs (Fig 4.1(b)). Compressing the button touched the plates, grounding V_{out}.

In order to change the sensors without redesigning the entire circuit board, the new finger sensors were attached to where the function keys were. The software was then modified to map the function key input to the finger sensors. This could be done since the function key input expected a digital signal, as opposed to the finger sensor inputs which were designed for an analogue one. This did not cause a problem since no function keys were needed.

The new sensor design was used for the remainder of the experiments, However, as a result, the subjects who used the foam sensors suffered a degradation in performance, as compared to those who used the plate sensors. The effects of this is discussed in more detail in Section 6.1.1

4.2.3 The Keymap

Table 4.1: The reduced keymap

chord	lower case	upper case	number and math	punctuation
● ● ○ ○○	a	A	5	&
● ● ○ ●○	b	B	*	*
● ● ● ●○	c	C	0	%
● ○ ● ●○	d	D	/	\
○ ○ ○ ●○	e	E	3	!
● ○ ○ ●●	f	F		
○ ○ ● ●○	g	G	)	>
○ ○ ● ○○	h	H	2	@
○ ○ ○ ○●	i	I	4	?
● ● ● ○●	j	J		
○ ● ● ●●	k	K	^	^
○ ● ● ●●	l	L	=	<
○ ● ● ●○	m	M	-	_
○ ● ● ○○	n	N	(	"
● ○ ○ ●○	o	O	7	|
○ ● ○ ●○	p	P	+	#
○ ● ● ○●	q	Q	?	?
● ● ● ○○	r	R	9	‘
● ○ ● ○○	s	S	6	$
○ ● ○ ○○	t	T	1	’
○ ● ○ ○●	u	U	[	{
○ ● ○ ●●	v	V		
● ● ○ ○●	w	W	]	}
● ● ○ ●●	x	X	*	*
● ○ ○ ○●	y	Y	8	~
● ○ ● ○●	z	Z		
● ○ ● ●●	,	,	,	;
○ ○ ● ●●	.	.	.	:
● ○ ○ ○○	space	space	space	space
○ ○ ○ ●●	back-space	back-space	back-space	back-space
● ● ● ●●	return	return	return	return

A slightly reduced keymap was used in the experiment (Table 4.1). The special characters <`Tab`> and <`Escape`> were not shown, since they were not used in the experiment. In addition AutoCaps was turned off in the experiments as it would only add another variable to an already complex situation.

4.3 Method

4.3.1 Subject Selection

Advertisements were placed around the Queen Mary and Westfield College campus. Additional advertisements were posted on the several appropriate local internet newsgroups. Potential subjects replied by email or telephone. They were asked to fill out an initial questionnaire either on paper or via email. Adequate subjects were accepted and times were scheduled. Details on the selection criteria are available in the appendix in Section D.1.1).

Fifteen subjects were used, but 5 dropped out after the tutorial. Of the remaining 10, there were 4 males and 6 females, all right-handed and aged between 18 and 28. All the subjects described themselves as competent typists with six subjects using keyboards for only a few hours a week, and the rest typing on a daily basis. No subjects reported ever having used a chord keyboard or having experienced any RSI. Each subject was paid £1.50 per session after the tutorial, and an additional £15.00 on completing all the experiments. One subject chose not to be paid. The subjects each performed ten chording sessions following a tutorial session, spread out over a period of approximately 2–3 weeks.

4.3.2 Tutorial

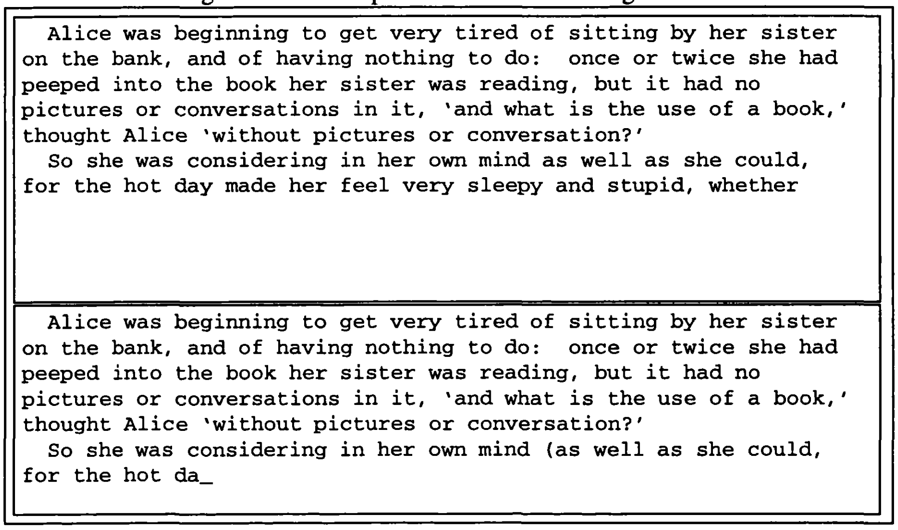
Figure 4.2: Sample text from the tutorial session. The entire tutorial is available in Appendix D

```
Type the letter g
gggg

Now type:
ing
ing

Note that 'ng' is formed by the first two fingers(index and middle)
followed by the second two fingers (middle and ring).

Type again: ing
ing

Now type:
thing
thing

this and that thing
this and that thing

let their chimney fly.
let their chi_
```

The initial session was a tutorial to teach the chord keymap. The tutorial was designed to last approximately one hour (Figure 4.2). The subject was given two sheets of paper for the tutorial. The first listed the keymap (Figure 4.1) and the second displayed the chords which are similar in shape to the characters they make (Figure 3.4). The subject learned each chord by being asked to chord a character and then generate several short phrases using it and some previously learned chords. The subjects could take as long as they needed to finish. Most took between 1 and $1^1/_2$ hours to complete. After finishing the tutorial, the subject was given a questionnaire to fill out. See Appendix D for more details on the tutorial and questionnaire.

4.3.3 Chording Sessions

Figure 4.3: A sample section of a chording session



```
  Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, 'and what is the use of a book,'
thought Alice 'without pictures or conversation?'
  So she was considering in her own mind as well as she could,
for the hot day made her feel very sleepy and stupid, whether
```

```
  Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, 'and what is the use of a book,'
thought Alice 'without pictures or conversation?'
  So she was considering in her own mind (as well as she could,
for the hot da_
```

There were ten chording sessions following the tutorial. These sessions consisted of preparation, fifty minutes of text input, and wrap-up.

Preparation

Before starting the subject is asked, *"Have you experienced any unusual or unexplained pain in your hand, arm, or back since the last session?"* The subject then puts on the glove and is asked if they are ready to begin.

Trials

The subject would then perform four chording trials. Each trial consisted of entering text which would appear on the monitor. The screen was divided into two windows. The top displayed the text, one line at a time. The subject would chord this text and their work would appear in the bottom window. When the subject pressed <Return> the next line would appear in the top window. In each session the subject was told to chord as quickly and as accurately as possible. Fixing errors was not too important, and the subjects were told to fix mistakes "only if it doesn't require too much effort".

The trial length and content were varied to avoid mental fatigue in the subject. The first and third trials were always 15 minutes long, while the second and fourth were always 10 minutes. Between each trial the subject could take a break of up to five minutes. Most subjects preferred to wait less than a minute before continuing with the trials.

The first and last trials always consisted of entering text from the novel *Alice's Adventures in Wonderland.* This was chosen because it is easy to read, but is still interesting enough to keep the subject's attention. The second and third trials alternated between text and data entry. If the first was text entry the second would be data entry, and vice versa. Each session alternated which was performed first. The data entry mostly consisted of tabular stock market data. The data was mostly numbers, but there were some words. The full text of the data entry trials is in Appendix D.

Every few sessions, during a randomly selected trial (any except the first one), the subject was asked to chord without a chair to sit in, or without a desk to chord on. This was to test the portability of the Chording Glove. Only four of the subjects were used in these tests.

Wrap-Up

After the final trial of the day, the subject was given a questionnaire. This was the same one that was used after the tutorial (see Appendix D). The subject then was asked to schedule more sessions if they had not already done so. Afterwards the subject was given £1.50 and were done for the day. If the subject finished all ten sessions they were given a £15 bonus.

4.4 Results

4.4.1 Text Entry Speed and Error Rate

At the end of the tutorial the average overall speed was 8.9 ± 1.4wpm. The speed was calculated as the average digram time in seconds (t_d), assuming an average of 5 characters per word (MacKenzie et al., 1997). The conversion equation is:

$$speed[\text{wpm}] = \frac{12[\frac{\text{wpm}}{\text{s/char}}]}{t_d[\text{s/char}]} \tag{4.1}$$

The error rate was calculated as the ratio of errors to the total number of characters. This was 27% ± 2.5% after the tutorial. The chording speed increased over the sessions with no signs of levelling off. The chording speed at the final session was 16.8 ± 2.5 wpm. The final error rate had fallen to 17.4% ± 0.6% with some signs of levelling off (Figs 4.4 and 4.5).

4.4.2 Keymap Learning Rate

The tutorial took an average of 80 minutes to complete. By the end of the tutorial, the subjects were able to enter text while only rarely needing to view the keymap. This indicates that the chords had been learned well enough for continuous text entry. This is supported by the fact that less than 4% of the first chording session was spent looking up unremembered chords (this was the first session for which this data was available). By the 10th session the amount of time spent looking at chords had dropped to 0.4% (Figure 4.6).

One subject was asked to return 3 and 6 months later for additional chording sessions. During these periods she did not use the Chording Glove at all. The subject's final chording speed after the tenth session was 16.2wpm. After three months, at the start of the session her input speed was 13.2wpm. However, this rose to 17.0wpm within 20 minutes. Her final chording speed was nearly 5% higher than her speed at the end of her last session. The subject spent 2.8% of the time looking up chords. This increase may be due to the change from the old, resistive foam sensors to the new metal plate ones (Section 4.2.2). Regardless, it is clear her chording speed did not suffer from the absence. Three months later, at the start of the session her input speed was 13.9wpm, and rose to 17.9wpm within 30 minutes. This is 10% better than her speed at her tenth session, and 5% better than her speed at her previous trial, three months before. The subject spent 1.2% of the time looking up chords. Since the same sensors were used in both return sessions, it is clear that her performance did not decay.

Figure 4.4: Average chording speed in WPM per session

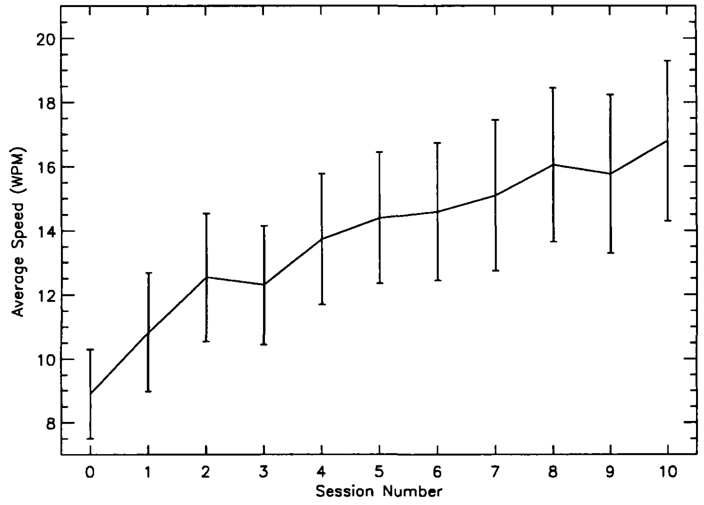


Figure 4.5: Average percent error per session

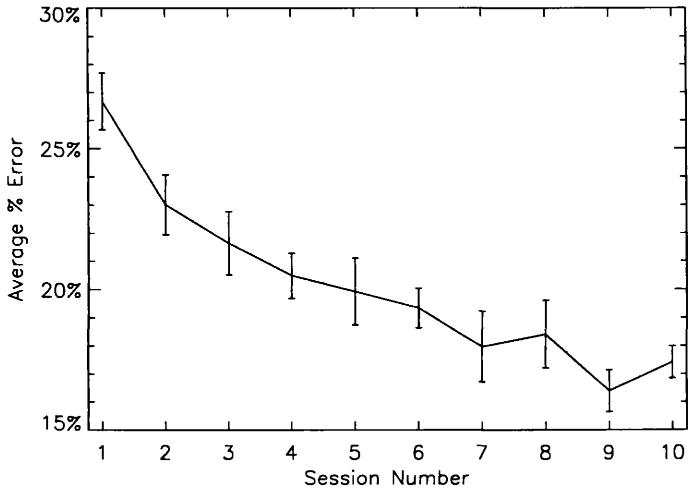
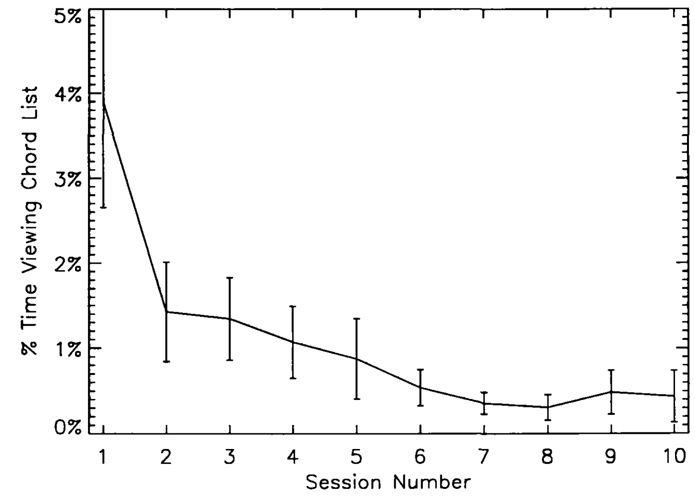


Figure 4.6: Average time spent looking up chords per session



4.4.3 Portability

To measure portability, four subjects were asked to perform a text entry trial while standing up. They were allowed to chord on any surface they wished (a desk, a file cabinet, the computer monitor, etc). The input speed of the standing-up trial was compared to the average input speed of the two other text entry trials in the same session, which were performed while sitting. The average difference in input speed was +0.47wpm ± 5.48wpm. The speed is so small compared to the standard deviation that there is really no significant difference in input speed while standing and while sitting.

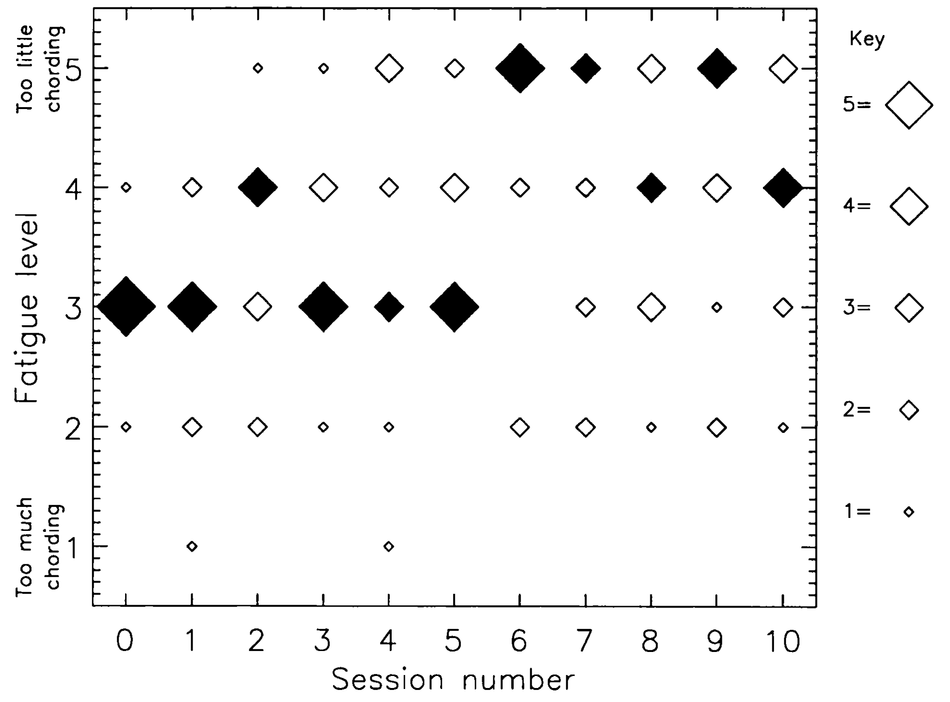
4.4.4 Questionnaire Results

The following is a summary and analysis of the subjects' responses to the daily questionnaire (Appendix C).

Fatigue

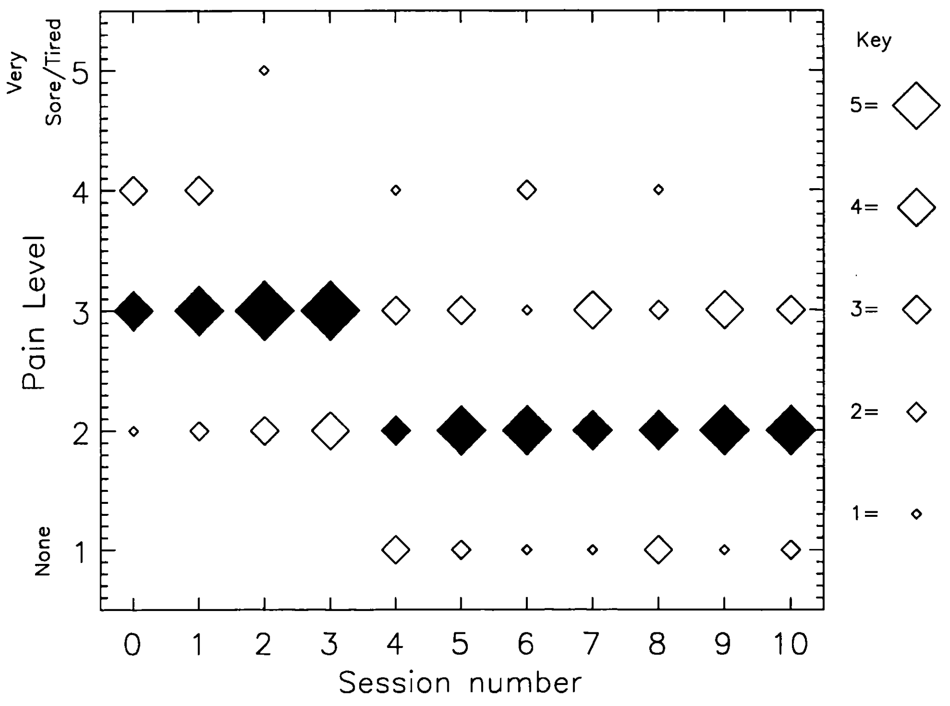
Fatigue was measured by asking the subject if they felt that they could chord for longer, had chorded too much, or if they had chorded for the right length of time. The scale was 1 to 5, where 1 was too much chording and 5 was too little. Since the fatigue levels are opinions, the scale is not necessarily linear. As a consequence computing an average value may not be meaningful. The most frequently occurring value was used instead. The mode fatigue level after the tutorial was 3 with some even distribution about that value. This corresponds to a "just about right" amount of chording. The value slowly tended upward over the sessions (corresponding to less fatigue) to the final mode value of 4 with some tendency towards 5 (Figure 4.7). A value of 4 means that they could have chorded for longer.

Figure 4.7: Distribution and mode of fatigue levels per session. The size of the glyph corresponds to the number of subjects who reported that level of fatigue. The filled glyphs are the most frequently reported fatigue level for each session.



Muscle Strain

Figure 4.8: Average muscle strain over time. The size of the glyph corresponds to the number of subjects who reported that level of fatigue. The filled glyphs are the most frequently reported fatigue level for each session



The level of muscle strain was assessed in the questionnaire given after each session. The subject was asked to rate any pain in their hand or arm on a scale of 1 to 5, where 1 was "No pain at all" and 5 was "Very sore/tired". The initial mode pain level was 3 with a tendency towards 4. This corresponds to a bit more than "some" pain. Over the course of the first five sessions more subjects reported less levels of pain (Figure 4.8). After the fifth session, the mode pain value levelled out at 2, where it stayed for the rest of the experiments. The final pain level was 2 with a slight tendency towards 3.

Preferences

After each session the subjects were asked to compare the comfort of the Chording Glove to writing and typing. These were measured on a scale of 1 to 5, where 1 was "much worse than typing/writing" and 5 was "much better than typing/writing". Neither of these values varied significantly over the course of the experiment. The final mode value for typing equally was distributed between 2 and 3, which means the subjects found it as good as or slightly worse than typing. The final mode value for writing was 2 with slightly more tendency towards 3 than 1. This means the subjects preferred writing over chording, but not by much.

The subject were also asked if they would want the Chording Glove for personal use. The scale for this was 1 to 5, where 1 was "Never!" and 5 was "Definitely". There was no significant change in the responses over the duration of the experiment. The final mode value was 3 with most of the rest of the subjects reporting 2, this corresponds to being somewhat less than "sometimes".

Opinions

Table 4.2: Top 5 positive and negative comments about the Chording Glove

	Likes		Dislikes	
	comment	*number*	*comment*	*number*
1	One handed use	5	Some chords hard to make	5
2	Visual supervision	5	Chording is too slow	5
3	Mnemonics/memorisation	3	General hardware problems	5
4	Portability	2	Straining to use	4
5	Similar to piano	2	Difficult to memorise	2

Table 4.2 shows the top five most common positive and negative comments made by the subjects over the duration of the experiment and the number of subjects who made those comments. The most preferred aspects of the glove were the one handed use and the lack of visual supervision in chording. The least preferred aspects were almost all due, at least in part, to the low sensitivity of the finger sensors (both old and new). The insensitive sensors made the chords which used more fingers more difficult to create. This slowed down the chording speeds and added to the muscle strain.

4.4.5 Analysis of Questionnaire Data and Final Results

Table 4.3 tabulates the results of the initial questionnaire, the experimental data, and the final daily questionnaire. The meaning of each column is described below.

Personal data Twelve questions were asked on the initial questionnaire (Section D.1.1). Three questions were answered the same by all subjects: All subjects were right-handed with no previous RSI or chord keyboard experience. The nine remaining questions are described below:

Subject This is a unique random number assigned to the subject.

Language This is the subject's native language.

Sex Male or female.

Glasses Does the subject wear glasses? Yes or no. All but two of the subjects wore glasses, making this variable unusable in a statistical analysis.

Music This is a rating of 0 to 5 of how well the subject can play a musical instrument. If the subject can play more than one, the highest rating is used. Zero corresponds to no knowlegde and 5 correponds to proficient.

Sign This is a rating of 0 to 5 of how well the subject knows any sign language. Zero is no knowledge and 5 is fluent.

Usage This is a rating of 0 to 5 of how well the subject can type on a keyboard. Zero is "never used one" and 5 is "very good at touch typing".

Table 4.3: Experimental data for each subject

Subject	Personal data								Experimental Data					Opinion data				
	Language	Sex	Glasses	Music	Sign	Usage	QWERTY	Frequency	Speed	Error	Help	Tutorial	Portability	Pain	Fatigue	Typing	Writing	Use
1	English	Male	Yes	5	0	4	60	4	19.3±11.7	20.6±2.0	0.99±1.48	112.1	—	3	4	2	3	3
2	English	Female	Yes	3	2	4	55	3	12.0±4.4	18.2±1.4	0.15±0.30	44.2	—	2	4	2	2	3
3	English	Female	Yes	0	1	4	18	3	21.7±8.4	14.3±2.6	0.04±0.03	77.7	+1.2±14.4	3	3	3	2	3
4	Spanish	Male	Yes	0	2	3	40	5	19.7±6.4	16.7±2.2	0.00±0.00	76.4	-0.74±7.87	1	5	3	2	4
5	English	Female	Yes	5	0	3	27	4	20.6±10.3	18.8±0.8	0.01±0.01	50.3	+1.4±11.4	3	2	2	2	4
6	Spanish	Female	Yes	3	4	3	30	3	16.4±8.1	9.3±1.4	0.00±0.00	77.5	—	2	4	2	1	4
7	English	Male	No	2	1	4	30	2	10.2±4.5	18.3±1.8	2.41±2.49	89.7	—	2	4	3	3	4
8	French	Female	No	4	0	4	40	4	16.2±6.8	27.7±2.9	0.48±0.64	120.0	—	2	5	3	2	3
9	French	Female	Yes	0	0	3	8	3	15.7±8.1	9.8±0.2	0.00±0.00	87.4	+0.01±8.96	1	5	2	1	3
10	English	Male	Yes	3	0	3	40	2	16.1±7.2	20.4±1.5	0.30±0.59	63.0	—	2	3	3	3	3

QWERTY This is the subject's self-reported typing speed in words per minute on a QWERTY keyboard. The validity of this is questionable, since it was not tested.

Frequency This is a rating of 0 to 5 of how often the subject uses a keyboard. Zero is "never" and 5 is "several hours per day".

Experimental data These are the final values for each person of the data collected in the experiment.

Speed This is the subject's average speed on the Chording Glove during their last session, measured in words per minute.

Error This is the subject's final ratio of incorrect characters to total characters.

Help This is the subject's final percent time spent looking up chords.

Tutorial This is how long (in minutes) the subject took to complete the tutorial.

Portability This is the change in the subject's chording speed between sitting down at a desk and standing up, in words per minute. A negative number indicates that they chorded faster when standing.

Opinion data These are the final values each person gave for the daily questionnaire (Section D.1.2.

Pain This is the subject's final reported pain level after using the Chording Glove for an hour. This is on a scale of 1 to 5 where 1 is "no pain at all" and 5 is "very sore/tired".

Fatigue This the final value of how tired the subject was reported to be after chording for an hour. This is on a scale of 1 to 5 where 1 is "too much chording" and 5 is "could chord for longer".

Typing This is the final reported value for the Chording Glove's comfort relative to typing. This is on a scale of 1 to 5 where 1 was "much worse than typing" and 5 was "much better than typing". This value did not change significantly over the course of the experiment.

Writing This is the final reported value for the Chording Glove's comfort relative to writing. This is on a scale of 1 to 5 where 1 was "much worse than writing" and 5 was "much better than writing". This value did not change significantly over the course of the experiment.

Use This is the final value for the subject's opinion if they would want the Chording Glove for personal use.This was on a scale of 1 to 5, where 1 was "Never!" and 5 was "Definitely". This value did not change significantly over the course of the experiment.

Table 4.4: Correlations of Personal, Experimental, and Opinion data. The value given is the square of the correlation coefficient. Values in **bold** are significantly different from zero at the 5% level. $n = 10$ for all calculations except for the following: [†] Subjects who had no musial experience were not used in this calculation ($n = 7$). [‡] Subjects who did not know any sign language were not used in this calculation ($n = 5$). [§] Only the subjects who took part in the portability tests were used in this calculation ($n = 4$).

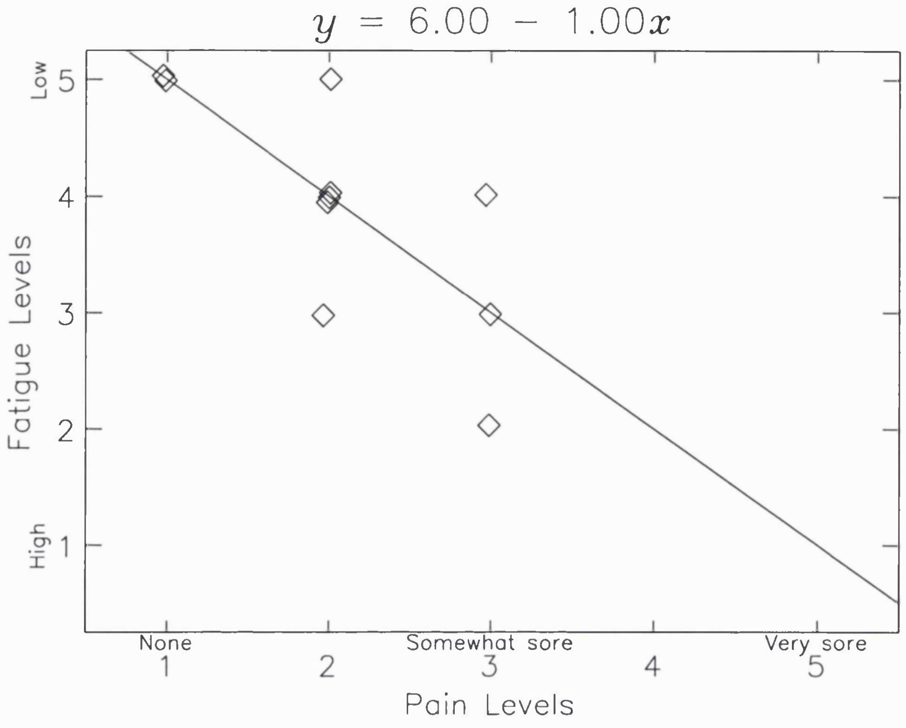
	Speed	Error	Help	Tutorial	Port.	Pain	Fatigue	Typing	Writing	Use
Music	**0.80**[†]	0.31	0.17[†]	0.005	0.38[§]	0.74[†]	0.14	0.14	0.12	0.139[†]
Sign	0.025	0.58[‡]	0.22[‡]	0.075	0.34 [§]	0.083[‡]	0.083[‡]	0.56[‡]	0.75[‡]	0.23
Usage	0.065	0.22	0.27	0.15	0.23[§]	0.18	0.011	0.040	0.18	0.17
QWERTY	0.020	0.33	0.033	0.005	0.11[§]	0.045	0.006	0.007	0.30	0.028
Frequency	0.36	0.043	0.15	0.043	0.26[§]	0.002	0.073	0.012	0.043	0.033
Speed	1.0	0.003	0.32	0.002	0.27[§]	0.15	0.10	0.000	0.018	0.000
Error	0.003	1.0	0.090	0.10	0.080[§]	0.075	0.000	0.17	**0.41**	0.067
Help	0.32	0.090	1.0	0.39	0.36[§]	0.014	0.007	0.083	**0.41**	0.036
Tutorial	0.002	0.10	0.39	1.0	0.35[§]	0.001	0.32	0.058	0.016	0.051
Portability	0.27[§]	0.080[§]	0.36[§]	0.35[§]	1.0[§]	0.90[§]	0.88[§]	0.074[§]	0.091[§]	0.025[§]
Pain	0.15	0.075	0.014	0.001	0.90[§]	1.0	**0.55**	0.20	0.15	0.014
Fatigue	0.10	0.000	0.007	0.32	0.88[§]	**0.55**	1.0	0.011	0.083	0.017
Typing	0.000	0.17	0.083	0.058	0.074[§]	0.020	0.011	1.0	0.18	0.000
Writing	0.018	**0.41**	**0.41**	0.016	0.091[§]	0.15	0.083	0.18	1.0	0.014
Use	0.000	0.067	0.036	0.051	0.025[§]	0.014	0.017	0.000	0.014	1.0

Relationships A series of regression analyses were performed on the data. Language, Sex and Glasses were left out of the analysis because they are categorical variables not suited to such an analysis. A table was generated of the correlations of all the variable. Personal data were not correlated against each other because any significance would tell nothing about the Chording Glove. Table 4.4 lists the squares of the correlation coefficients (r^2) for each of the variables. Bold numbers are significant at the 5% level. The details of the correlations significant at the 5% level are show in Table 4.5 and the results are analysed below.

Table 4.5: Summary of significant correlations

x	y	n	slope	S_e	r^2	T	p
Pain (explanitory)	**Fatigue** (explanitory)	10	-1.00	0.32	0.55	-3.13	<0.02
Music (explanitory)	**Speed** (dependent)	7	2.93	0.65	0.80	4.54	<0.05
Writing (explanitory)	**Error** (dependent)	10	4.66	0.28	0.41	2.34	<0.05
Writing (explanitory)	**Help** (dependent)	10	0.66	2.00	0.41	2.34	<0.05

Figure 4.9: Muscle Strain vs. Fatigue. For additional clarity, duplicate points are slightly offset.



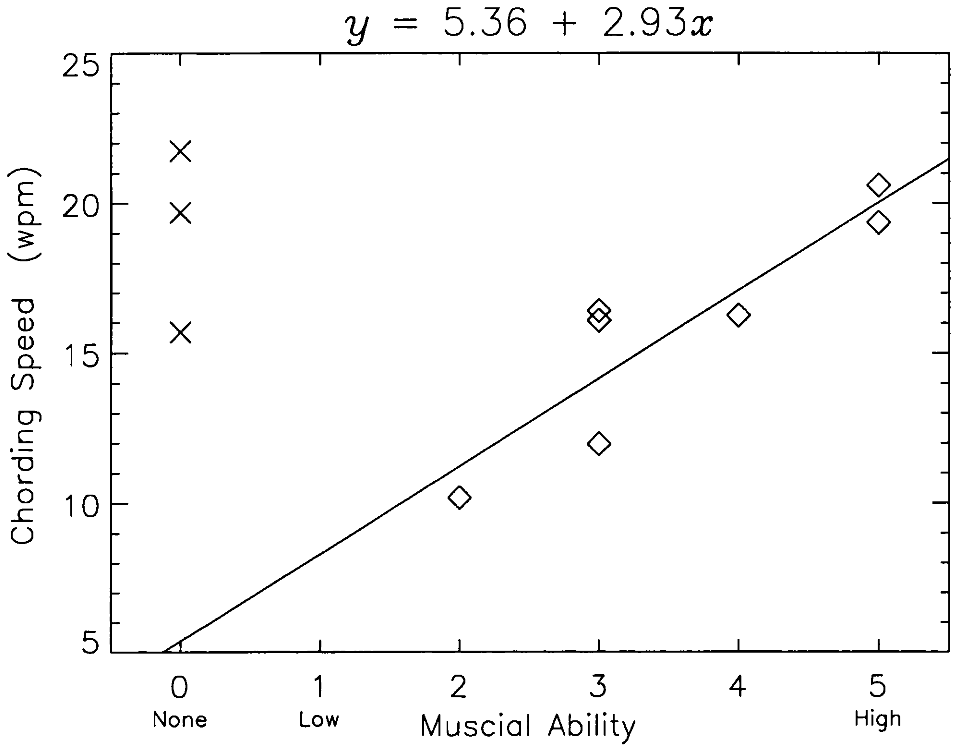
Muscle Strain and Fatigue The regression analysis gives a greater than 98% likelihood of a negative linear correlation between the levels of pain and fatigue (r^2=0.55) (Figure 4.9). The high degree of correlation is not unexpected. It states that those subjects who were uncomfortable chording became tired faster. The experimental results agree with what we would intuitively expect.

Chording Speed and Musical ability There appears to be a linear relationship between musical ability and chording speed. The greater the person's self-reported musical ability, the faster their final input speed (Figure 4.10). The correlation coefficient squared (r^2) is 0.80, which is significant at the 5% level. However this relationship only holds when those who do not play any instruments are weighted out of the calculations. When these people are included r^2 drops to 0.00.

The people who gave zero for an answer are those who do not play any instruments. One interpretation of these results is that practicing a musical instrument can help with the independent finger coordination needed for chording. However, this is obviously not the only way to achieve proficiency, as some of the non-musical people had fairly high chording speeds.

Writing preferences There was a positive correlation between the error rate and preference to writing significant at the 5% level (r^2=0.41). There was also a positive correlation between the help use and preference to writing which is significant to the same level at 5% (r^2=0.41). The relationship is counter-intuitive. It states that those who remembered the keymap less or made more mistakes preferred to use the Chording Glove, rather than write. There was not very much variation in writing preferences, as it ranged from 1 to 3, on a scale of 1–5. Half of the values were clustered at 2, while two were at 1 and three values were at 3 (Figures 4.11(a) and 4.11(b)) . Considering the poor distribution of values and the

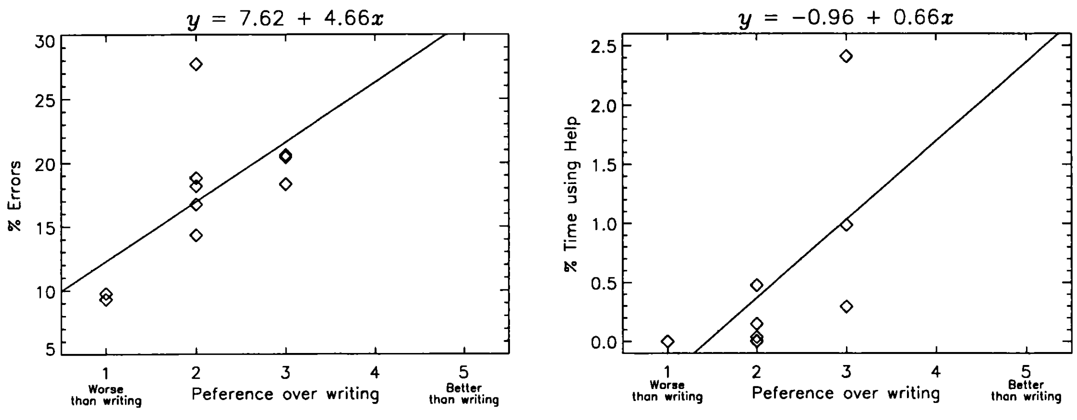
Figure 4.10: Chording speed vs. Musical ability. Diamonds (◇) are used for the people who know a sign language and are used in the regression. × is used to mark those who do not know any sign languages and are not used in the regression.



realtively low r^2 values, it is likely this relationship is coincidence.

4.5 Discussion

The subjects took an average of 80 minutes to learn the entire chord set well enough to allow continuous text entry. Most chord keymaps claim to take between 30 to 60 minutes to memorise (Gopher & Raij,



(a) Writing preferences vs. error rate (b) Writing preferences vs. help use

Figure 4.11: Correlations of writing preferences

1988; Kirschenbaum et al., 1986; Microsoft Corporation, 1995; Roberts, 1995). The time for memorisation for this keymap is slightly longer because the character set is much larger, containing all the characters from a standard keyboard. Most other chord keymaps use only the letters when measuring the time taken to memorise the keymap. Since just under half the tutorial is spent with numbers and punctuation, a rough estimate would put the time to learn just the letters at 40–45 minutes. This falls right in the middle of the standard 30–60 minute range.

The evidence suggests that it is faster to learn touch typing on this chording system than on a standard keyboard. After about an hour the subjects could chord without constantly looking at a guide. After 6 hours the learning curve levelled off, with subjects needing to look up a few seldom used chords. Touch typists often require visual supervision long after the keyboard layout has been memorised. Many casual typists are never able to type without visual supervision (Gopher & Raij, 1988).

The data also suggests that the keymap goes into long term memory within ten sessions. While only one subject was used to determine the long term effects, the results implied that a user returning to the Chording Glove after a long absence can quickly recall the keymap. In addition, the fact that the subject's performance improved both times she returned from an absence implies that the coordination skills are retained as well. To give a fuller understanding of the effects of long-term disuse, this experiment needs to be extended to include more users and testing over a longer period of time.

The above implications agree with the hypothesis that the learning and remembering the keymap for Chording Glove is no more difficult than for a normal chord keyboard. This further suggests that the keymap designed for the Chording Glove is as good as existing keymaps for chord keyboards.

After 11 hours of training the average input speed for the Chording Glove was 16.8 wpm. This is less than the expected value of 20wpm. However, this is higher than Gopher & Raij's (1988) value of 14wpm for the QWERTY. The slower input speed of the Chording Glove is due to the low quality finger sensors used in the experiment. They were not sensitive enough for general text input. In addition these sensors were also too large to allow comfortable freedom of movement. Smaller, more accurate sensors should increase the text input speed significantly. The steady state expected error rate is about 7%. We would expect the error rate after 11 hours to be slightly higher. The Chording Glove's experimentally determined error rate after 11 hours is 17.4%, which is more than twice the expected value. The inflated error rate can be explained by the low accuracy of the sensors which often caused problems with chords which used most of the fingers. Again, more accurate sensors should reduce the error rate to a more comparable level.

There was no significant difference in the subjects' speed when chording while standing or sitting. In addition, one-handed use, lack of visual supervision, and portability were three of the top four most-liked attributes of the Chording Glove. All of these imply that the Chording Glove has potential for an unobtrusive input device in a mobile environment.

Most subjects reported feeling some pain in their hand when they first used the glove. This occurred for the first few sessions. By the last session, the subjects claimed to feel some pain immediately after chording, but this quickly diminished. The reported pain level slowly decreased over the course of the experiment, implying that the discomfort they felt was temporary and would eventually disappear as they became acclimated to the glove. This is reinforced by the fact that, as the subjects performed more experiments, they felt more and more willing to chord for longer periods. None of the subjects reported pain in their upper arms or back during the course of the experiment, only the hand. However, as each session was only an hour long, there is no data as to the effects of prolonged use of the Chording Glove. This needs to be addressed in further experimentation.

The muscle strain and fatigue that the subjects felt lessened as they used the Chording Glove more, however the final values were still higher than desired. This is unexpected because research has shown that a chord keyboard should cause less exertion than a standard one (Kirschenbaum et al., 1986). Like error rate and input speed, it its likely that the extra strain was due to the low-sensitivity of the finger sensors, which required more pressure to trigger than they should have. Smaller sensors would allow the users to chord in more comfortable positions, reducing strain. Lowering the noise and increasing the sensitivity of the sensors should reduce the amount of work in chording, reducing fatigue.

Chapter 5

Testing the Biofeedback Pointer

5.1 Introduction

There are two basic questions about the Biofeedback Pointer which can only be answered empirically. The first is: Is the neural network adaptable enough that the Biofeedback Pointer can be used by different people? The second is: How well does the Biofeedback Pointer work compared to other graphic input devices? In order to answer these questions, a series of short tests were administered to 3 people to determine their indices of performance on a standard PC mouse and the Biofeedback Pointer. This would determine if the Biofeedback Pointer can adapt to different users and the range of performance one can expect from it.

In addition to the two main questions, there is the question of the structure of the neural network. Are there any trends in the weights from person to person, or even with the same person who has trained it more than once? Any trends discovered could be used to improve the neural network and thereby the performance of the Biofeedback Pointer. Comparing the networks generated by the subjects in the tests might show some potentially exploitable patterns.

5.2 Materials

5.2.1 The Mouse

The mouse used in this experiment was a two-button Microsoft™ serial, PS/2 compatible mouse. The mouse was used in the right hand on a mouse pad. All input was performed using the left button.

5.2.2 The Biofeedback Pointer

The Biofeedback Pointer was used as described in Section 3.3, with one exception: The device was designed only for moving a pointer, not for selection. Selection is intended to ultimately be done by using the shift buttons on the Chording Glove, but this was impractical for the performance tests, since it would provide much more hardware and software than would be necessary. Instead of using the Chording Glove, a mouse was held in the subject's left hand and was used to activate the buttons. Since the mouse was held, and not placed on a surface, it had no effect on the pointer's motion. This was found to be sufficient for the tests.

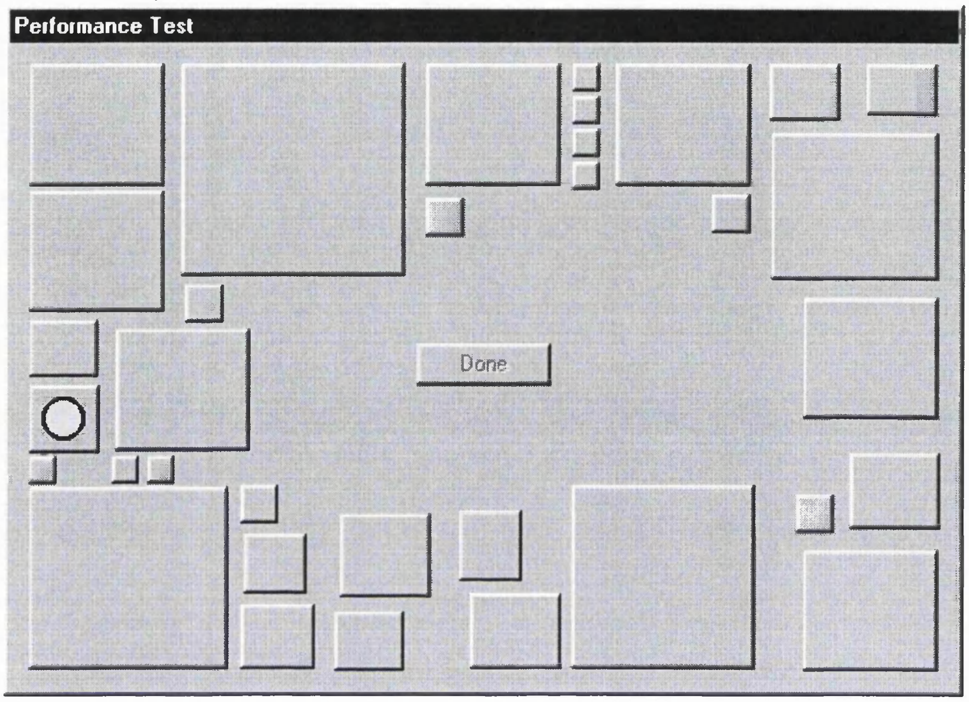
3M Red Dot™ pediatric electrodes were used in the tests. In preparation for use, the last $^1/_2$cm of the electrode was cut off to fit properly in the amplifier. One of each pair of electrodes was marked with the number of the corresponding amplifier.

5.2.3 The Performance Test Window Setup

The performance test was run using the EMG pointer software (see Appendix H) on a Pentium 133MHz computer in the Windows 95 environment. EMG data was read and filtered in software during both the mouse and Biofeedback Pointer trials. Even though this data was ignored in the mouse trials, calculating it ensured that any software delays brought about due to the computationally intensive calculations of the data filtering would be the same for both input devices.

The test window consisted of 30 near-square buttons of various sizes and positions (Figure 5.1). In addition there was one button labelled "Done" in the middle, which was disabled throughout the trial.

Figure 5.1: The button arrangement in the Performance Test window



When the trial was finished, the "Done" button would enable, requiring the subject to press it to end the trial.

In the Windows 95 environment, a click on a button widget is registered only when the pointer has stopped on the button. If the pointer is moving while clicking on a button, the focus is changed to the button, but it is not activated. There are few problems with this when using a mouse, because, in practice, the user always stops moving before clicking on a button. The mouse trials kept this standard method for button activation.

The Biofeedback Pointer could not activate buttons in the above manner. When using the Biofeedback Pointer, the user merely slows down when clicking on a button. Using the original method, the button would only become the focus, requiring the user to make a second or third pass before slowing down enough to have the click activate the button. As a consequence the behaviour of the environment had to be changed to accommodate the Biofeedback Pointer. The buttons in the performance test window were adjusted so that any click on a button would register as a full press and activate the button.

5.3 Method

The experiment consisted of two parts. In the first part, at least 2 trials were performed using the mouse. From these trials the mouse's index of performance was calculated. These same trials were repeated using the Biofeedback Pointer to calculate its index of performance.

The mouse trials consisted of just the performance test, as described in Section 5.3.2 below. No training or practice was necessary. Before the Biofeedback Pointer trials, the subject would need to train the pointer's neural network and then practice using the pointer. Training was done by the method described in Section 3.3.4. After training, the subject would practice using the pointer until they stated that they felt comfortable with it. During the practice, if the subject was not satisfied with the performance, they could retrain the pointer. However, the subject was not allowed to retrain after the first trial had started. This was to ensure that the same neural network was used for each trial. Each subject performed at least four trials.

5.3.1 The Subjects

There were three subjects used in the tests. Two of the subjects were female and one male. Different sexes were used because there are significant differences in amplitude and frequency distribution of EMGs between males and females (Cioni et al., 1988). All of the subjects were right handed and the right arm was used in all trials with the mouse and Biofeedback Pointer. In addition all the subjects had used the Biofeedback Pointer at least once before these tests. This was to help minimise the effects of the subject being unfamiliar with the device. Subject C was a computer science graduate student who had used the Biofeedback Pointer once before. Subject B had only a passing familiarity with computers and had also used the Biofeedback Pointer once before. Subject A was the author and had used the Biofeedback Pointer many times before.

Daily Preparation

Before each day of testing, the battery was tested. If it was significantly less than 9 volts it was replaced with a fresh one. This was to ensure that the amplifiers were giving their maximum performance. The next step would be to prepare the electrodes by clipping the ends and numbering them. After that the EMG collection hardware would be turned on and attached to the ADC on the parallel port. The last step before attaching the electrodes was to load and run the EMG Pointer software.

The electrodes were attached to the right arm in their standard locations (see Section 3.3.3). The reference electrode was placed over the acromium process in the right shoulder. The other electrodes were placed in the following locations: 1 - flexor carpi radialis, 2 - extensor digitorum, 3 - flexor carpi ulnaris, 4 - extensor carpi ulnaris (Figure 5.2). The electrodes would be attached by the following procedure:

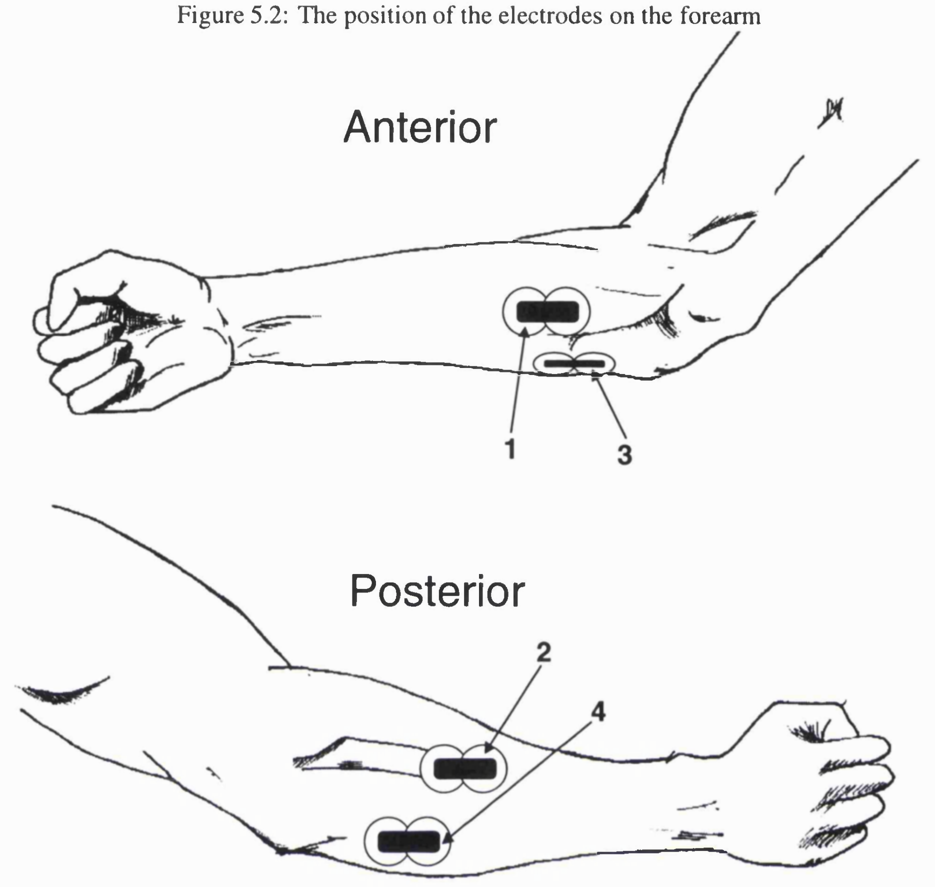
1. The data collection on the EMG Pointer software is turned on. This displays the input from each of sensors on the monitor (see Appendix H for more details).

2. The subject is asked to place the reference electrode over "the bony part" of their right shoulder. The subject was allowed to do this because exact positioning of this electrode is not important.

3. The experimenter attaches each electrode in turn by palpating the muscles in the subject's arm and then positioning the electrodes to give the best reading of the EMG displayed on the computer monitor.

The component box was usually attached to the subject's belt to keep it secure and out of the way. If the subject did not have a belt, they usually placed it in their lap while performing the test. This was not a problem as the subject would be sitting for the duration of the test.

Figure 5.2: The position of the electrodes on the forearm



5.3.2 The Performance Test

The performance test is made up of a series of button-pressing tasks of varying levels of difficulty. In each task, one of the blank buttons, selected at random is marked by a large yellow dot accompanied by an audible beep. When this occurs, the subject moves the pointer over the button and presses it as quickly as possible. The test then pauses for a random amount of time between 0.5 to 2 seconds before the next task. This is repeated 40 times. When the last task is completed, the "Done" button enables, accompanied by a beep. When the subject presses it, the window closes and the trial is over.

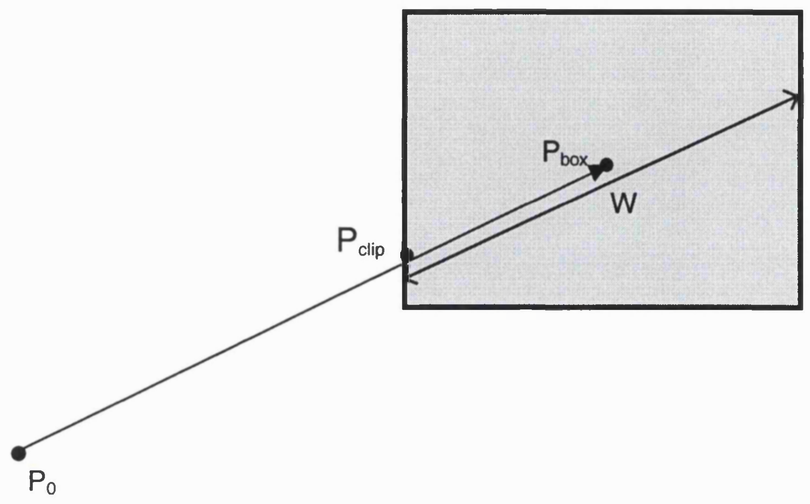
The software saves the beginning and ending time of each task, along with the entire path of the pointer during the test. In addition the neural network used for each subject is saved.

5.4 Results

5.4.1 Training

Each subject started to train the Biofeedback Pointer as soon as they were connected. After it was trained they were told to move around the pointer and see if they were happy with how it worked. If they were unhappy with it, they were allowed to retrain. Subjects B and C retrained twice, and subject A retrained three times before being satisfied with the neural network.

Figure 5.3: Calculating the effective width of the target. $\mathbf{P}_0$ is the starting position, $\mathbf{P}_{\text{box}}$ is the centre of the target. The vector from $\mathbf{P}_0$ to $\mathbf{P}_{\text{box}}$ is the approach vector. The point at which the approach vector intersects the target is $\mathbf{P}_{\text{clip}}$. W is twice the distance from $\mathbf{P}_{\text{box}}$ to $\mathbf{P}_{\text{clip}}$.



5.4.2 Index of Performance

Calculating W

The width (W) in Fitts' law is a one-dimensional term. The buttons in the performance test were two-dimensional with varying aspect-ratios ranging from 0.90 to 1.39[1]. MacKenzie (1992) addresses this problem and describes a few methods for determining the effective width. One common method described in the paper is to take the larger of the height or the width. This method was deemed too simplistic, despite its popularity and was passed over in favour of the method of using the span of the target along the approach vector. The approach vector is defined as the line from the position at the start of the task to the centre of the button. In this method, described in Figure 5.3, the approach vector is clipped against the button's edge. Twice the distance from the clipping point to the centre of the button is W. The specific details of the calculation are described in Appendix E.

Mouse

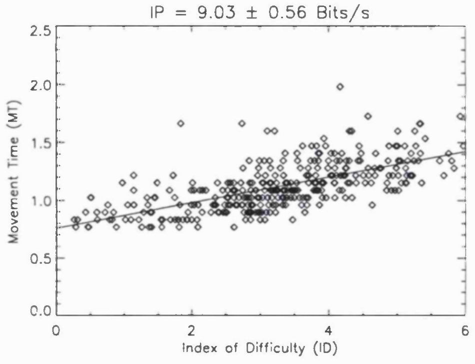
The Index of Performance (IP) for the mouse was calculated by the method described in Section 2.3.1. This consisted of finding the line which best fits the plot described by the movement time (MT) plotted against the index of difficulty (ID). IP is calculated by the inverse of the slope of the line. The y-intercept (a) corresponds to the reaction time.

The values for IP were 9.03 ± 0.56, 6.63 ± 0.96, and 5.65 ± 0.66, which average to 7.10. The reaction times were (in ms): 504 ± 72.0, 647 ± 76.0, and 759 ± 24.1, which average to 637. These can be seen in Figures 5.4(a)–5.4(c).

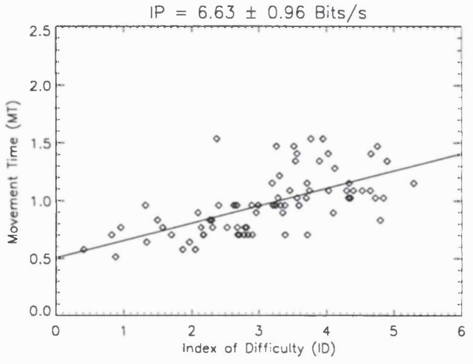
The values for the mouse described in MacKenzie (1992) range from a = 1030 and IP = 10.4 to

[1]Except for the "Done" button which has an aspect ratio of 3.26

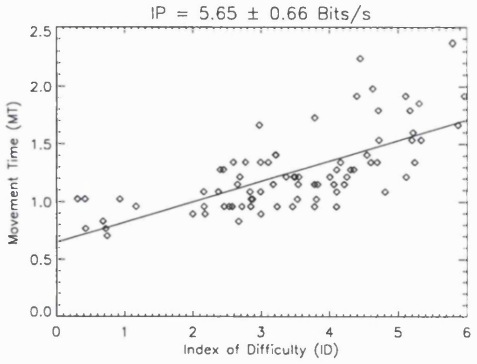
Figure 5.4: Index of performance for the mouse. The vertical grouping is an artifact caused by the fact that MT is measured in multiples of $^{1}/_{16}$th of a second.



(a) Subject A

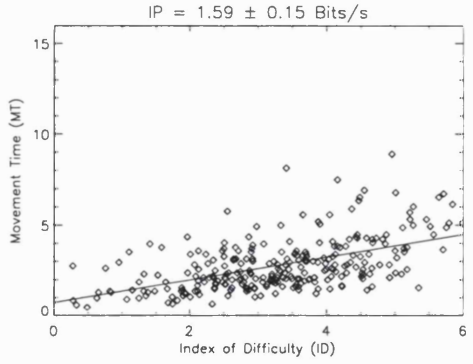


(b) Subject B

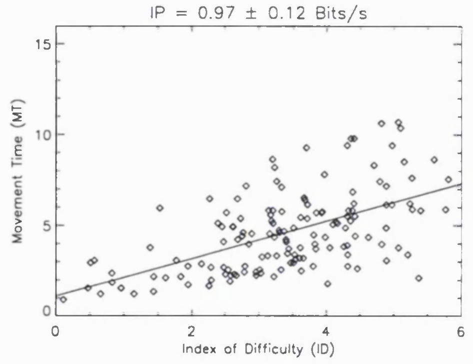


(c) Subject C

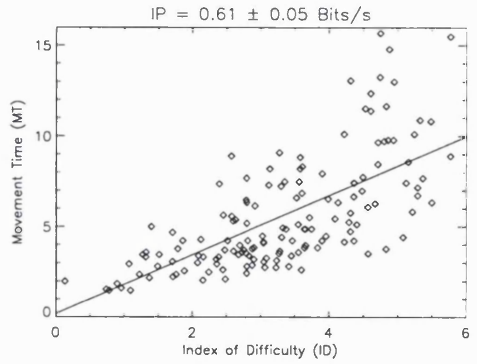
Figure 5.5: Index of performance for the Biofeedback Pointer



(a) Subject A



(b) Subject B



(c) Subject C

a = 108 and IP = 2.6. The values for IP and a from the experiment fit into this range well enough to be considered valid. The exact value of IP for the mouse is not as important as its relation to the IP for the Biofeedback Pointer.

Biofeedback Pointer

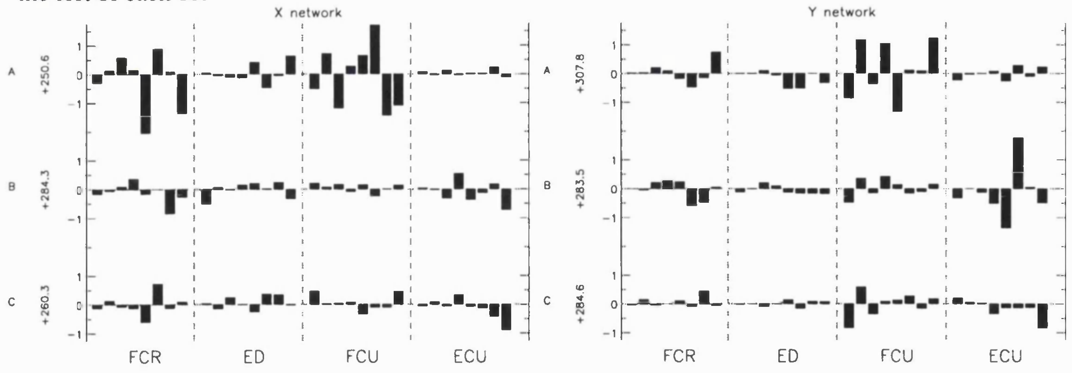
The IP for the Biofeedback Pointer was calculated in the same manner as for the mouse. The plots for each subject can be seen in Figures 5.5(a)–5.5(c). Table 5.1 summarises the values of IP and a for each subject. As the table shows, the IP for the Biofeedback Pointer rages from 11% to 18% of mouse's IP, averaging at 14%. The y-intercept value tends to vary much more, ranging from 27% to 219% of the mouse's value.

Table 5.1: Comparison of performance results for the Biofeedback Pointer and mouse

Subject	Mouse				Biofeedback Pointer				Ratios			
	IP_M		a_M		IP_B		a_B		IP_B/IP_M		a_B/a_M	
	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ
A	9.03	0.56	759	24	1.59	0.145	720	200	0.177	0.019	0.949	0.265
B	6.63	0.96	504	72	0.967	0.118	1101	454	0.146	0.028	2.186	0.953
C	5.65	0.66	647	76	0.611	0.053	174	503	0.108	0.016	0.270	0.779
Average	7.10	0.75	637	62	1.06	0.112	665	408	0.144	0.012	1.135	0.420

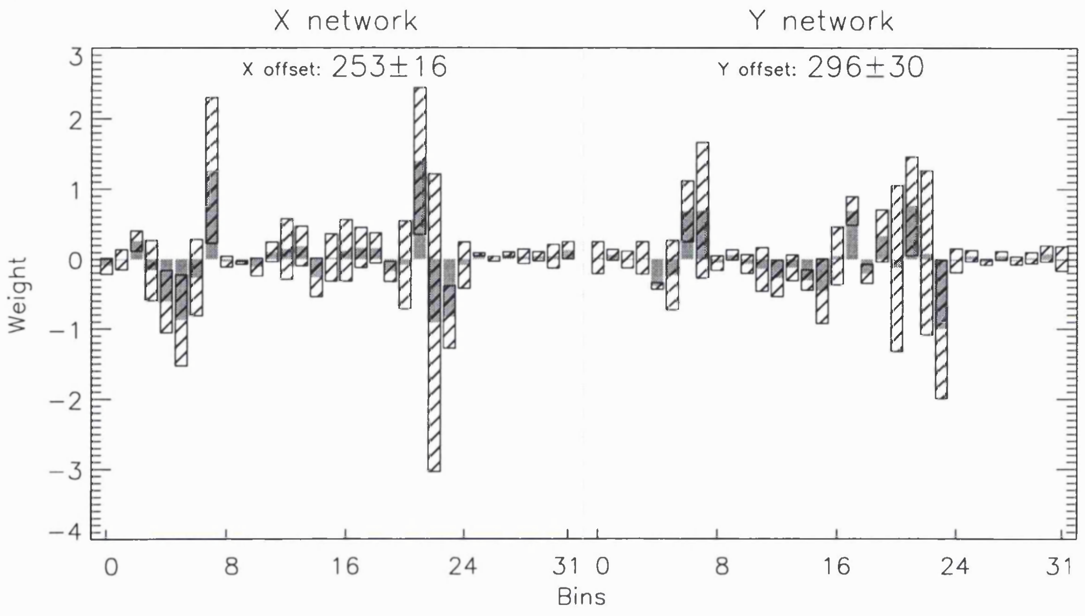
5.4.3 The Neural Networks

Figure 5.6: The neural networks weights used in the experiment for each subject. The offset is shown on the left of each set



Each neural network consists of 66 weights. One set of 33 weights for the x network and the other 33 weights for the y network. The first value in both sets of 33 weights is the offset value, which is added to the sum of the product of each weight and its input. The remaining 32 values consist of 4 groups of 8. Each group corresponds to one sensor, or more specifically, one muscle. The 8 values in each group are the weights for the power of the muscle's EMG spectrum which falls in the corresponding frequency bin. The value of all the weights for each subject are shown in Figure 5.6.

Figure 5.7: The average and standard deviation of the weights of 4 neural networks generated by subject A



In examining the weights we are looking for three potentially exploitable trends. The first is a consistency over the entire set of weights from one session to another, or one person to another. If the variation in the value of weights is small, this would indicate that averaging the weights together could improve the accuracy of the network. The second trend comes from determining if specific muscles tend to have larger or smaller weights. This would indicate the importance of the muscle to the overall network, and whether or not it is actually necessary to measure the muscle. The third possibility is a relationship between the weights and frequencies. It could be possible that certain frequencies are more important than others. For example, if the high frequencies turn out to be unimportant, a lower sampling rate can be used.

To start off, we examine the variation in weights for a single subject. Figure 5.7 shows the average and standard deviation of the weights for 4 networks trained by subject A. All of these networks were trained on the same day, without moving any electrodes. The solid bar shows the average value while the striped bar shows the standard deviation.

If the value of a weight is completely random, then the average of different sets would be zero. A significantly non-zero average shows that the weight is consistent between sessions. There are only 5 values out of the 64 weights which *might* be significantly different from zero at the 5% level. The origin of the x and y offsets is at (270,250), indicating that these values are not significant at the 5% level. The only inference one can make from this plot is that the high variability of individual weights means that a linear combination of multiple sets of weights will not yield a more accurate neural net.

Figure 5.8 shows the sum of the absolute values of all the weights for each muscle. This is averaged over the 4 networks generated by subject A. The absolute value is used to determine the importance of each muscle, not the effect. As the graph shows, the variation of each of the weights is much smaller than Figure 5.7 , with mean to standard deviation ratios ranging from 2.4 to 9.2. The height of the bar

Figure 5.8: The total of the absolute values of all the weights for each muscle, averaged over the 4 neural networks produced by subject A

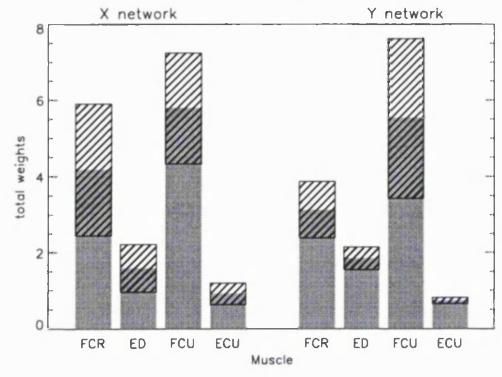
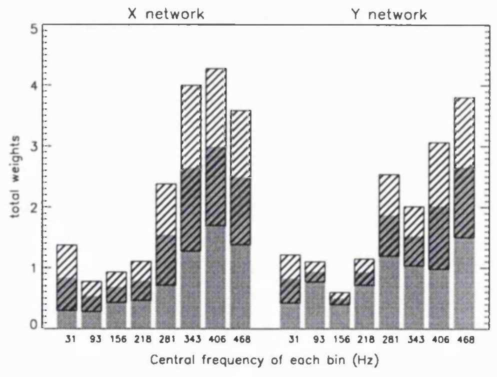


Figure 5.9: The total of the absolute values of each frequency bin, averaged over the 4 neural networks produced by subject A



is related to how much amplification the muscle needs. The ECU and FCU have smaller EMGs. The weights of the ECU are also small, implying that it has a small effect on the pointer. The FCU has a large weight, which means it is amplified by a fair amount. This could mean the FCU has anywhere from a small to a very large influence on the pointer, depending on the values of the EMG and the signs of the individual weights. The FCR has a large EMG which means that it is likely that the FCR has a large effect with regards to the pointer motion, but, like the FCU, it cannot be shown from these data alone. Another interesting point to note is that the muscles have a more-or-less equal relative importance in the x and y networks.

Figure 5.9 shows the weights grouped by frequency range. This is averaged over the 4 networks generated by subject A. The mean to standard deviation ratio of this plot is smaller than Figure 5.8, with a range from 2.1 to 5.7. The shape of the plot may be due to the fact that the spectra have higher amplitudes at lower frequencies, which drop off at high frequencies. This could imply that all frequencies of the muscle are important, but this cannot be known without further data.

Since there are trends in the neural networks which are consistent for one person, it begs the question if similar trends also can be found when comparing the neural networks generated by different people. The above analysis was applied to the neural networks generated by all the subjects in order to see if the same patterns, or any new ones emerged.

Two neural networks were used for each subject, one was used during the performance test, and one from a previous time. Both networks were recorded on different days, thus the electrode positioning may not have been exactly the same. All six networks were averaged together in the same manner as above. Figure 5.10 shows the average and standard deviation for all 3 pairs of networks for each subject. None of the values are significantly different from zero. This indicates that it is unlikely that there can be a "perfect" set of weights for everyone.

Comparing total weights for each muscle for all subjects (Figure 5.11) to that of just one subject (Figure 5.8) shows that the differences from person to person are larger than for an individual, which is what we would expect. An interesting point to note is that the plots for the x and y networks have similar shapes, quite similar to the pattern for a single subject (Figure 5.8).

Figure 5.10: The average of the neural networks of all the subjects

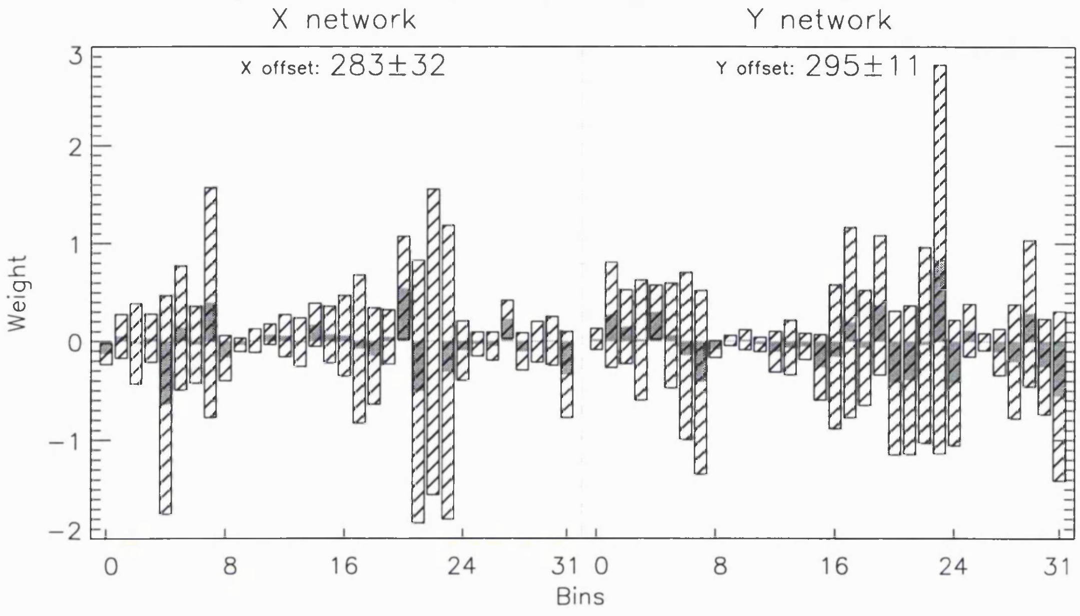


Figure 5.11: The total weights for each muscle, averaged over all subjects

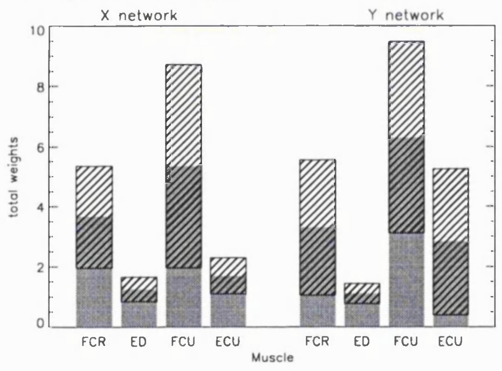


Figure 5.12: The total weights for each frequency bin, averaged over all subjects

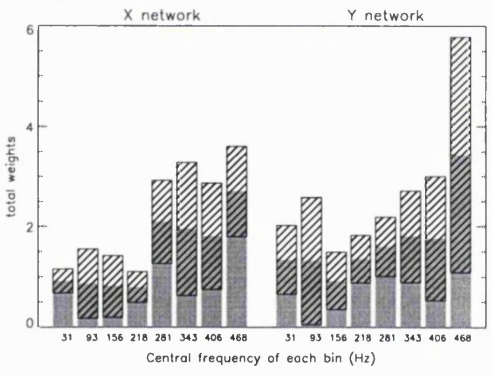


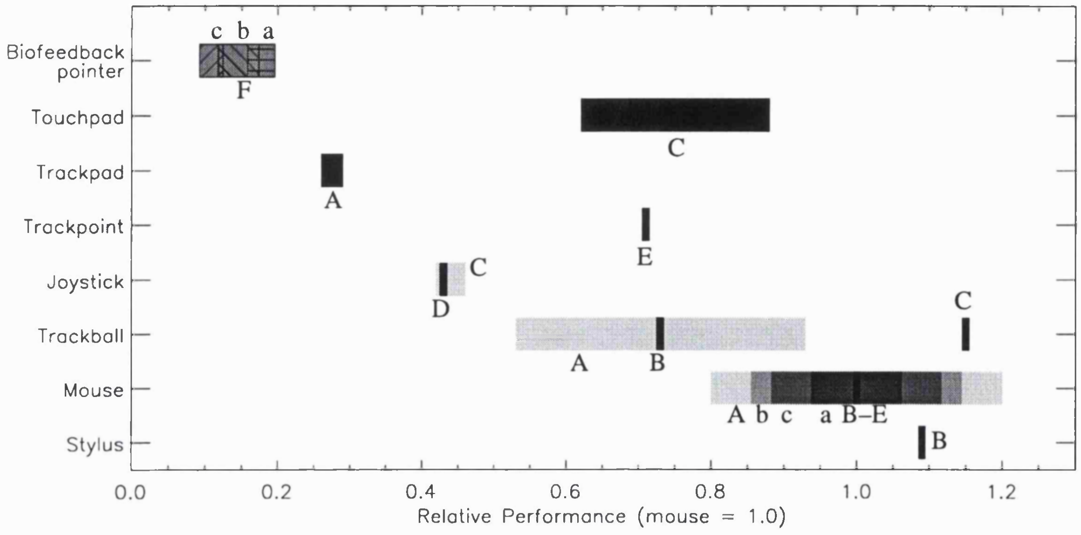
Figure 5.12 groups the weights by frequency range. Higher frequencies tend to have larger weights, and the x and y networks have similar shapes. There is a larger level of variation in the weights as compared to the frequency plot for a single person (Figure 5.9), but the overall shape is fairly similar.

5.5 Discussion

The Biofeedback Pointer was adaptable enough to be used by three different people. The neural network was able to cope with the individual differences between the subjects to provide an interface that each of them could use. While the tests were performed using too few people to generalise that the Biofeedback Pointer can be used by *anyone*, no problems have been identified which could prevent an able bodied person from using it. Further research must be done to observe the effects of differing body types (i.e. various levels of fat and muscle) on the device.

An analysis of the weights in the neural network showed certain trends when looking at the effect of muscles and frequency distribution. It was found in the overall analysis that there was too much variation

Figure 5.13: Relative performance on various graphic input devices. The colours are used to help differentiate between experiments. Notes: A = MacKenzie & Oniszczak (1998); B = I. Scott MacKenzie & Buxton (1991); C = Epps (1986); D = Card et al. (1978); E = Mithal & Douglas (1996); F (a–c) = This experiment (subjects a–c)



in each weight for an average of the weights to have any meaning. This means that multiple neural networks cannot be combined to create a more accurate network. In the muscle analysis it was determined that the ECU has the least influence over the outcome. This supports the theory mentioned in Section 3.3.2 that only 3 muscles are necessary to provide the 2 degrees of freedom. The frequency analysis shows that the weights are larger at higher frequencies. Since the high frequencies are low power, the larger weights are most likely to compensate for this. This implies that all frequencies are important to the final outcome. Had this been otherwise, and the higher frequencies had a low influence, then the EMGs could be sampled at a lower rate, which would be less computationally expensive and could use a slower ADC. As it stands, the full kilohertz sampling is necessary and a fast processor is required to handle all the computations.

The subjects' performance on the Biofeedback Pointer averaged at just over $^1/_7$ their performance with a mouse, which is fairly low compared to most graphic input devices (Figure 5.13). The implication is that, in its current version, the Biofeedback Pointer is not adequate to be used as a general pointer input. According to the chart, the Biofeedback Pointer's IP is also about half of the trackpad, which is a graphic input device common with notebook computers. This implies that the performance of the Biofeedback Pointer is not far off from an acceptable level. The Biofeedback Pointer does show promise and further research must be done to improve its performance by at least a factor of two.

Chapter 6

Future Directions

6.1 Sources of Error in the Experiments

6.1.1 The Chording Glove

In the Chording Glove experiment, the finger sensors were changed after it became clear that they could not handle long term use. This calls into question some of the data gathered in the experiments which used the old finger sensors. Two subjects used the old sensors for all their sessions, while three subjects used them for either 1, 3, or 4 sessions, making 28% of the experiment performed using the old sensors. It is worthwhile to understand the effect this could have had on the results. The change in sensors primarily could have effected the chording speed, error rate, or tutorial length. We will address each of these issues in turn to determine the potential effects.

Figure 6.1: Comparison of the average chording speeds including and excluding the old finger sensors

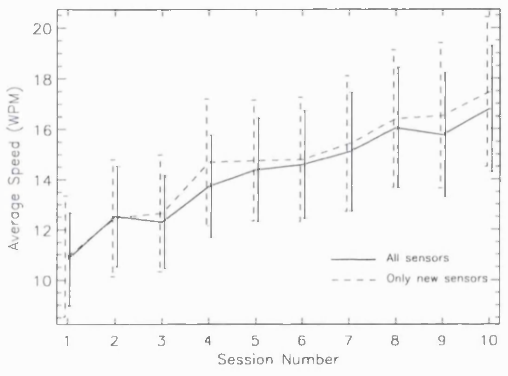


Figure 6.2: Comparison of the average percent error including and excluding the old finger sensors

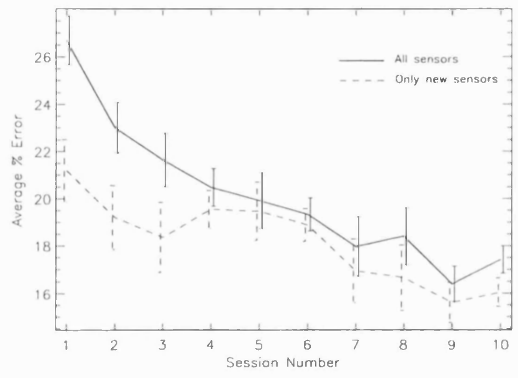


Figure 6.1 shows the average chording speed per session. The solid line is the speed for all the subjects. The dashed line excludes those subjects who used the old finger sensors. The plot shows that speed without the old sensors is consistently higher, although the difference between corresponding points is not statistically significant at the 5% level. The final input speed without the subjects who used the old sensors is 17.5 ± 3.0wpm. This is only slightly higher than the average calculated with all the subjects, 16.8wpm ± 2.5wpm.

Figure 6.2 shows the average percent error per session. Like speed, the error is consistently lower when the subjects using the old sensors are eliminated from the calculation. The difference between the cases is statistically significant at the 5% level for all except the middle 3 sessions. An interesting thing to note about these values is the rise in the error for these sessions. The rise comes in sessions 4 and 5, the same time at which two subjects switched over from the old sensors to the new sensors. If these subjects are removed from all the sessions, the rise disappears. This implies that switching between the two sensors requires a period of adjustment before the error rate drops to the lower value. The final error rate for all the subjects was 17.4 ± 0.6%. Excluding those using the old sensors gives a final error rate of 16.0 ± 0.6%.

The average time it took to perform the tutorial with the old finger sensors was 86 ± 32 minutes. The average time with the new sensors was 74 ± 14 minutes. Again, the new sensors performed better, but the difference is not statistically significant at the 5% level.

These results imply that if only the new sensors were used the Chording Glove's performance would have been higher, but not by very much. The error rate was affected most by the use of the old sensors. The difference in the final error rates is 1.4%. This is not unexpected, because the poor sensitivity of the old sensors served to increase the likelihood of mistakes. The plot of the overall performance of these subjects was the same shape as the other subjects, just worsened by the old sensors. Including these subjects does not significantly degrade most of the results, while using them gives more information on the effect of performance over time and other aspects of the Chording Glove not effected by the finger sensors, such as the learning rate.

The results of the portability tests tended to show no difference between the speed while standing or sitting, but these results have their limits. First, the lack of a mobile computer in the original experiments prevented any situations more mobile than standing up to use the glove. A continuous mobile environment is expected to provide much more demanding situations than this. The second limitation to the experiment was the lack of a control. Would the same experiment done with a QWERTY keyboard show no difference in performance as well? This needs to be looked into. The last problem is the low number of subjects used in the experiment. While there were enough to show that there was no significant difference between the speeds, a larger sample size is needed for the more complicated experiments necessary to prove the extent of the portability. The experiment served to show that the device is portable in one of the simplest portability tasks one might encounter in a mobile environment. If the Chording Glove failed this test it would imply that since it could not handle the easiest portability task, it would be unlikely to suffice for a more intensive task, and no further experimentation need be done to show this. Given the results of the experiment, we know that further testing is necessary and that this experiment needs to be redone with a control, more subjects, and in more demanding situations which require use of a mobile computer.

6.1.2 The Biofeedback Pointer

The first question that comes to mind regarding the Biofeedback Pointer tests is the low number of subjects. These tests were designed as a proof of concept to show that the Biofeedback Pointer works, and will work for more than one person. This only requires two subjects. Three were used to get a better idea of the variability of results. It is true that there were too few subjects to obtain statistically significant results, but the tests were only intended to determine the relative performance, not its feasibility as an input device. It is hoped that by using the results of this experiment, a new prototype can be built which performs closer to the desired level of a graphic input device. Later in this chapter we will discuss potential directions for such research.

Another potential problem regarding the subjects is that the creator of the device was used in the tests. The rest of the subjects used were near-beginners, having only used the Biofeedback Pointer once before. This would give a good idea of the starting performance range of the device. Since no one was more experienced or familiar with the device, the creator was used to give an idea of the expert level performance. This is justified by the fact that knowing the internal workings of the performance test experiment could not have effected the results in any way.

In this experiment, the width of the target (W) was calculated by taking the span of the button along the approach vector (see Section 5.4.2). The potential problem with this method is that the subject will not always move straight toward the target, but may approach via a curved path, which could place their true approach vector perpendicular to their estimated one. In the worst case, the estimated approach vector is perpendicular to the longest side, giving the smallest W of W_0 and the real approach vector passes through one of the corners, giving the largest W of $W_1 = W_0\sqrt{r^2+1}$. Where r is the aspect ratio. This makes the "real" index of difficulty:

$$ID_1 = \log_2 \frac{2A}{W_0\sqrt{r^2+1}} \tag{6.1}$$

where the calculated index of difficulty is $ID_0 = \log_2 \frac{2A}{W_0}$. Simplifying and calculating the potential change in ID gives:

$$\begin{aligned} ID_1 &= ID_0 - {}^1\!/_2 \log_2(r^2+1) & (6.2)\\ ID_1 - ID_0 = \Delta ID &= -{}^1\!/_2 \log_2(r^2+1) & (6.3) \end{aligned}$$

Except for the "Done" button, the aspect ratio of the buttons varies from 1.0 to 1.39, which means that ΔID is at most -0.78, and therefore the ID used in the experiment will be 0.78 bits less, or, in other words, slightly less difficult. Odds are that the errors will be random and cancel out, but we will look at the worst case situations to see the maximum effects. If the all the lowest IDs are off by -0.78 and the highest are all correct, this would alter the range of ID from a width of about 6 bits to 6.8 bits. The opposite effect gives a width of around 5.2 bits, which is a 13% difference in either direction. Since MT is the same, this means that IP could be off by as much as 13% in either direction. The mean value of the normalised IP would then range from 0.125 to 0.163, which is only slightly larger than the standard deviation. In the worst case, this error would only increase the standard deviation by 60%, which is small enough for the results to remain valid.

Another potential problem with the experiment was that number of trials was not the same for all the subjects. Each subject performed at least four trials on the Biofeedback Pointer before they were given the option to leave. It was believed that four trials would be enough to generate a respectable number of points. Any number of trials beyond this should only improve the accuracy of the calculation and should not do any harm. Each of the subjects' data were analysed separately, meaning that the number of trials of one should have no effect on the results of the others.

A related issue is that there were fewer mouse trials than Biofeedback Pointer trials. The reason for this is that the IP of the Biofeedback Pointer was more important to calculate, so more time was spent performing trials on the Biofeedback Pointer. The smaller number of mouse trials might account for a slightly larger standard deviation in the mouse's IP. The average ratio of σ_{IP} to IP for the mouse is 0.108, while for the biofeedback pointer it is 0.100. However, this difference is unlikely to have any effect, since the experiment was only to give an idea of that values of IP could be expected.

The largest potential source of error in the Biofeedback Pointer experiment was the occasional trouble with the window manager during the experiment. The computer used was a Pentium 133MHz. The computationally intensive task of analysing the data was near the limit of the processor's abilities. The performance test application was run in the Windows 95 operating system. Data reading was done as a thread run by the OS, once every millisecond. Every 64 milliseconds, the data analysis was performed, repositioning the pointer. This thread had the highest priority. Lower on the priority level were window messages, such as button clicks. As a consequence, sometime the OS would get too overwhelmed by the calculations and other necessary maintenance that a button click would get delayed. The effect of this would be that the user would click on a button, and nothing would happen. Often times they would think they missed and try clicking again. These events would happen anywhere from zero to six times per trial, but usually occurred only once or twice. It was noted when these events occurred and the bad data were deleted from the calculations. It may be possible that a few short delays were unnoticed and the corresponding data were not deleted. This would have the effect of an unnecessarily high MT, which would very likely lower the IP. Beyond this problem, the most noticeable effect would be the confusion and annoyance of the subject, which may have effected their subsequent performance in that trial.

6.2 Future Work

6.2.1 The Chording Glove

Future Experiments

There are a few questions left unanswered from the Chording Glove experiment than can only be answered by further experimentation. The first is the time to learn the alphabet part of the keymap. The time to learn the entire set of 97 chords was found, but most chord keyboards tend to specify learning time in terms of just the alphabet, not all possible characters. Further experimentation must be done to determine this learning time to allow a fair comparison between the Chording Glove and other chord keyboards.

Another question the experiment left unsatisfactorily answered was the keymap retention. Only one subject was used to determine this. The exact change in her performance is in doubt due to the change of sensors between her last session and when she returned, but there is no doubt that one person's performance did not suffer after a three month absence. In order to sufficiently determine the long term effects of absence, this experiment must be repeated with a larger number of subjects.

The experiment did not use the full functionality of the Chording Glove in order to limit the scope of the experiment. This left several aspects of the Chording Glove untested. The only one of the function keys which was tested was `<Help>`. The rest of the function keys, and the use of `<Control>` were are untried. Are `<Help>`, `<Pause>`, `<Escape>`, `<AutoCaps Toggle>` and the arrow keys the only ones truly necessary? Would more be useful? How many function keys can conveniently fit on the hand? What about the possibility of using a "soft" function keypad made from a small touchscreen? The full range of possibilities has yet to be explored and needs to be researched.

Improving the Chording Glove

The area with the most vital need for future research on the Chording Glove is the finger sensors. It is clear that the current sensors are insufficient for commercial use. The experiment shows that the sensors need to be smaller than the current metal plate ones in order to allow more flexibility in hand-orientation while chording. The activation pressure needs to be less to reduce fatigue, and the accuracy must improve to allow faster chording with fewer errors. A soft sensor, like the foam ones described in Appendix C, would be ideal, since they could be made light and flexible, and possibly even sewn into the lining of the glove, causing minimal disturbances to real-world actions. Unfortunately, all the foam sensors tried so far were insufficient for this task. The most promising alternative for the sensors would be miniaturised switches or sensors based on capacitive technology like that used in touchpads.

The use of the function keys was briefly mentioned in Section 3.2.1, but not in any great detail. The use of <`Help`>, <`Escape`>, and the arrow keys translates easily from their use with normal keyboards and requires no further explanation. <`AutoCaps Toggle`> and <`Pause`> both act in somewhat different ways than one would expect from a normal keyboard and need further explanation.

The AutoCaps feature has been mentioned before as a method for automatically capitalising the next word when a period, question mark, or exclamation point are pressed. This is to allow the user to continuously type sentences without stopping to press the shift at the start of each one. The usefulness of such a feature is untested. It may improve performance, but may get in the way as well. The AutoCaps idea may be extended to the possibility of using a predictive keyboard style input. In this method, the most likely ending for a word is displayed on the monitor. This is updated with each letter entered. If the predicted word is incorrect, the user keeps entering more characters. If the word is correct, the user presses a trigger key, (for example, double-pressing <`Space`>) and the word is completed for them.

There needs to be a fast and easy way to turn off or ignore the Chording Glove's input to allow interaction with the real world. As it stands, pressing the <`Pause`> function key will cause the glove to ignore all the finger sensors until <`Pause`> is pressed again. This requires use of the other hand, which may be inconvenient. Another alternative is that the <`Control`>-`S` sequence could be used to stop input until either <`Pause`> is pressed or the <`Control`>-`Q` sequence is made. The <`Control`>-`S`, <`Control`>-`Q` pair is a good choice for two reasons. The first reason is that this sequence is currently in common use as the pause-unpause commands for many existing systems. The second is that <`Control`>-`S` can be performed very quickly. The <`Control`> sticky shift is first pressed by the thumb. Then the sensors on the thumb and middle finger are pressed together to make `S`. This can be quickly done, even without a surface to chord against. <`Control`>-`Q` is a good choice because `Q` is one of the harder to make chords and thus less likely to be made by accident.

One possible alternative to the suggested system is to place an entire handheld-style computer on the back of the hand, in the current location of the function keys. This would be used for graphic input and display while the Chording Glove would provide text input. The entire system would be contained on the glove, instead of distributed about the person like a wearable computer. The first question which arises is the performance of the non-preferred hand for controlling the graphic input. Non-preferred hands work as well as dominant hands when performing gross tasks such as scrolling, but leaves something to be desired for tasks which require detailed pointing (Kabbash et al., 1993). Another possible problem would be placing the display on the back of the chording hand. The hand's orientation while chording may partly occlude the display, or the movement involved in chording might make it hard to focus on the display.

6.2.2 The Biofeedback Pointer

Future Experiments

Selection Control In the experiment, selection was performed by using the buttons of a mouse with the left hand. This method was only used to test the use of the pointer and is inappropriate for general usage. The buttons need to be operated by the same hand as the pointer. It is intended that the shift buttons on the Chording Glove are to be used as selection buttons for the Biofeedback Pointer. These buttons are mounted on the index finger and pressed by the thumb. This leads to the important question: Will the action of pressing a button with the thumb be interpreted as pointer motion? This would create jitter when pressing a button, similar to the jitter problems of a touchpad. The muscles used to move the thumb tend to be either deep or located far from the electrodes sites. The flexor pollicis longus is the only muscle which might interfere (see Table 3.8), but it is likely to have no effect because it is occluded by the superficial muscles in the upper forearm in the area where the electrodes are positioned. Preliminary tests show that only strong actions of the thumb cause visible jitter when using the Biofeedback Pointer. Experiments must be done to show the likely levels of thumb use and to see if interference is a potential problem.

Alternative Muscle Groups It is possible to use other muscle groups to control the pointer without any changes in the hardware or software. If other factors weigh more than the limitations imposed by convenience as mentioned in Section 3.3.2, one can theoretically place the electrodes anywhere they can acquire a reliable signal.

The fingers, eyes, and head are particularly interesting sites to choose for pointer control. If the fingers are used, the electrodes can be placed in more or less the same location as for the wrist, since the muscles which move the fingers can be sensed from there. The only problem would be abduction and adduction of the fingers. These are difficult to measure since the muscles which control them are located in the hand. Using smaller electrodes might avoid this problem.

The Biofeedback Pointer could be used for eye tracking. According to Gips & Oliveri (1996), the voltage per degree of arc is around $20\mu V$. The Biofeedback Pointer can measures voltages in multiples of $2.5\mu V$, thus being able to discriminate down to 7.5 minutes of arc. The eye angle is a function of the difference between two of the electrode sensors. This is a linear combination of the electrodes which can be generated by the neural network. There is no reason why the Biofeedback Pointer would not be able to be used as an eye tracker.

The movements of the head and neck also can be easily measured. Like the wrist, the rest location in is the middle of the motion. The size and shape of the electrodes currently used may make them difficult to position over the necessary muscles, but smaller ones may be sufficient.

Any other body part can be used as long as the EMG is conveniently measurable and the user can consistently control it. It is theoretically possible to set up a system in which left and right motions are controlled by flexion of the left and right elbows respectively, and up and down motions are controlled by flexion of the left and right knees. This setup is easily measurable, but the user may find it hard to visualise, causing difficulties in training and using. However, there are no hardware or software problems preventing such a setup. Given sufficient time and need, the Biofeedback Pointer should be operable with just about any set of muscles.

Improving the Biofeedback Pointer

Larger motions generated by the Biofeedback Pointer are scaled up by a factor of 3. This is an arbitrary scaling which was chosen for its ease of calculation and success in empirical testing. However, this is not the only option. The piecewise function used (Equation 3.9) is first order continuous, but not second. The scaling factor jumps from 1 to 3 for $r \geq cs$. It may be better to use a second order continuous piecewise function (splines) or even a non-piecewise function like r^2 or a higher degree function. This must be further investigated in order to improve the efficiency of the Biofeedback Pointer.

Another important factor in the performance of the Biofeedback Pointer is the training. Currently training is done by moving the wrist through each of the basic directions in the theory that if the neural network can handle those, it can handle any input given it. One problem is the reaction time in following the cursor. This is compensated for by a lag component in the calculations, but the lag may not be consistent throughout the training period. Also the user's motions may not be as precise as necessary. For example, the program expects the motion to be a diagonal of 45°, but the user might move at only 15°. A solution for these problems must be found. One solution is to perform more than one set of motions. This might compensate for the inconsistencies, but may require a fairly long time to perform. Another solution is to put a motion sensor on the hand to get the exact orientation of the wrist. This is similar to the approach tried by Hiraiwa et al. (1993). It would be preferable to avoid this method because it requires extra software and expensive hardware. A simpler method might involve using audio cues to help synchronise the screen motion with the subject. What we need from an improved set of training data is a relatively short training process and to require no external hardware beyond the current system. In addition the training process needs to use a set of motions which are easier to follow with less lag, and less possibility of deviation.

Improving the recognition of the neural network is essential in improving the accuracy of the Biofeedback Pointer. What we would like is to find a way for the neural network to be improved after the initial training session. At the moment the network can only be retrained from scratch. The tests implied that averaging different sets of weights would not yield a better set. This was confirmed directly in an informal test where the average of multiple sets of weights was found to be completely unusable.

It is possible that another kind of linear combination, or even a non-linear one may work in combining the weights. Another solution is to find a less computationally expensive back-propagation neural network. The BPN mentioned in Section 3.3.4 took up most of the computing power of the computer, required too much training, and gave insufficient accuracy. It may be possible to use another type of BPN, or even a genetic algorithm-based system, which would be fast and accurate enough to use.

Another option is to change the electrode setup. Since the ECU was found to be used least of all the muscles, it may be possible to find a better position for the electrode, which will weigh more in the neural network. Alternatively it may be useful to remove the fourth amplifier completely and just use three in the calculations, reducing the amount of work for the computer. It may also be possible to move the electrode to another site (e.g. the thumb) and use it to recognise gestures for selection. This would provide a system using biofeedback for pointer control and selection.

One advantage of using the current set of muscles is that they are located in a circle around the arm, just distal to the elbow. In this arrangement, the connectors can be placed on an arm band, which can be easily affixed to the proper location, facilitating measurements. In addition this provides a much tidier package, since there would be only one bundled wire containing the four signals coming from the arm band, as opposed to four separate wires from various locations on the arm. Ultimately the hardware from the main box will be merged with the Analogue/Digital converter and put on a board or PCMCIA card. Moving the FFT from the software to specialised hardware would free up a great deal of memory and computational power. This would yield a very small device consisting of an arm band which plugs into the computer. This could be worn underneath clothing, allowing a very discrete method for interacting with the computer.

6.2.3 General Input Device Research

Combining the Biofeedback Pointer and Chording Glove

The Chording Glove and Biofeedback Pointer are intended to make a unified graphic and text input system for a wearable computer. There is one major unanswered question we need to solve before this can be done: How can we switch between graphic and text input? The finger motions used in chording will doubtlessly be interpreted by the Biofeedback Pointer as motion, at least through the extensor digitorum, if not other muscles. This is not so much of a problem, since the nature of the devices precludes their simultaneous use. The Biofeedback Pointer requires the wrist to be free to move, while the Chording Glove needs to orient the wrist next to a solid surface. The problem is how to switch between the two quickly and easily. There are a few options for this. The first option is to use a function key on the back of the hand. This is undesirable because it would sometimes require constant use of the other hand, eliminating the one-handed benefits. The second option is to use a special chord key like `<Control>-A` or double-`<Space>`. The former has the benefit of being able to be made without a typing surface, but it may not be comfortable or fast enough to perform as often as might be needed. Double-`<Space>` is likely fast and easy enough, but it may be desired for use by another action like predictive text entry. It also has the disadvantage of needing a surface to make the chord.

Another option would be to use a gesture. It is possible to have a quick and easy to recognise gesture be interpreted to mean "switch input devices". One such gesture could be to quickly flex and extend the fingers (or just one finger). Since this uses the extensor digitorum and flexor carpi ulnaris, it is easy to detect with the existing setup. However, gesture recognition software must be added. Experimentation must be done to determine if this is a viable option.

A final alternative is to use the Chording Glove on one hand and the Biofeedback Pointer on the other. The Chording Glove should be used with the dominant hand to enable maximum performance for text input. The non-dominant hand should be used to control the Biofeedback Pointer. Kabbash et al. (1993) determined that the non-dominant hand is just as effective as the dominant one for low-precision tasks. Since the Biofeedback Pointer is ideal for those sort of tasks, it should be just as usable in the non-dominant hand. The Biofeedback Pointer also has the advantage that the hardware is hand-independent, unlike the Chording Glove. This means that if use by one hand is ineffectual, it can be easily switched to the other one.

In situations where both hands are expected to be free to use the PC, there may be increased performance from being able to enter text and control a pointer at the same time. As mentioned in Section 3.2.4, evidence exists showing the benefits of performing tasks in parallel. Not all tasks can be made more efficient by performing simultaneously. The most efficient tasks are those in the non-dominant had supports or alternates tasks with the dominant hand. This does not work well in tasks where the hands perform the exact same tasks, or completely different tasks (Buxton, 1995). While it is possible on a desktop computer to use a mouse and keyboard simultaneously, this is done very rarely in practice. This could be interpreted in two ways. First that there is little practical need to perform text and graphic tasks simultaneously. The second interpretation is that people tend not to have the cognitive ability to perform text and graphic tasks at the same time. In either case, it is unlikely there would be much benefit from devoting one hand to each task. Then again, if the other hand would be idle the entire time, nothing would be lost from separating the devices. In the best case, performance could be improved. In the worst case, the problem of switching between text and graphic input would be solved.

The experiments used to measure the performances of the Biofeedback Pointer and Chording Glove made the implicit assumption that computer input would be the primary task and that real-world interaction would be secondary or non-existent. It is likely that the performance of the devices would be different if they were used for secondary tasks, with the primary task involving a real-world interaction. Siegal & Bauer (1997) describe an experiment in which a wearable computer was used to assist with the primary task of aircraft maintenance. The input device was a board-mounted dial, which was worn on a belt and operated with one hand. It was found that a one-handed system was more efficient, since certain tasks required the users to lean over objects, bracing themselves with one hand while operating the computer with the other. The body-mounted nature of the device did have disadvantages since leaning against the plane could accidentally activate the device. This demonstrates the advantage of an arm-based system such as the ones described here, which cannot be accidently activated in such a manner. In addition this also illustrates the need to easily deactivate and reactivate the input devices, possibly using the method described above.

Experimentation needs to be done to determine the performance of a wearable computer using the Biofeedback Pointer and Chording Glove. This not only needs to measure the performance of the input devices (input speed, error rate, IP, etc.), but the performance of the primary real-world task as well. This experiment should be able to determine if there is any degradation in the performance of either task, and to what extent it occurs. Furthermore, the experiment should also provide some insight into how the devices can be changed to be more efficient.

Terminal Typing Speed

For some tasks, text entry speed is the most important performance factor for the text input device. If a text input device is used solely for entering previously composed text or data, with little or no interaction, the time to complete the job is dependent only on how fast the user types. This is what the typewriter was designed for. This used to be the most common use for a text input device. Most modern text input tasks require interaction, such as reading email or searching for a file. A user only enters a minimal amount of text, followed by a great majority of time spent viewing the output. Speed is no longer the most important factor in text entry.

Take for example, the one of the most text-entry intensive task on a computer: composing a document on a word processor. In this, the author spends time considering what to write. This generally means text entry in short bursts, with significant pauses in between. This gives an effective speed limit, which could be far below the maximum speed for the device. This speed limit is the point where the ability to enter text faster will not be any more productive.

Assume a decent sized paragraph has around 100 words. If the entire paragraph is conceived of at once, a 60wpm typist could enter it in 100 seconds, while a 30wpm typist would take 200 seconds. Is that 100 seconds significant? How long will it take to conceive the next paragraph? What about error correction or reconsidering the words while typing? If the fast typist takes six minutes to write the paragraph, will the slower one take six minutes as well, or seven and a half? These are all important considerations. A speed of 60wpm is more difficult to maintain that 30wpm. Is there anything to be gained with such speeds in casual use? Ultimately, is the "terminal velocity" low enough to be significant (20–40wpm) or is it excessively high (50wpm+). This is a potentially important factor determining the appropriate application for a text input device.

Other Uses for the Biofeedback Pointer

Currently the Biofeedback Pointer has only been used for control of a 2D pointer in cartesian coordinates. It is possible to adapt the software to use another coordinate system. For example, one can use supination and pronation of the wrist instead of adduction and abduction. This lends itself quite well to polar coordinates. This would not be an ideal input for a windows-type system, partly because it singles out the origin as a special point, and partly because the wrist cannot move into the full 360° circle it would need to reach all the points on the screen. This may however be a useful interface for a navigation system which required turning left or right, and acceleration or deceleration. Because the muscles are in such close proximity, the electrodes can be left in the same position as they are for adduction and abduction. This would allow the user to be able to switch between the two interaction methods in software. There is precedent for this. Some existing applications use the mouse in non-standard ways, such as for navigation in 3D environments, so this is a reasonable option.

In Section 6.2.2 we suggested that the Biofeedback Pointer can be used on the neck muscles to control a pointer. It may be possible to further adapt the system to yield the orientation of the head to provide a head tracking system for virtual reality. Most head tracking systems work on broadcast technology, limiting the range to a specific room, or even a small part of a room. Using the EMG to determine the head's orientation removes the range constraints.

Another potential adaptation of the Biofeedback Pointer would be to use it for gesture recognition. A simple neural network, similar to the one currently used, could be trained to recognise a small number of gestures to perform various actions. In Section 6.2.2 we suggested the possibility of using gestures for selection instead of a hardware button. One possible gesture would be a quick flick of the thumb. This could provide the equivalent of a single or double click. In Section 6.2.2 we also suggested a gesture-based method for switching between text and graphic input.

As mentioned above, the coordinate system can be switched in software. There is no reason that the Biofeedback Pointer cannot be switched as needed from graphic to gesture input as well. In certain situations a set of arrow and function keys is faster than pointer input (Greenstein & Arnaut, 1988). In environments where interaction is constrained to a few responses, it may simplify matters to use a few easily recognised gestures. For example, in filling out a checklist, one might just need gestures for "yes", "no", "forward", and "back". Alternatively, gestures can be recognised as application specific macros or function keys.

Another use of the Biofeedback Pointer could be to replace a DataGlove-style virtual reality interface. Placing a 6-degree of freedom motion tracker on the arm would yield its position and orientation. The Biofeedback Pointer can then be trained to yield the orientation of the fingers and wrist, in a manner similar to Hiraiwa et al. (1993). This would give a hand-based VR interaction tool, without needing to wear a DataGlove or hold a 3D mouse.

Most of these options can be done by changes in software, not changes in electrode positioning or hardware. So the same device can be used as a 2d pointer, gesture recognition system, and (with the addition of a motion tracker) a VR interface.

6.3 Summary

There were a several potential problems with the implementation of both experiments. An analysis of the errors in the Chording Glove experiment showed that the experimentally determined performance is likely to be worse than the true performance. Thus the results are, in fact, a lower bound for the Chording Glove's performance. The performance test carried out on the Biofeedback Pointer was intended to give a range for the expected performance of the device. The possible problems with the tests were insufficient to cause a significant difference in the results. Consequently, like the Chording Glove, the errors could only serve to lower the performance, thus the Biofeedback Pointer's results also provide a lower bound for the true performance. Future experimentation and research on these devices needs to be done. The Chording Glove needs further research on keymap learning rates and retention, and the use of function keys. Research on the Biofeedback Pointer needs to be done on the selection buttons and the use of alternative muscle groups.

Several improvements for both devices have been suggested in this chapter. The Chording Glove would benefit from improved finger sensors, specialised software interaction, and possible "soft" function keys. The performance of the Biofeedback Pointer could be improved by exploring possible alternatives to the current motion scalings, training methods, and neural networks. Specialised hardware could be used to decrease the size, improve the speed, and lessen the drain on computer resources.

In this chapter we have suggested areas for input device research for the Chording Glove, Biofeedback Pointer and a wearable system using the two. From this we have made a few recommendations. The Chording Glove and the Biofeedback Pointer are designed to work together, but there are potential problems with switching between text and graphic input, and possible jitter from selection that need to be looked into. Research should be done to find out the true requirements of input speed for casual text entry to determine what upper limits we need from text input devices. The Biofeedback Pointer hardware and software is very flexible and it is worthwhile investigating its potential use for 3D interaction and gesture recognition.

Chapter 7

Conclusions

7.1 Introduction

The graphic and text input devices currently in use by mobile computers were not *designed* for portability, but were more *re*-designed for portability. These devices were often originally intended for desktop systems and then shrunk down to fit a smaller size of computer. Instead of using modified input devices, mobile computers need input devices specifically designed for them: devices that are as unobtrusive as possible in order to free the computer to take a more efficient shape.

The wide variety of characters, editing commands and other functions needed for text entry makes text input devices much more complex and, consequently, much less portable than graphic inputs. Interfaces like the keyboard are quite efficient on a desktop, but become rather difficult to use when shrunk for use in a mobile environment. In order to minimise the restrictions on the portability of a text input device, it is necessary to find a new approach which does not use board-based interfaces. Graphic input devices tend to be more portable than text input devices because the interaction is simpler. Pointing devices currently exist which are completely unobtrusive and portable. These are usually based on tracking some part of the body and are often expensive, imprecise, or have a limited range, precluding their use on a mass scale. In order to make these devices more feasible, it is necessary to use new technologies to overcome their deficiencies and thereby make new input devices which are designed to be not only portable, but are easy to operate, efficient, and acceptable enough for widespread use.

7.2 Review of Design Goals

A set of four primary goals were developed to provide direction for the design process by outlining the requirements of the final devices. These design goals were as follows:

1. The devices must be operable in a mobile environment. They cannot depend on any external surfaces or objects.
2. The devices must be usable by both nomadic and continuous users. They must not hinder the user in performing real-world tasks.
3. The devices must be able to operate a standard windows-style interface and to perform a wide range of simple text entry tasks.
4. The devices must be able to be used with existing wearable computer systems by using standard communications ports and requiring reasonable levels of computing power.

The Biofeedback Pointer is very well adapted for mobility. No aspect of the device hinders movement nor requires any external devices or surfaces for operation. The Chording Glove has the potential for high mobility as well. The only constraint on its portability is the need for a solid surface to chord against. It is unknown what the limits imposed by this constraint are. The Chording Glove has been experimentally shown to work in a simple mobile environment, but further tests are necessary to determine the effectiveness in more demanding situations.

The devices' usefulness in a mobile environment makes them well suited for the continuous mobile computer user. Both devices are operated by one hand and require no visual supervision. The heads-up use allows the user to keep track of the surrounding environment and the one unused hand allows the user to freely interact with it while computing. The user can stop computing at a moment's notice and have full, unhindered interaction with the real-world, should the need arise. In addition to the mobile computer user, these devices are easily adaptable for use by disabled persons as well. Chording devices like the Chording Glove have been used by people with motor disabilities. The wide range of muscle groups which can be used by the Biofeedback Pointer makes it particularly suited for users who are partially paralysed.

The Biofeedback Pointer is best suited for tasks involving navigation and low precision pointing. This encompasses the standard windows interface as well as basic exploration of two and three dimensional data. The Chording Glove, while slower than a desktop keyboard, is not much slower than existing mobile devices such as a stylus or soft keyboard. With further work, the Chording Glove's input speed may even surpass them by reaching the theoretical input speed of around 40wpm. This implies that the Chording Glove is fast enough for the casual levels of use of existing mobile devices, but further research must be done to determine exactly what range of speeds "casual use" covers.

The Chording Glove can be easily run on commonly existing systems. There is very little computation or memory necessary to run the device, making it easily usable on even the simplest computers. Communication between the prototype and the computer is carried out though the parallel port, but his could be changed to a serial or even a keyboard port if necessary. The Biofeedback Pointer is more complicated, requiring at least a Pentium 133MHz to run. The prototype did not use any specialised hardware, but future versions might. Moving the data sampling and FFT to a specialised device would take a significant fraction of the computing power away from the computer, allowing the pointer to be used on a simpler computer. Data acquisition for the Biofeedback Pointer was performed by an ADC attached to the parallel port. ADCs also exist which can be attached to a serial port or via a PCMCIA card.

The existing prototypes of the Biofeedback Pointer and Chording Glove have been shown to satisfy most of the basic design goals. There are a few remaining goals which were not directly shown to be satisfied. In some cases the goals were implied to be satisfied by research done on similar devices. In the other cases, the goal was shown to be partly satisfied by experiments which did not test sufficient range to know if the entire goal was reached.

7.3 Summary of Contributions

7.3.1 Review of Input Devices and Mobile Computers

The discussion on text and graphic input devices was intended to provide an overview of the current options for input devices. We were specifically interested in their performance, portability, and ergonomics. By considering these for each type of device we were able to generalise the requirements for graphic and text input devices, and how to make such devices work well in a mobile environment. This was essential in order to efficiently design new input devices.

The analysis of mobile computers discussed the range of technology currently being used and the likely directions the technology will take in the next few years. By examining the state of the art of mobile computers we were able to determine how both the style of the computer and the input devices used effect the portability of the system. From this we were able to conclude that a boardless input device is the most efficient for a wearable computer. This conclusion, plus the design requirements discovered earlier, were used to design new input device which could fully complement the mobile nature of wearable computers.

7.3.2 The Chording Glove

The Chording Glove was designed and built using the design criteria generated from the review of devices and systems. This text input device is worn as a glove, making it unnoticeable when not being used, but it is always there, ready for use, whenever the need should arise.

The device's performance was experimentally tested. After training for 11 hours, users were able to input text at a rate of 16.8wpm, with 17% of the characters made by mistake. This was somewhat worse than expected, or desirable, and was due to inaccurate finger sensors. These need to be replaced in future versions of the device. Consequently the performance of the Chording Glove in this thesis should be used as a lower limit for its potential performance. The Chording Glove was also shown to be usable without loss of performance in a simplified mobile environment. However, further research must be done to determine exactly what range of mobile environments in which the Chording Glove will function.

The chord keymap must be learned before the device can be used. It took users, on the average, 80 minutes to complete a tutorial, in which they learned to chord all 97 characters. After 6 hours of use, users needed to look up forgotten chords around 0.5% of the time. There is evidence that the keymap is remembered after significant periods of non-use, and that the previous performance is quickly re-attained, but this needs to verified with further research.

In creating the Chording Glove a generalised method for developing a chord keymap was designed. In this method, the characters are first grouped into logical groups, such as letters, numbers, and punctuation. These are used for the different shift states of the device. The characters are then assigned to the chords, with frequent characters aligned with the easiest chords. Once a basic keymap is made, it is altered in five steps:

1. Assign shifted chords to provide a logical relationship between them and their unshifted chords (e.g. `P` and `+`)

2. Revise so that finger positions for the chords bear a resemblance to the characters they make.

3. Revise so that frequent digrams and trigrams are made by chord sequences which are easy to perform.

4. Revise so that characters which look similar are made by chords which look similar (e.g. `U` $\rightarrow$ `W` = $\circ^{\bullet\,\circ\,\circ\bullet} \rightarrow \bullet^{\bullet\,\circ\,\circ\bullet}$)

5. Clean up the keymap to fix any poorly assigned characters.

This method for creating a chord keymap can be extended to most styles of chord keyboards. Alternatively it can be used to create new keymaps for the Chording Glove, either for use with foreign languages, specialised applications, or personal customisation.

It is easy for a user to customise a standard keyboard by using software to change the layout. However, this is rarely done in practice because it tends to create more problems than it solves. The most obvious problem is that the symbols on the keys conflict with their function. Either the user changes the key labels every time they alter the layout, or they need to memorise the layout and never look at the keys. Another problem is when the user attempts to operate someone else's keyboard. The interference between the two layouts lowers the user's performance. The Chording Glove does not suffer from either of these problems. When the user cannot remember a chord, they can use the `<Help>` key to display the keymap on the screen, whichever keymap is used. Since the Chording Glove is portable, the user can take their own customised interface with them, avoiding the need to adapt to each new system they come across.

A theoretical method for testing chord keymaps was developed along with the Chording Glove. These tests use the relative probability for each character along with Seibel's (1962) DRTs and error rates for all 31 chords. The first test is the normalised finger work. Strong fingers like the thumb and index finger should be used most, and weak fingers like the little finger should be used least. The work for each finger is found by the sum of all probabilities of the characters made by chords using that finger. Each result is normalised by dividing by the sum of the results for all fingers.

The second test and third tests determine the theoretical speed and error rate of the device. Seibel measured the DRT and error rate for each chord. The DRT is the time it takes to make each chord and the error rate is the probability of entering the wrong chord. The theoretical speed is determined by totalling the product of each chord's DRT and its probability. The theoretical error rate is equal to the sum of each chord's probability multiplied by its percent error. In other words, for chord c with probability $p(c)$, digram time DRT(c) and error probability of $\epsilon(c)$:

$$\text{time per character} = \sum_c p(c) \cdot \text{DRT}(c) \qquad (7.1)$$

$$\text{error rate} = \sum_c p(c) \cdot \epsilon(c) \qquad (7.2)$$

Equation 7.1 can be converted into words per minute by Equation 4.1.

7.3.3 The Biofeedback Pointer

In designing the Biofeedback Pointer, the benefits of using the EMG, EEG, and GSR were considered. The EMG was used because the signal was the most consistent and the easiest to measure. Hiraiwa et al.'s (1993) method for EMG analysis determined to be the most promising and was adapted to work with the Biofeedback Pointer. This neural network was simplified to produce only an x and y value instead of the angles of all the finger joints. These two values could be generated by any two independent motions, making the system usable with just about any muscle group. The Biofeedback Pointer's simplified network was less computationally expensive, allowing a higher refresh rate. However, the side effect was a reduction in accuracy. This was compensated for by using more EMG sensors. Instead of using special hardware to train the device, the training was performed by requiring the user to follow the pointer's motion on the screen. This may not be quite as accurate as using a motion sensor, but it enables the Biofeedback Pointer to be used with any standard PC without requiring specialised hardware.

The Biofeedback Pointer succeeds very well in its intent to be truly portable. Electrodes and amplifiers are very small and can be worn underneath clothing. The box containing the hardware is around the size of a cassette case and can be clipped to the belt. If a long sleeve shirt is worn over the electrodes and wires, the device is completely invisible. Even when it is used it cannot be seen.

A user can train the Biofeedback Pointer in about half a minute. This is fast enough that, in case of problems with the neural network, retraining is not a nuisance. As the user starts using the pointer, they also start learning how to manipulate the device to fine tune their control using biofeedback.

The mouse and Biofeedback Pointer were experimentally compared using Fitts' Law to measure the performance of each device. The index of performance of the Biofeedback Pointer was 1.06, and the mouse was 7.1. This gives a ratio of the Biofeedback Pointer to the mouse of 0.14. Existing mobile pointing devices have IP ratios ranging from 0.26 to 1.09. The Biofeedback Pointer is just over half the lowest performance in that range. Future enhancements, such as a more sophisticated neural network or better training methods will be useful in closing this gap.

7.4 Conclusions

The size of mobile computers has always been limited by the input devices. Notebook computers have been limited in size by the standard keyboard, which quickly loses its efficiency when shrunk. Usable handwriting recognition systems were a significant factor in the current boom in handheld computers. The mobile computer is now limited to a new, smaller size, but the shape is still limited to a notepad-style interface. The next step in mobile computer evolution is to make an interface which frees the computer from the constraints of the input devices to allow an entirely wearable system. The devices described in this thesis are wearable computer input devices which put no constraints on the size of the system. They work in the same manner with a desktop computer as they would with one the size of a matchbook. The Chording Glove and Biofeedback Pointer meet all of the design requirements set out in this thesis, but work still needs to be done to improve their performance. It is recommended that further research be done to enable these devices to achieve their potential.

Appendix A

Glossary

A.1 Special Symbols

Mathematical Symbols

The following mathematical variable types are used in this thesis:

m	A matrix (also: $m_{i,j}$ or $\mathbf{m}_i$)
$\mathbf{v}$	A vector or array (also v_i)
$\hat{\mathbf{n}}$	A normalised vector
s	A scalar

Other Symbols

The names of keys or functions on a text input device are represented in a `monospace` type in angled brackets, such as: `<Space>`. Individual characters are represented by the symbol printed in a `monospace` type, such as: `a`.

A special graphical notation is used to describe the finger patterns used to create a chord on the Chording Glove. The diagram uses filled or unfilled circles positioned in the approximate positions of the fingertips on the right hand. The thumb is lowered and on the left, and the other fingers are raised, with the order: index, ring, middle, and little. A filled circle means the finger is pressed, and an empty circle indicates that the finger is not. For example, the chord $_{\circ}{}^{\circ}\,\bullet\,{}^{\bullet\circ}$ is generated by pressing the middle and ring fingers.

A.2 Glossary of Physiological Terms

The *anatomical position* is the standard orientation of the body from which all directions and motions are defined. In this position, the person is standing, with the arms down, feet pointing forward, and the thumbs pointing outward so that the palms face forward. Palastanga et al. (1990), pages 2–7 provides a good summary of the basic anatomical terminology.

The following physiological terms are used frequently in this thesis:

abduction A movement in the *lateral* direction, i.e. outward from the centre of the body. The opposite of *adduction*.

adduction A movement in the *medial* direction, i.e. towards the centre of the body. The opposite of *abduction*.

anterior In front of, or towards the front. For example, the nose is on the *anterior* part of the face. Also called *ventral*. The opposite of *posterior*.

deep The direction inward from the surface of the skin. The opposite of *superficial*.

distal The direction away from the root of a limb. The opposite of *proximal*.

dorsal Behind, or towards the rear. Also called *posterior*. The opposite of *ventral*.

extension Returning a joint to the neutral position, or moving beyond the neural position in the opposite direction of *flexion*.

flexion Bending a joint so that two anterior or posterior faces are brought together. The opposite of *extension.*

lateral The direction outward from *median plane*, ie out from the centre of the body. The opposite of *medial.*

medial The direction toward the *median plane*, ie towards the centre of the body. The opposite of *lateral.*

median plane The plane of symmetry between right and left halves of the body. Also called the *sagittal plane.*

posterior Behind, or towards the rear. For example, the spine is on the *posterior* part of the trunk. Also called *dorsal.* The opposite of *anterior.*

pronation Rotating the forearm along the axis of the arm, so that the palm faces backwards. The opposite of *supination.*

proximal The direction toward from the root of a limb. The opposite of *distal.*

radial deviation *Abduction* of the wrist, ie bending the wrist in the direction of the thumb. The opposite of *ulnar deviation.*

sagittal plane The plane of symmetry between right and left halves of the body. Also called the *median plane.*

superficial The direction toward the surface of the skin. The opposite of *deep.*

supination Rotating the forearm along the axis of the arm, so that the palm faces forward. The opposite of *pronation.*

ulnar deviation *Adduction* of the wrist, ie bending the wrist in the direction of the little finger. The opposite of *radial deviation.*

ventral In front of, or towards the front. Also called *anterior.*

Appendix B

Relative Character Frequency for Keymap Design

Character probabilities were calculated by counting the appearances of each of the ASCII characters found in a large amount (596648 characters) of text in various formats which were considered to be likely to be the sort of text which is entered by hand (as opposed to computer generated). The breakdown of text types is as follows: (from most used to least).

1. Informal text (email, internet news posts, etc)
2. Formal text (papers, memos, etc)
3. C code
4. Theatre scripts
5. Recipes
6. Bibliographies
7. X Resources
8. Telephone numbers
9. Street Addresses

The following tables show the probabilities of each characters. `<Space>` and `<Return>` were omitted from the lists due to their high relative frequencies.

Table B.1: Character probability sorted by ASCII code

% Probability	**Char.**	**% Probability**	**Char.**	**% Probability**	**Char.**
0.835	tab	0.511	A	5.916	a
0.104	!	0.269	B	1.367	b
0.238	”	0.499	C	2.351	c
0.068	#	0.360	D	2.823	d
0.021	$	0.435	E	9.436	e
0.054	%	0.249	F	1.739	f
0.057	&	0.180	G	1.863	g
0.364	’	0.215	H	3.484	h
0.607	(	0.574	I	5.438	i
0.609	)	0.080	J	0.123	j
0.709	*	0.129	K	0.859	k
0.131	+	0.448	L	3.273	l
1.104	,	0.297	M	2.035	m
1.224	-	0.343	N	5.569	n
1.830	.	0.309	O	6.564	o
0.167	/	0.269	P	2.043	p
0.546	0	0.048	Q	0.090	q
0.441	1	0.357	R	5.545	r
0.285	2	0.552	S	4.688	s
0.223	3	0.612	T	6.736	t
0.148	4	0.178	U	2.335	u
0.161	5	0.110	V	0.827	v
0.138	6	0.190	W	1.168	w
0.119	7	0.353	X	0.451	x
0.116	8	0.112	Y	1.491	y
0.138	9	0.009	Z	0.136	z
0.703	:	0.142	[	0.067	{
0.399	;	0.131	\	0.033	|
0.108	<	0.142	]	0.067	}
0.957	=	0.001	^	0.012	~
0.189	>	0.456	_		
0.084	?	0.008	‘		
0.035	@				

Table B.2: Character probability in order of decreasing probability

% Probability	Char.	% Probability	Char.	% Probability	Char.
9.446	e	0.552	S	0.138	9
6.736	t	0.546	0	0.138	6
6.564	o	0.511	A	0.136	z
5.916	a	0.499	C	0.131	\
5.569	n	0.454	_	0.131	+
5.545	r	0.451	x	0.129	K
5.438	i	0.448	L	0.123	j
4.688	s	0.441	1	0.119	7
3.484	h	0.435	E	0.116	8
3.273	l	0.397	;	0.112	Y
2.823	d	0.364	’	0.110	V
2.351	c	0.360	D	0.106	<
2.335	u	0.357	R	0.104	!
2.043	p	0.353	X	0.090	q
2.035	m	0.343	N	0.084	?
1.863	g	0.309	O	0.080	J
1.830	.	0.297	M	0.068	#
1.739	f	0.285	2	0.067	{
1.491	y	0.269	B	0.067	}
1.367	b	0.262	P	0.057	&
1.224	-	0.249	F	0.054	%
1.168	w	0.238	”	0.048	Q
1.104	,	0.223	3	0.035	@
0.957	=	0.215	H	0.033	|
0.859	k	0.190	W	0.021	$
0.836	tab	0.189	>	0.012	~
0.827	v	0.180	G	0.009	Z
0.709	*	0.178	U	0.008	‘
0.703	:	0.167	/	0.001	^
0.612	T	0.161	5		
0.609	)	0.148	4		
0.607	(	0.142	]		
0.574	I	0.142	[		

Table B.3: Character probability by type

Lower case letters (default)		**Upper case letters** (CAPS)		**Numerical characters** (NUM)		**Punctuation** (NUM + CAPS)		**Control characters** (CTRL)	
	Return		Return		Return		Return		
	Space		Space		Space		Space		
	Backspace		Backspace		Backspace		Backspace		
9.44	e	1.83	.	1.83	.	0.71	*	0.84	tab
6.74	t	1.10	,	1.22	-	0.70	:		
6.56	o	0.61	T	1.10	,	0.45	_		
5.92	a	0.57	I	0.96	=	0.40	;		
5.57	n	0.55	S	0.71	*	0.36	’		
5.55	r	0.51	A	0.61	)	0.24	”		
5.44	i	0.50	C	0.61	(	0.19	>		
4.69	s	0.45	L	0.55	0	0.13	\		
3.48	h	0.43	E	0.44	1	0.11	<		
3.27	l	0.36	D	0.29	2	0.10	!		
2.82	d	0.36	R	0.22	3	0.08	?		
2.35	c	0.35	X	0.17	/	0.07	#		
2.34	u	0.34	N	0.16	5	0.07	{		
2.04	p	0.31	O	0.15	4	0.07	}		
2.04	m	0.30	M	0.14	]	0.06	&		
1.86	g	0.27	B	0.14	[	0.05	%		
1.83	.	0.26	P	0.14	9	0.04	@		
1.74	f	0.25	F	0.14	6	0.03	|		
1.49	y	0.21	H	0.13	+	0.02	$		
1.37	b	0.19	W	0.12	7	0.01	~		
1.17	w	0.18	G	0.12	8	0.01	‘		
1.10	,	0.18	U	0.001	^	0.001	^		
0.86	k	0.13	K						
0.83	v	0.11	Y						
0.45	x	0.11	V						
0.14	z	0.08	J						
0.12	j	0.05	Q						
0.09	q	0.01	Z						
31 characters		**31 characters**		**25 characters**		**25 characters**		**1 character**	

Appendix C

The Chording Glove Hardware

C.1 Introduction

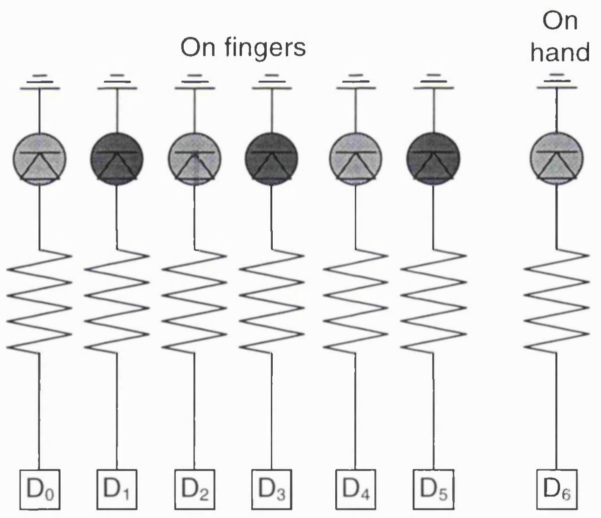
This appendix describes the sensors and indicators used in the Chording Glove. The design and circuitry are described in detail in order to facilitate experimentation by other investigators. In addition, sensor designs that were tested, but not used in the final version are included here so future researchers will know that the designs are insufficient.

C.2 LED Outputs

The LEDs are used to show the current shift state. The LEDs are all independent in the hardware and can be turned on and off in any manner. This could be additionally used to alert the user to some error. To save space, 2.21mm square sub-miniature LEDs were used. These had an intensity of about 3mcd, which is bright enough to be easily noticed. The circuit diagram is shown in Figure C.1.

There are two groups of LEDs. Six on the index finger and one on the back of the hand. The index finger LEDs are divided into 3 groups of two LEDs, one red and one green. Only one of a group (i.e. red or green) should be on at a time. The green indicates the shift is on and will operate on the next character. The red indicates that the shift is locked and will stay on until turned off. These LEDs are located next to the shift buttons (Figure 3.1). The LED on the back of the hand is near the AutoCaps key. When AutoCaps is on the LED is on, when off the LED is off. The hardware design leaves room for future use of an eighth LED.

Figure C.1: Circuit diagram of the LED setup. D_n is the nth output bit. Resistor values are 180Ω.



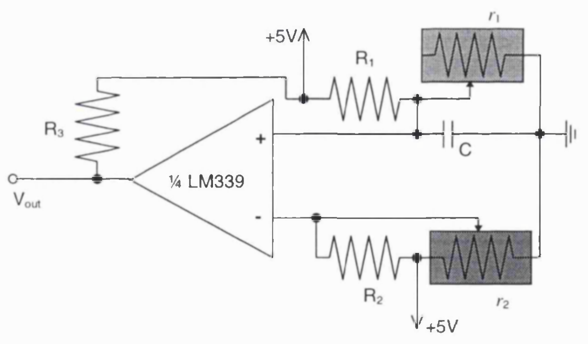
C.3 Finger Sensors

Ten designs were tested for the finger sensors of the Chording Glove. These sensors were either digital or analogue. All the analogue sensors had the same circuit diagram, as shown in Figure C.2(a). In this diagram, r_1 is the sensor, with a variable resistance and r_2 is a potentiometer which sets the cutoff voltage for the sensor. These are fed into a comparator to yield a digital on/off reading for the sensor. The capacitor is used for a low pass filter to remove excess noise. The resistor R_3 is used to pull up the comparator output. The digital sensors has the same circuit diagram which is shown in Figure C.2(b). An open circuit gives $v_{\text{out}} = 5$V and a closed circuit grounds v_{out}. The capacitor is used for a low pass filter to eliminate bounce noise.

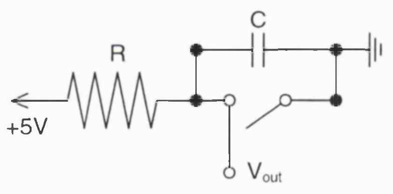
The ten sensor designs were as follows:

1. Double foil plates.
2. Interlaced foam.
3. Foil plate and single laced foam.
4. Single laced foam and conductive unlaced foam with foil backing.
5. Single laced foam and foil cover.
6. Moulded resistive foam with foil backing.
7. Moulded low density resistive foam with foil backing.
8. Resistive foam with silvered faces.
9. Resistive foam with foil backing.
10. Hard copper plate.

The Resistive Foam with Foil Backing sensors were initially used in the experiment, but were soon changed after they were found to be long-term unstable. They were replaced by the Hard Copper Plate sensors which were successfully used for the remainder of the experiments.



(a) The analogue finger sensor. C=22pF, R$_1$=1kΩ, R$_2$=430kΩ, R$_3$=100kΩ, r_1 is the variable resistance sensor, and r_2 is a $20k\Omega$ potentiometer



(b) The digital finger sensor. C=22pF and R=56kΩ

Figure C.2: Circuit diagrams for the finger sensors

C.3.1 Double Foil Plates

Description The sensor consists of two aluminium foil plates separated by foam on the edges. The centre of each plate bulges out due to the placement of the wires, which are taped onto the foil (Figure C.3).

Operation Pressing with a finger causes the bump in the middle of one plate to hit the bump in the middle of the other plate, completing the circuit.

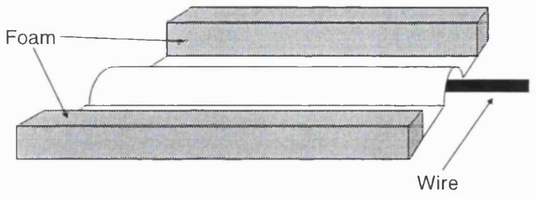


Figure C.3: Double Foil Plate Finger Sensor. For clarity, the top plate is not shown in this diagram.

Advantages Very easy to make.

Problems

- The sensor does not have necessary accuracy. It only works on well-defined presses near the middle. It is not sensitive near the edges.
- It is too small to sense the entire fingertip, and too big to use multiple sensors on one finger.

C.3.2 Interlaced Foam

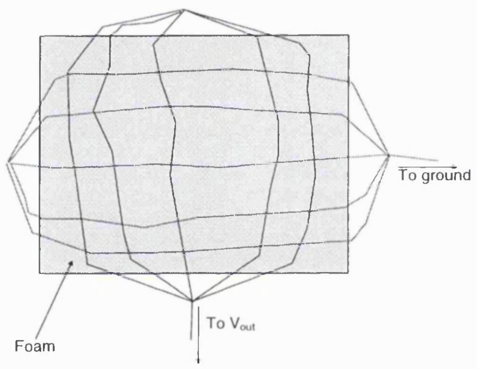


Figure C.4: Interlaced Foam Finger Sensor

Description The sensor is made from a rectangular piece of foam about the size of the fingertip. Several small, flexible threads of wire are separated and sewn though the foam. One group of wires are sewn horizontally on one side of the foam, the others are sewn vertically on the other side. Each group meets at the end of the foam. Each groups meets outside the foam and are bundled together to make one wire (Figure C.4).

Operation Pressing anywhere on the foam with a finger causes the wires to touch completing the circuit.

Advantages

- Relatively easy to make.
- Should be accurate over the entire surface, as compared to the first sensor attempted.

Problems

- It is easy to make, but in practice it is hard to standardise. Threads do not go through entirely straight. This and irregularities (holes or bunching up) in the foam causes areas of extra sensitivity or no sensitivity.

- It is easy for wires to touch accidentally. If this happens it is very hard to fix.
- It is not flexible enough to mould to the shape of a fingertip.

C.3.3 Foil Plate and Single Laced Foam

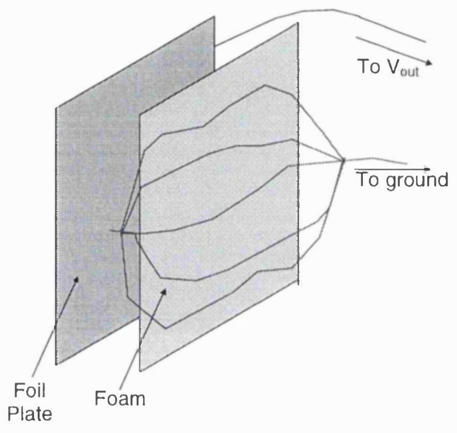


Figure C.5: Foil Plate and Single Laced Foam Finger Sensor

Description The sensor is made from two parts. The main part is a rectangular piece of foam about the size of the fingertip. The wire connected to it is divided into several small flexible threads. These threads are sewn through the foam, lengthwise, very close to the surface. The other wire is connected to the back of a strip of aluminium foil the same size as the foam. The foam is taped to the front of the foil (Figure C.5).

Operation Pressing anywhere on the foam with a finger causes the wires in the foam to touch the foil, completing the circuit.

Advantages

- Relatively easy to make
- There are no insensitive areas on the sensor.

Problems

- In practice it is hard to standardise. Irregularities (holes or bunching up) in the foam causes areas of extra sensitivity or no sensitivity.
- The sensor not flexible enough to mould to the shape of a fingertip.

C.3.4 Single Laced Foam and Conductive Unlaced Foam with Foil Backing

Description Two aluminium foil plates are separated by a high density resistive foam, and a non-conductive foam, laced with wires (Figure C.6).

Operation When the sensor is pressed, the wires touch the conductive foam, completing the circuit.

Advantages None.

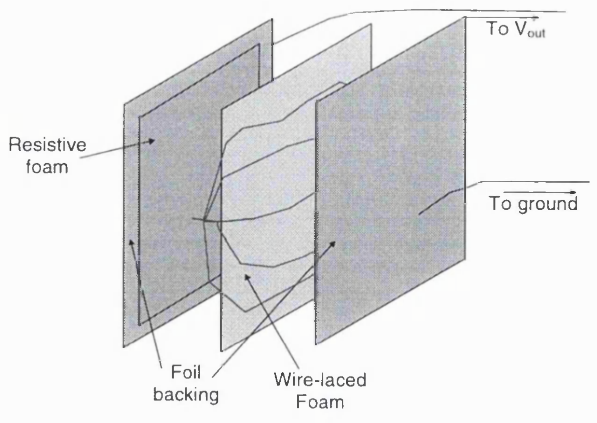


Figure C.6: Single Laced Foam and Conductive Unlaced Foam with Foil Backing Finger Sensor

Problems The wires in the foam cannot be made close enough to the surface to reach the foam when compressed . As a consequence the design cannot work.

C.3.5 Single Laced Foam and Foil Cover Finger Sensor

Description The first half of the sensor is a rectangular piece of foam about $1^1/_2$ inches by 1 inch. The wire connected to it is divided into several small flexible threads. These threads are sewn through the foam, lengthwise, just below the surface. This is sewn onto the glove covering the fingertip. The other half of the sensor is foil, moulded to the shape of the fingertip and placed over the foam. A second wire is taped to the foil (Figure C.7).

Operation Pressing with a finger causes the foil to touch the wires in the foam, which completes the circuit.

Advantages

- This is the first design shaped specifically for the fingertip.
- There are no insensitive areas on the fingertip.

Problems

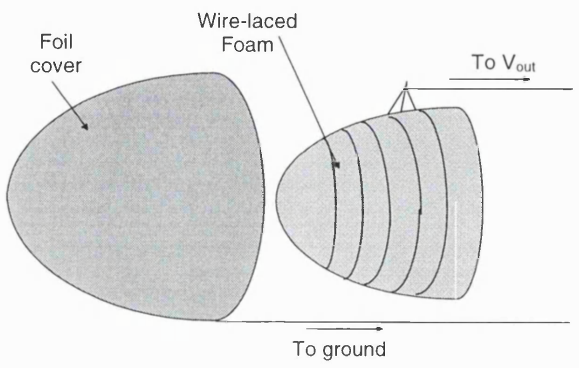


Figure C.7: Single Laced Foam and Foil Cover Finger Sensor

- It covers the entire fingertip, therefore it does not leave room for any shift keys on the last knuckle of the index finger
- Irregularities in the size of the gaps in the foam cause too many unwanted circuit connections.

C.3.6 Moulded Resistive Foam with Foil Backing Finger Sensor

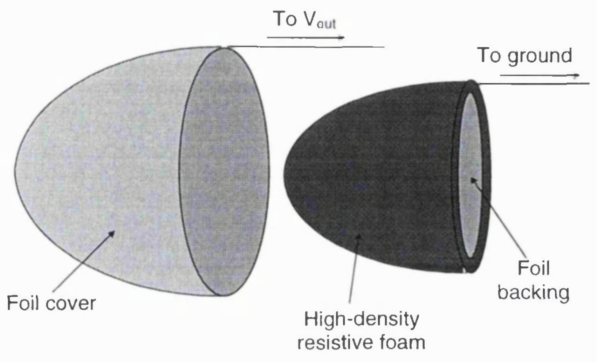


Figure C.8: Moulded Resistive Foam with Foil Backing Finger Sensor

Description The sensor is made from a resistive foam shaped to the shape of a fingertip. There are two aluminium foil plates, one on the inside and one on the outside. A wire is attached to each foil plate (Figure C.8).

Operation Pressing the sensor with a finger causes the foam to compress, lowering the resistance. This tried with both a Schmitt Trigger (with fixed positive and negative voltages) and a double voltage divider (Figure C.2(a)). The latter was found to work better.

Advantages

- Easy to make
- Adjustable after making
- The sensor is not connected to the glove, making it easily fixed and interchangeable

Problems

- A large number of components are necessary to use the sensor
- Since the sensor covers the entire fingertip, finger motions can set it off.

C.3.7 Moulded Low Density Resistive Foam with Foil Backing

Description The sensor is made from a low-density resistive foam moulded to the shape of a fingertip. There are two aluminium foil plates, one on the inside and one on the outside. A wire is taped to each foil plate (Figure C.9).

Operation Pressing with a finger causes the foam to compress, lowering the resistance. The resistance is detected by a double voltage divider (Figure C.2(a))).

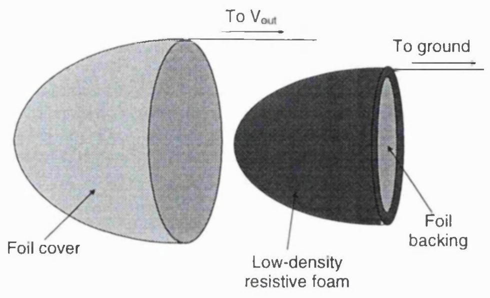


Figure C.9: Moulded Low Density Resistive Foam with Foil Backing Finger Sensor

Advantages

- Easy to make
- Adjustable after making.
- The sensor is not connected to the glove, so it is easily fixed and interchangeable.
- The sensor is more sensitive and less fragile than the high-density foam.

Problems

- The sensor requires more additional hardware than a digital sensor.
- Since the sensor covers the entire fingertip, finger motions can set it off.

C.3.8 Resistive Foam with Silvered Faces

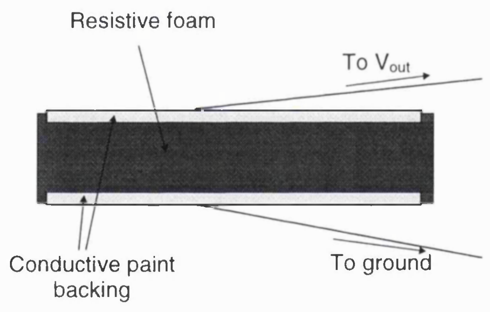


Figure C.10: Resistive Foam with Silvered Faces Finger Sensor

Description The sensor is made from a small rectangle of resistive foam which is coated with conductive paint on both sides. Wire is taped to each side. The foam is then sewn on the fingertip (Figure C.10).

Operation Pressing the sensor with a finger causes the foam to compress, lowering the resistance. This is detected by the change in voltage across a voltage divider.

Advantages

- Easy to make
- Adjustable after making
- Movements do not set it off

Problems

- The sensor requires more additional hardware than a digital sensor.
- Since the sensor is small some finger presses may go unnoticed.

C.3.9 Resistive Foam with Foil Backing

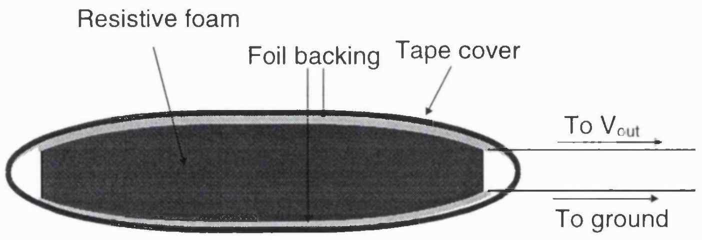


Figure C.11: Resistive Foam with Foil Backing Finger Sensor

Description A layer of foil is taped to the top and bottom of a small rectangle of resistive foam. Wire is taped to each side (Figure C.11).

Operation Pressing the sensor with a finger causes the foam to compress, lowering the resistance. This is detected by the change in voltage across a voltage divider.

Advantages

- Easy to make.
- Adjustable after making.
- Movements do not set it off.
- If wires are pulled out, they can be easily replaced.
- Weaker fingers can use thicker foam for more sensitivity.

Problems

- The sensors require a comparator and potentiometer in order to use.
- The wires are very easy to pull out.
- Since it is small, some finger presses may go unnoticed.
- The resistance in the foam is temperature dependent. The longer it is used, the more the resistance changes. The heating comes from a combination of internal heating due to the resistance and external heating from the body heat of the finger. As a result potentiometers which control the threshold level must be constantly changed during use. The resistance eventually goes beyond the range of the potentiometers, rendering the sensor useless.

This sensor was the one initially used in the experiments. The temperature dependency was only discovered after the first few sessions. After this was discovered it was replaced by the next sensor.

C.3.10 Hard Copper Plate

Description Two springs connect a pair of small rectangular plastic plates. Thin sheets of copper are glued to each of the plastic backing plates. Wires are soldered to the copper plates. Additional solder acts as a conductive extension of the copper plates. The entire sensor is encased in insulating tape to prevent damage and facilitate connecting to the glove (Figure C.12).

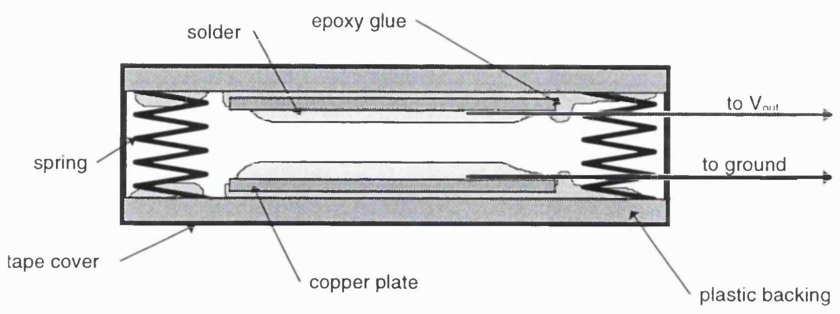


Figure C.12: Hard Copper Plate Finger Sensor

Operation Pressing the sensor with a finger causes the springs to compress and the copper solder on top of the copper plates to complete the circuit. Bounce noise is removed by a filter in the hardware Figure C.2(b).

Advantages

- Easy to make.
- The height of the solder surface can be adjusted slightly to alter sensitivity.
- Movements do not set it off.
- Wires are attached with epoxy and solder and should not be able to be disconnected short of major breakage.

Problems

- Extra capacitors are necessary to remove bounce noise.
- Flat sensors do not mould to the fingertip shape well causing stress on the thread connecting them to the glove.
- The components are large and bulky, making the sensor somewhat inconvenient to use.
- There is a layer of solder on top of the metal plate. When the solder is initially put on, some of the flux forms a thin layer on top. After repeated use, the flux fractures, forming what appears to be a white powder, at the contact points. This acts as an insulator and prevents contact between the plates, causing the sensor to become difficult or impossible to use. The excess flux can be easily removed by burning off with a soldering iron. The more flux removed, the less is left to cause a problem. This problem disappears after long term use.
- Repeated heating of the metal pads (to remove flux, repair dislodged wires, etc) can cause the plastic backing to deform.
- There is not enough room on the thumb for two sensors.
- These are the most expensive of all the attempts at finger sensors.

These sensors replaced the Resistive Foam with Foil Backing sensors in the experiment, and were found to function adequately.

C.4 Shift Buttons

Two designs were tested for the shift buttons:

1. Taped aluminium foil plates.
2. Taped copper foil plates.

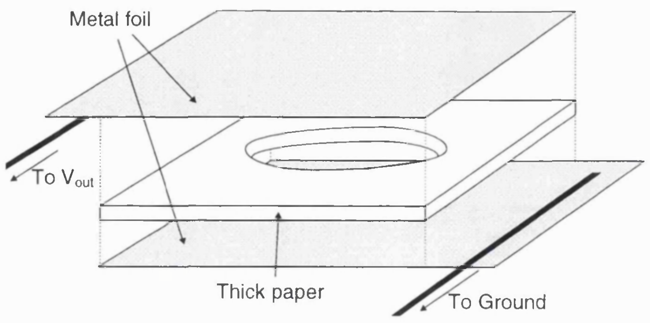


Figure C.13: Taped Aluminium Foil Plate Shift Button

C.4.1 Taped Aluminium Foil Plates

Description The button consists of two aluminium foil plates separated by a thick piece of paper with a hole cut out of the middle. The wires are connected to the foil at the sides. The button is taped together with blue PVC tape (Figure C.13).

Operation Pressing the button with a finger causes the foil plates to connect, completing the circuit.

Advantages

- Very easy to make
- Accurate

Problems

- No tactile feedback
- The buttons are too fragile to handle long term usage because the wires cannot be soldered to the aluminium, and consequently keep pulling loose.

C.4.2 Taped Copper Foil Plates

Description Two copper foil plates are separated by a thick piece of paper with a hole cut out of the middle. The wires are soldered to the foil at the sides. The button is covered with blue PVC tape. The setup is the same as for the Taped Aluminium Foil Plates (Figure C.13).

Operation Pressing the button with a finger causes the foil plates to connect, completing the circuit.

Advantages

- Very easy to make
- Accurate
- Wires will not pull out
- Stable enough for long term usage

Problems

- No tactile feedback

C.5 Function Keys

The function keys use the same Taped Copper Foil Plates buttons as the shift buttons (Figure C.13).

Appendix D

The Chording Glove Experiment Details

This appendix explains in detail various aspects of the Chording Glove experiment. These are: the questionnaires, the tutorial, and the chording sessions.

D.1 Questionnaires

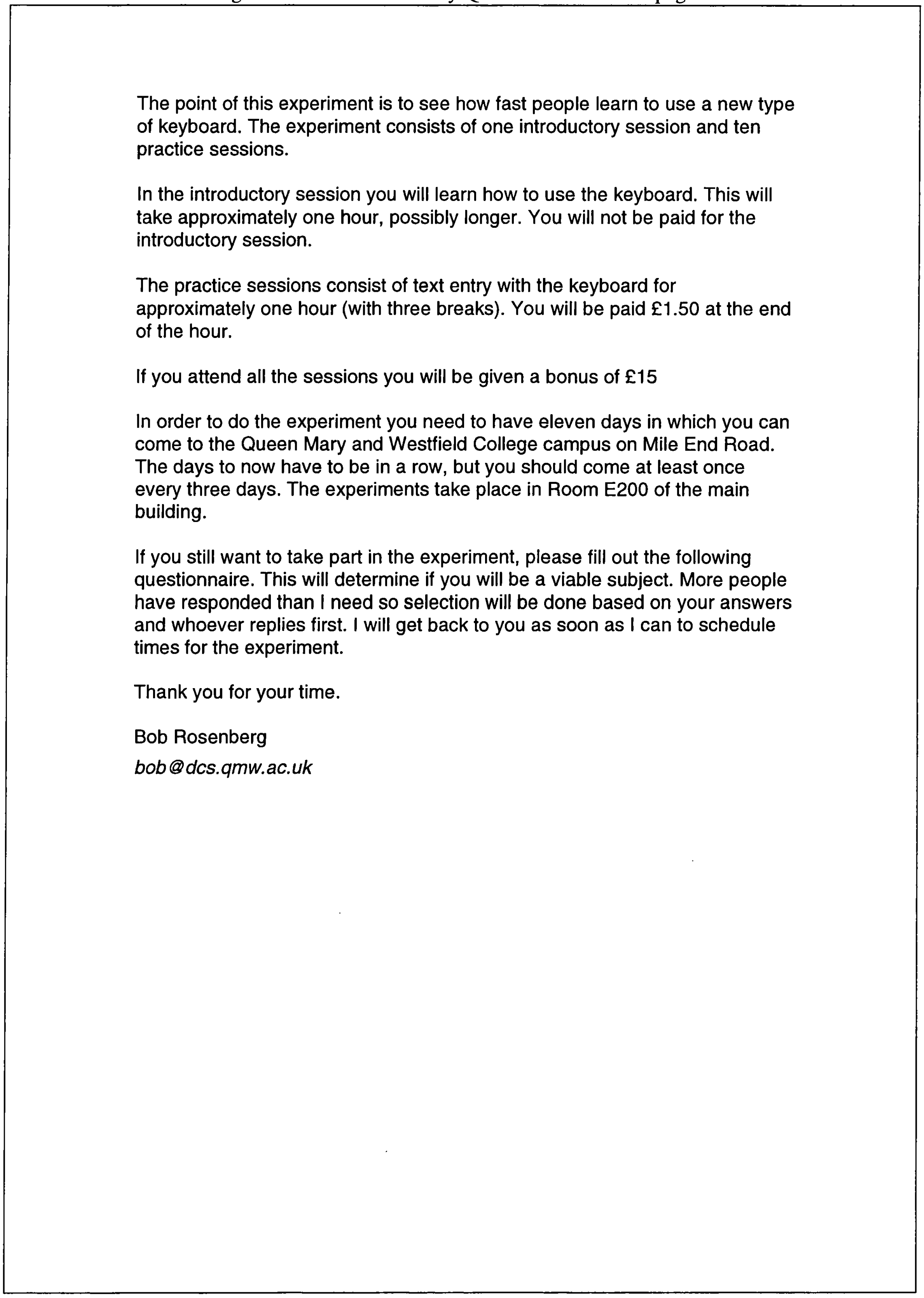
D.1.1 Preliminary Questionnaire for the Glove Text Input Experiment

Figures D.1–D.3 shows the preliminary questionnaire used to determine subject suitability for the glove text input experiment. $\boxed{\textit{Boxed}}$ text explains the rationale behind asking the question.

D.1.2 Daily Questionnaire for the Chording Glove Experiment

The questionnaire in figure D.4 is used to determine the subjects reaction to the glove after each session. The $\boxed{\textit{boxed}}$ text explains which features of the Chording Glove this is trying to measure.

Figure D.1: The Preliminary Questionnaire: Cover page



The point of this experiment is to see how fast people learn to use a new type of keyboard. The experiment consists of one introductory session and ten practice sessions.

In the introductory session you will learn how to use the keyboard. This will take approximately one hour, possibly longer. You will not be paid for the introductory session.

The practice sessions consist of text entry with the keyboard for approximately one hour (with three breaks). You will be paid £1.50 at the end of the hour.

If you attend all the sessions you will be given a bonus of £15

In order to do the experiment you need to have eleven days in which you can come to the Queen Mary and Westfield College campus on Mile End Road. The days to now have to be in a row, but you should come at least once every three days. The experiments take place in Room E200 of the main building.

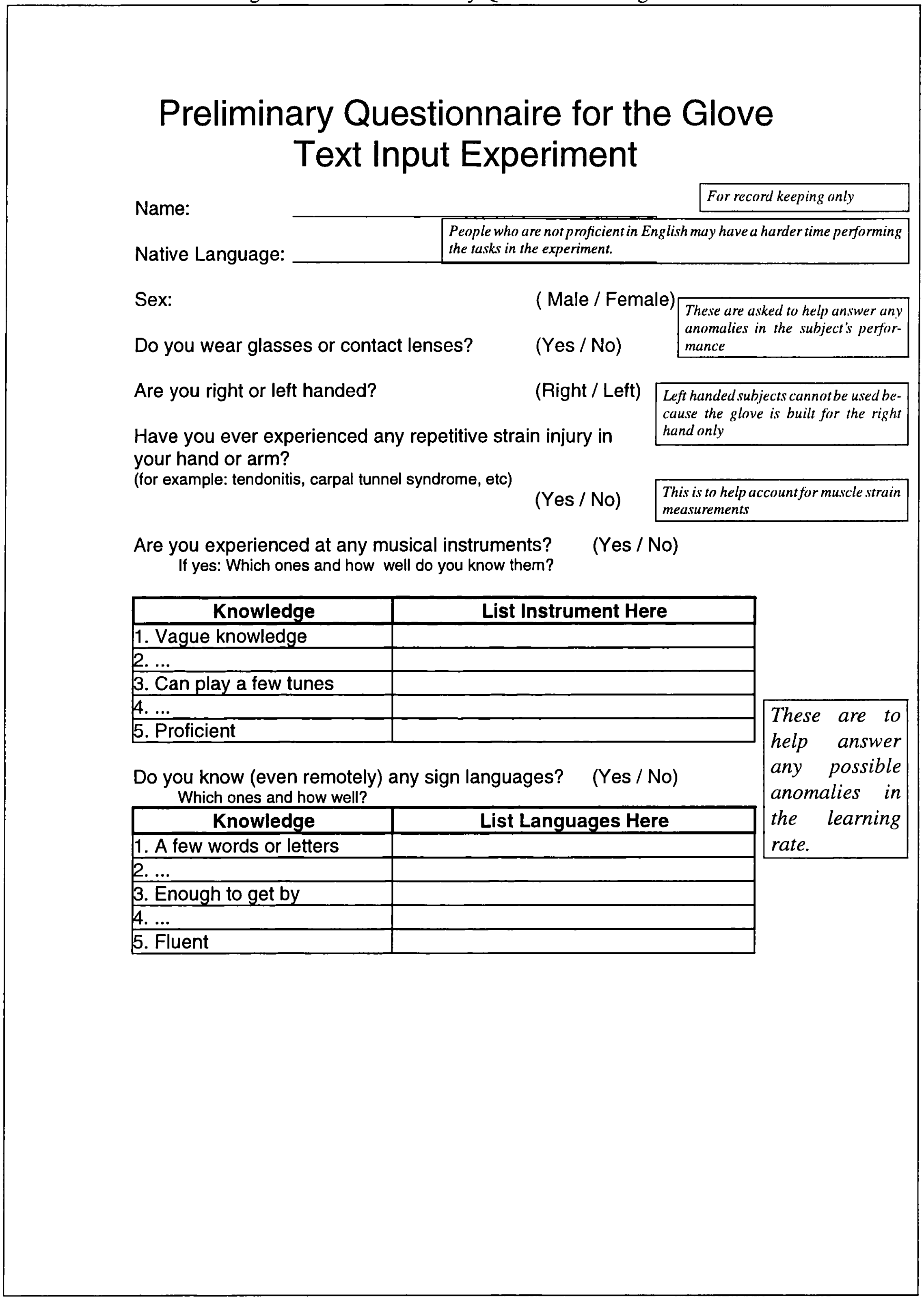
If you still want to take part in the experiment, please fill out the following questionnaire. This will determine if you will be a viable subject. More people have responded than I need so selection will be done based on your answers and whoever replies first. I will get back to you as soon as I can to schedule times for the experiment.

Thank you for your time.

Bob Rosenberg

bob@dcs.qmw.ac.uk

Figure D.2: The Preliminary Questionnaire: Page 1



Preliminary Questionnaire for the Glove Text Input Experiment

Name: ____________________ *For record keeping only*

Native Language: ____________________ *People who are not proficient in English may have a harder time performing the tasks in the experiment.*

Sex: (Male / Female)

Do you wear glasses or contact lenses? (Yes / No)

These are asked to help answer any anomalies in the subject's performance

Are you right or left handed? (Right / Left)

Left handed subjects cannot be used because the glove is built for the right hand only

Have you ever experienced any repetitive strain injury in your hand or arm?
(for example: tendonitis, carpal tunnel syndrome, etc) (Yes / No)

This is to help account for muscle strain measurements

Are you experienced at any musical instruments? (Yes / No)
If yes: Which ones and how well do you know them?

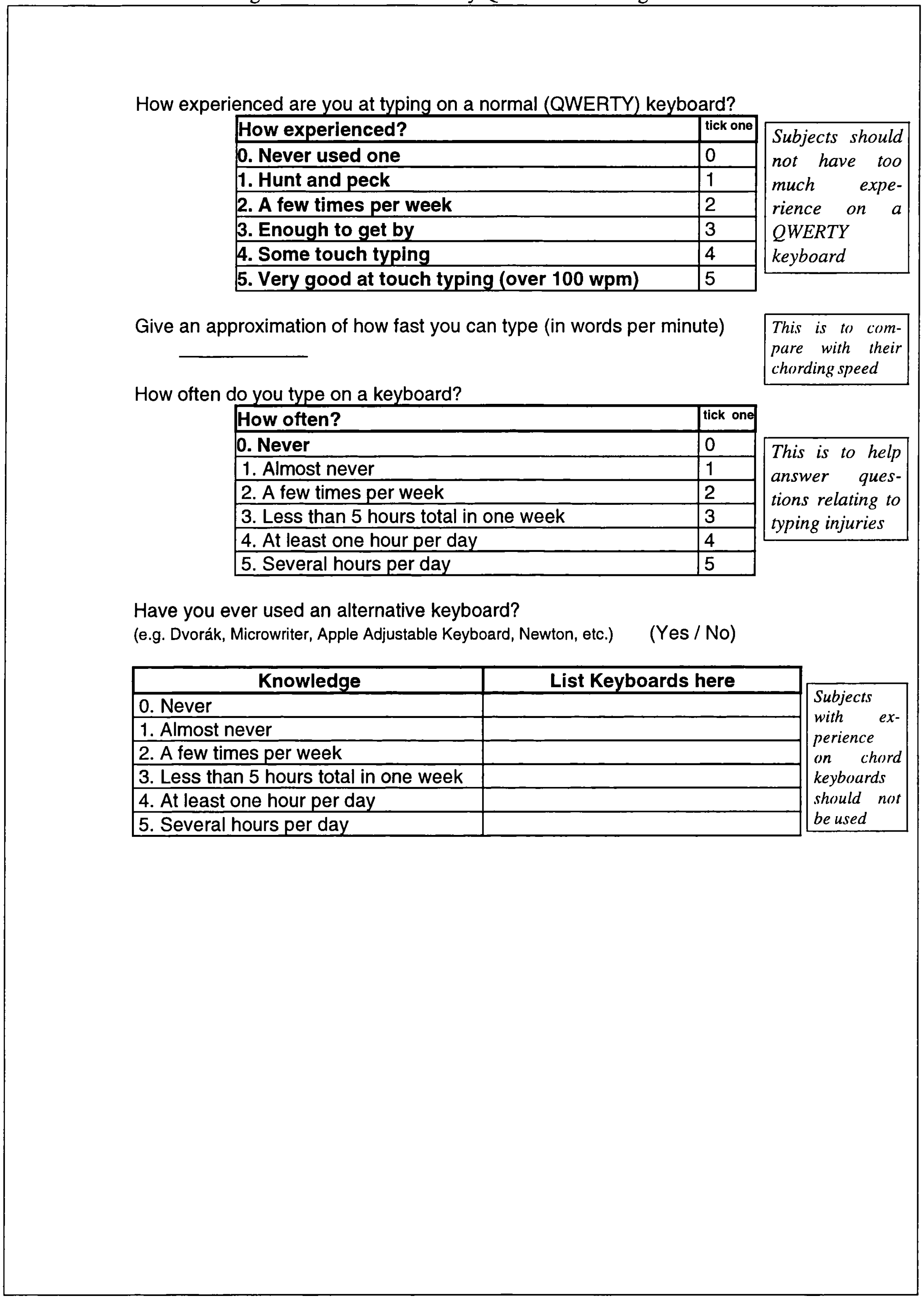
Knowledge	List Instrument Here
1. Vague knowledge	
2. ...	
3. Can play a few tunes	
4. ...	
5. Proficient	

Do you know (even remotely) any sign languages? (Yes / No)
Which ones and how well?

Knowledge	List Languages Here
1. A few words or letters	
2. ...	
3. Enough to get by	
4. ...	
5. Fluent	

These are to help answer any possible anomalies in the learning rate.

Figure D.3: The Preliminary Questionnaire: Page 2



How experienced are you at typing on a normal (QWERTY) keyboard?

How experienced?	tick one
0. Never used one	0
1. Hunt and peck	1
2. A few times per week	2
3. Enough to get by	3
4. Some touch typing	4
5. Very good at touch typing (over 100 wpm)	5

Subjects should not have too much experience on a QWERTY keyboard

Give an approximation of how fast you can type (in words per minute)

____

This is to compare with their chording speed

How often do you type on a keyboard?

How often?	tick one
0. Never	0
1. Almost never	1
2. A few times per week	2
3. Less than 5 hours total in one week	3
4. At least one hour per day	4
5. Several hours per day	5

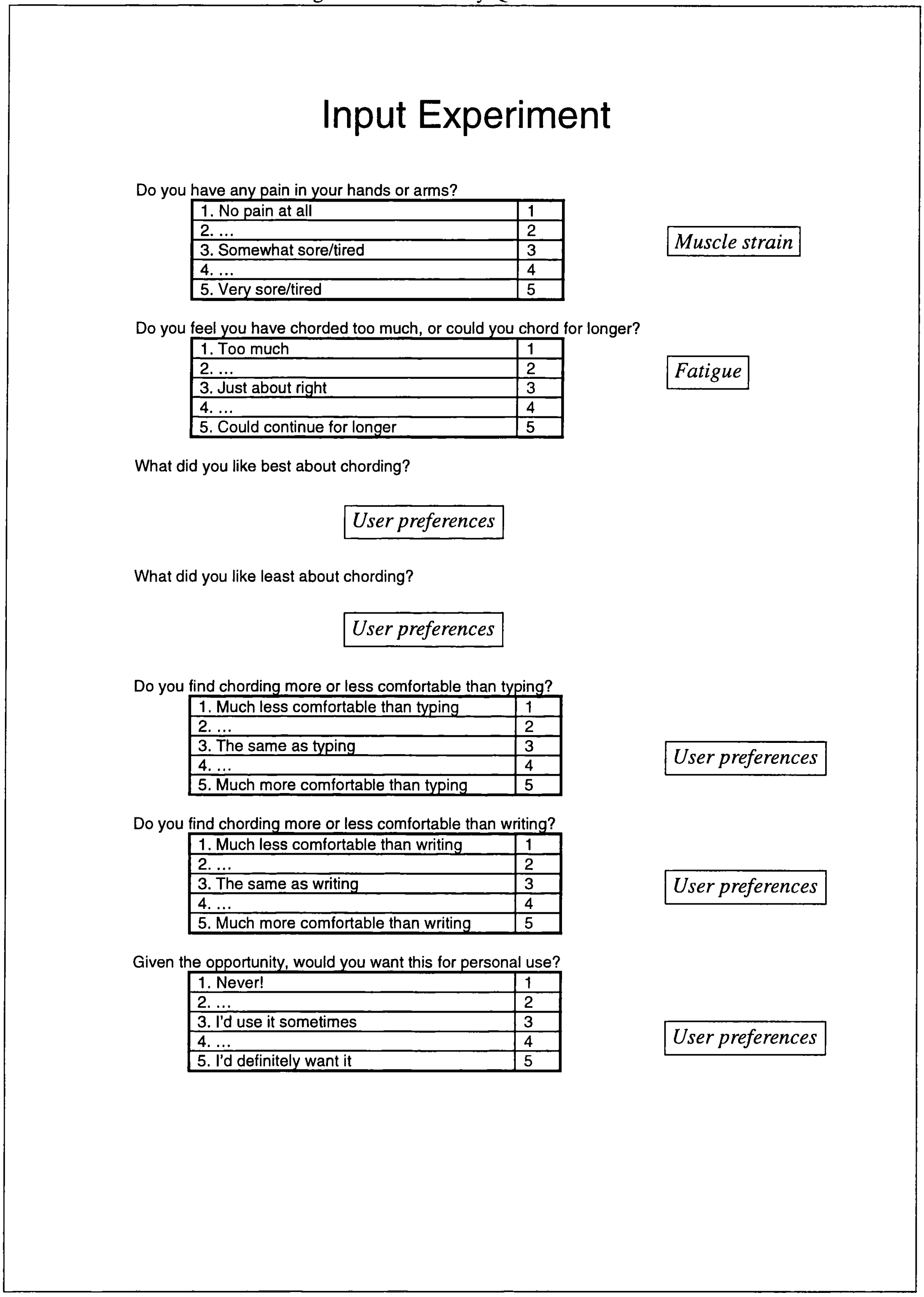
This is to help answer questions relating to typing injuries

Have you ever used an alternative keyboard?
(e.g. Dvorák, Microwriter, Apple Adjustable Keyboard, Newton, etc.) (Yes / No)

Knowledge	List Keyboards here
0. Never	
1. Almost never	
2. A few times per week	
3. Less than 5 hours total in one week	
4. At least one hour per day	
5. Several hours per day	

Subjects with experience on chord keyboards should not be used

Figure D.4: The Daily Questionnaire



Input Experiment

Do you have any pain in your hands or arms?

1. No pain at all	1
2. ...	2
3. Somewhat sore/tired	3
4. ...	4
5. Very sore/tired	5

Muscle strain

Do you feel you have chorded too much, or could you chord for longer?

1. Too much	1
2. ...	2
3. Just about right	3
4. ...	4
5. Could continue for longer	5

Fatigue

What did you like best about chording?

User preferences

What did you like least about chording?

User preferences

Do you find chording more or less comfortable than typing?

1. Much less comfortable than typing	1
2. ...	2
3. The same as typing	3
4. ...	4
5. Much more comfortable than typing	5

User preferences

Do you find chording more or less comfortable than writing?

1. Much less comfortable than writing	1
2. ...	2
3. The same as writing	3
4. ...	4
5. Much more comfortable than writing	5

User preferences

Given the opportunity, would you want this for personal use?

1. Never!	1
2. ...	2
3. I'd use it sometimes	3
4. ...	4
5. I'd definitely want it	5

User preferences

D.2 The Tutorial

The following is the chord keyboard tutorial. This includes the handouts, programme and the script for the experimenter. An arrow ($\longrightarrow$) appears whenever the subject is prompted for a response.

The subject is handed two papers. The first is the keymap (Table 4.1). The second is the list of chords based on hand motions (Figure 3.4).

Experimenter: This is a tutorial to help you learn to type using chords. This is called chording. Feel free to ask any questions you may have as we go along. A chord is made by simultaneously pressing one or more fingers against a surface. The combination of fingers determines what letter is typed.

Experimenter: This *(point to 1st paper)* lists all the chords. The first column is the finger combination. A filled dot means the finger is pressed. An empty dot means the finger is not pressed. The rest of the columns show which chord is made, depending on which shifts are on.

Experimenter: The shift keys are located on the side of the index finger. *(point)* Pressing a shift will turn it on for the next one letter typed. Pressing the shift twice, rapidly, (like double-clicking a mouse) will lock the shift on. The shift will then stay on until pressed again. There are two lights by each shift which tell you the state. No lights, mean the shift is off, the green light mean it's on, but only for the next letter. If the red light is on, the shift is locked. Note that the shift does not turn off when you type `<Space>`, `<BackSpace>`, or `<Return>`.

Experimenter: The third button on the side of the index finger operates the help window *(point)*. While the button is pressed the help window will be open. You don't need to use this for the tutorial unless you really want to, as you have this *(point to 1st paper)* to look at. During the experiment you won't have this, so you will have to use the help button.

Experimenter: When the first shift is on, the glove is in the *upper case* mode. All letters will be capitalised. `<Space>`, `<BackSpace>`, `<Return>`, `<Period>`, and `<Comma>` are unaffected. The second shift turns on the *number and math* mode. All the letters are turned into number and mathematical symbols such as `<Plus>` and `<Minus>`. Note that `<Minus>` is the same as `<Hyphen>`. When both shifts are on together the glove is in *punctuation* mode. This is all the punctuation keys that are above the numbers on the normal keyboard.

Experimenter: This *(point to 2nd paper)* lists all the chords which are similar in shape to the letters they make. You will use this in the tutorial. Practice using the glove until you feel you are ready to start the tutorial. Experiment with different hand positions, such as flat horizontal *(show)* or vertical *(show)*.Tell me when you are ready. *(experimenter must press* `<Escape>` *on the **keyboard** to continue with the tutorial)*

Experimenter: In the tutorial you will be asked to type all the characters, followed by one or more short phrases. When you are done with each line press `<Return>`. Remember to type exactly what it tells you. It may not left you continue until you get the line correct. You will know this has happened, when you press return and nothing happens. Any questions before you begin?

```
********          The Basic Chords          ********


In this section you will be introduced to the basic chords and
how to use them to make lower case letters.

Using the chord list as your guide, type the following chords:
space, backspace, and return
```

$\longrightarrow$

```
Now type full stop (period) and comma. Note that the comma is the same
chord as the full stop, with the thumb added.
Press return when you are done.
```

$\longrightarrow$

```
The word 'the' is very common. It can be easily made by pressing
the first 3 fingers in order:  index, then middle, then ring.
Type the word: the
```

$\longrightarrow$

```
********          The Basic Chords          ********


In this section you will be introduced to the basic chords and
how to use them to make lower case letters.

Using the chord list as your guide, type the following chords:
space, backspace, and return
```

$\longrightarrow$

```
Now type full stop (period) and comma. Note that the comma is the same
chord as the full stop, with the thumb added.
Press return when you are done.
```

$\longrightarrow$

```
The word 'the' is very common. It can be easily made by pressing
the first 3 fingers in order:  index, then middle, then ring.
Type the word: the
```

$\longrightarrow$

Experimenter: Use this *(point to 2nd paper)* for this part of the tutorial.

```
There are several chords that form the fingers into similar shapes to
the letters they represent.  Some of these chords are based on sign language.
Use the chord shape list you were given as a reference.
Type the following letters a few times each and press return:


c       shaped like c

⟶

f       shaped like F

⟶

i       shaped like i, as in sign language

⟶

l       all the fingers are in a line, they're shaped like l

⟶

m       shaped like m

⟶

n       shaped like n

⟶

r       shaped like r

⟶

y       shaped like y, as in sign language

⟶


Now type the sentence:
let their chimney fly.          (Don't forget the full stop!)

⟶


Referring to the chord list, type the letter s

⟶
```

Now type the sentences:

this is it.

$\longrightarrow$

let their chimney fly.

$\longrightarrow$

Type the letters a and d:

$\longrightarrow$

Now type:

and

$\longrightarrow$

this and that

$\longrightarrow$

let their chimney fly.

$\longrightarrow$

this is it.

$\longrightarrow$

Type the letter g

$\longrightarrow$

Now type:

ing

$\longrightarrow$

Note that 'ng' is formed by the first two fingers(index and middle) followed by the second two fingers (middle and ring).

Type again: ing

$\longrightarrow$

Now type:

thing

$\longrightarrow$

```
this and that thing
```

$\longrightarrow$

```
let their chimney fly.
```

$\longrightarrow$

```
The letters o, b, and p, are formed using similar chords.
Type three times: obp
```

$\longrightarrow$

```
Now type:
bop
```

$\longrightarrow$

```
this and that thing
```

$\longrightarrow$

```
let their chimney fly.
```

$\longrightarrow$

```
bop
```

$\longrightarrow$

Experimenter: Use this *(point to 2nd paper)* for the next part of the tutorial.

```
There are three letters similar to the letter u:
      u, w, and v.
The chords for these letters are also similar. Also note that the
chord for u is shaped like a U made of the index and little fingers.

Using the chord shape list as a reference, type a few times each:
u
```

$\longrightarrow$

```
w       (like u + the thumb)
```

$\longrightarrow$

```
v       (like u + the ring finger)
```

$\longrightarrow$

```
four wives

⟶

vipers usually wobble.

⟶

this and that thing

⟶

let their chimney fly.

⟶

four wives

⟶


Now type the letters x, z, k, j, and q, three times each.
Note that z is just s + the little finger.
also note: q is u plus the middle finger.

⟶

Type:
six joke quizzes

⟶

vipers usually wobble.

⟶

this and that thing

⟶

let their chimney fly.

⟶

four wives bop.

⟶

the quick brown fox jumps over the lazy dog.

⟶
```

```
Congratulations!  You are now over the hard part.
Rest for a moment, then press return to continue.
```

⟶

```
********        The Shift Modes         ********
```

```
In the last section you learned all of the basic chords and characters
that are listed in the first and second columns of the chord list.
Now you will learn about the next three columns:
caps (upper case), number, and punctuation modes.

There are two shift keys: caps and number.
Pressing a shift key once turns it on for the next one character typed.
Pressing it twice rapidly, locks it on for all characters typed.
When the shift is on, pressing it once turns it off.

The third column in the chord list shows which characters will be
generated by each chord when caps is on.

Press return to continue.
```

⟶

```
********        The Caps Mode       ********
```

```
Press the caps key once. A green light should turn on next to it. You
have now set the caps mode. The next one character you type will be
capitalised.

Type the sentence:
Let Their Chimney Fly.
```

⟶

```
Press the caps twice in rapid succession. A red light should turn on.
This means that caps are locked.
```

Type the sentence:

THIS AND THAT THING

⟶

To exit the caps mode, press the caps key once. Note that light turns off when you exit the mode.

Important: Return, space and backspace are unaffected by the shift keys. If the shift key is pressed once, it will stay on until a key other than return, space, and backspace are pressed. Also, the full stop and comma are unchanged by the caps key.

Press return to continue.

⟶

******** The Number Mode ********

The number key works in the same way as the caps key.
The fourth column in the chord list shows the characters generated by the chords when the number key is on.
(note that the full stop and comma are unchanged by the number mode)

Double press the number key to lock on the number mode.

The numbers 1-4 are made in this mode by using the first four fingers.
Type the following numbers five times each:

1	(index finger)

⟶

2	(middle finger)

⟶

3	(ring finger)

⟶

4	(little finger)

$\longrightarrow$

The numbers 5-8 are just the first four fingers, plus the thumb.
Type the follwing numbers five times each:

5	(index finger + thumb)

$\longrightarrow$

6	(middle finger + thumb)

$\longrightarrow$

7	(ring finger + thumb)

$\longrightarrow$

8	(little finger + thumb)

$\longrightarrow$

Type the following numbers five times each:

9	(index finger + middle + thumb)

$\longrightarrow$

0	(index finger + middle + ring + thumb)

$\longrightarrow$

Type:
1 2 3 4 5 6 7 8 9 10

$\longrightarrow$

The number mode also contains mathematical symbols. Some of them are based on letters. These can be used as mnemonic devices.
Type the following five times each:

+	P for Plus

$\longrightarrow$

-	M for Minus

$\longrightarrow$

*	X for multiply or B for By.

$\longrightarrow$

/	D for Division

$\longrightarrow$

=	L for equaL

$\longrightarrow$

^	K for Karat

$\longrightarrow$

Type:

2.7 + 9,000/45 - 2^6 = 138.7

$\longrightarrow$

Parentheses are made in the number mode by:

(	first two fingers (index and middle)

$\longrightarrow$

)	second two fingers (middle and ring)

$\longrightarrow$

Square brackets are made by:

[	(index and little fingers)

$\longrightarrow$

]	[+ thumb (index and little + thumb)

$\longrightarrow$

type:

() () () () [] [] [] [] []

$\longrightarrow$

Turn off number mode by pressing the number key once, and rest for a moment.

Press return to continue.

$\longrightarrow$

******** The Punctuation Mode ********

Punctuation mode is made by having both the caps and number modes on at the same time.
Double press caps. Now double press number.
Note: they can be pressed in either order.

Full stop and comma can NOT be made in this mode. The full stop becomes colon and comma becomes semicolon. Return, space and backspace remain unaffected.

Type colon and semicolon:

$\longrightarrow$

Experimenter: Use this *(point to 2nd paper)* for the next two characters.

There are three quote characters: ' (the apostrophe, single quote, or close quote), " (double quote), and ‘ (backwards quote or open quote).
Using the chord shape list as a reference, type the following:
' (index finger)

$\longrightarrow$

" (index + middle)

$\longrightarrow$

‘ (index + middle + thumb)

$\longrightarrow$

The rest of the punctuation chords in this mode correspond to letters.
Again, this can be used mnemonically.

Type the following five times each:
! E for Exclamation point

$\longrightarrow$

$ S for dollar sign

$\longrightarrow$

P for Pound sign

$\longrightarrow$

< L for Less than

$\longrightarrow$

```
>       G for Greater than

⟶

&       A for And

⟶

|       O for Or (as in the c programming language)

⟶

%       C for perCent

⟶

?       I for Inquiry or Q for Question

⟶

~       Y for wigglY

⟶

@       H for Hat - (Hat rhymes with 'at')

⟶

There are also punctuation chords based on the number chords:
For example, both * and ^ are the same chords in both punctuation
and number modes.

Type the following five times each:
*       X for multiply or B for By.

⟶

^       K for Karat

⟶

{       like [

⟶

}       like ]

⟶

_       like - (Minus)

⟶
```

```
\       like / (Division)

⟶


Turn off punctuation mode by pressing the caps and number keys once.


Press return to continue.

⟶



********         Review           ********

Type the following:


I said, "Hello world! I have 16 oranges."

⟶

six joke quizzes

⟶

Vipers usually wobble.

⟶

this and that thing

⟶

2.7 + 9,000/45 - 2^6 = 138.7

⟶

Let their chimney fly.

⟶

Four wives bop.

⟶

The Quick Brown Fox Jumps Over The Lazy Dog!

⟶
```

```
*******        Congratulations! You are finished!        *******
```

⟶

⟶

The experimenter presses `<Escape>` *on the keyboard to exit the tutorial.*

D.3 The Chording Sessions

The chording sessions consisted of text entry trial and data entry trials. The text entry trials used the text from the book *Alice's Adventures in Wonderland.* One subject managed to get as far as the third paragraph of Chapter Three (399 lines).

The data entry text was mostly tabular stock market data. The intent was to use characters in the Number mode, with occasional switching to punctuation and text. The furthest any subject achieved was 158 lines. The first 158 lines of the data entry text follows:

```
 162.31   +3.15   159.55   162.32   159.55
 471.23   +4.12   467.48   471.48   467.48
 399.51   +3.66   396.00   399.80   396.00
 448.62   +0.58   448.04   449.13   448.00
 263.98   -0.69   264.63   264.89   263.76
  94.32   -0.31    94.32    94.32    94.32
  99.13   -0.25    99.13    99.13    99.13
  89.51   -0.12    89.51    89.51    89.51
1277.22   +8.87  1267.78  1278.72  1267.78
3829.73  +28.26  3798.10  3833.43  3798.10
1484.84  +12.58  1472.03  1486.24  1472.03
 177.30   +0.59   176.84   177.76   176.71
 710.24   +0.46   709.64   711.02   708.50
 768.18   +6.06   763.61   768.18   762.12
```

```
   890.14    +0.42    890.20    890.47    888.59
   777.24    +4.14    773.69    777.30    773.69
   901.92    -3.86    906.40    906.52    900.53
   681.00    -0.17    681.73    682.72    678.96
   342.05    +2.84    339.90    342.05    339.21
   719.47    -0.67    720.14    721.38    718.79
   254.79    +1.54    253.25    254.94    253.25
   200.51    +0.95    199.56    200.60    199.56
   322.81    +2.19    320.62    322.97    320.62
   227.08    +0.09    226.99    227.61    226.67
   200.13    +1.17    198.96    200.26    198.96
   433.54    +3.07    430.52    433.97    430.52
   465.94    +3.59    462.32    466.30    462.32
   250.39    +1.01    249.58    250.39    249.57
   282.20    +1.06    281.17    282.21    281.17
  2089.29   +10.94   2086.75   2094.30   2086.75
  9378.92   +11.07   9363.44   9397.42   9334.89
   3095.3    +19.4    3076.9    3095.3    3069.5
2486.640  -40.350   526.990  2540.380  2484.900
  1941.08    -7.27   1940.27   1946.64   1927.64
  2326.94    +4.35   2322.59   2333.31   2322.59
   1926.5     +0.0    1926.5    1926.5    1926.5
19261.45   -22.91  19246.45  19298.96  19115.72
  1705.15    -3.45   1709.90   1714.41   1705.15
*****
1   4,659,300       31 1/4       -  1/8
2   3,258,600       48 7/8       +2 7/8
3   2,978,900       52 1/4       -1 1/4
4   2,901,200       23 1/2       -  1/8
5   1,928,700       62 1/2       +1
6   1,908,900       29           +  1/2
7   1,850,100       20 3/8       -  1/4
8   1,833,300       15 5/8       -  1/2
9   1,809,200       38 3/4       +  1/8
10  1,782,200       16 1/4        0
11  6,173,800       33 3/4       +  3/4
12  3,872,300       69 1/4       +7 7/8
13  3,678,800       46 5/16      -2 1/16
```

```
14  3,045,100       64 1/2       +2 1/4
15  2,861,900       74 1/4       +2 3/4
16  2,347,400       62 1/4       +1 5/8
17  2,297,500       23 1/8       +  3/8
18  2,213,800       22 1/8       +  1/2
19  1,868,800       44 1/8       +1
20  1,336,400       39 5/8       +  3/4
21    667,900       33           +1 3/4
22    481,100        8 3/8       +1 3/8
23    437,700       33 1/4       +  3/4
24    427,800        4 1/16      +  9/16
25    422,000        8 7/8       +  1/4
26    344,400        8           +  1/8
27    287,500       40 3/4       +  3/4
28    252,700       20 5/8       +3 1/4
29  1,858,800       26 5/16      -  5/16
30    250,300        3 7/8       -  3/16
*****
London Gold            $385.10    -0.15     385.25
New York Gold          $384.50    -0.5      385.00
Fed Funds Rate         $5.75      +0.875      4.875
UST 3m  Bill Rate      $5.27      +0.03       5.24
UST 6m  Bill Rate      $5.69      -0.02       5.71
UST 1y  Bill Rate      $6.11      +0.08       6.03
UST 3y  Note % Yield   $7.36      -0.05       7.41
UST 5y  Note % Yield   $7.65      -0.05       7.70
UST 10y Note % Yield   $7.93      -0.06       7.99
UST 30y Bond % Yield   $8.08      -0.07       8.15
*****
 4.500   4.125   4.375    808    +58% +0.125  (+3)
17.375  16.875  17.125   4379    +56% -0.125  (-1)
24.750  24.250  24.375   1365   +133% -0.375  (-2)
31.625  31.250  31.250  55511   +215% -0.125  (-0)
27.875  27.250  27.875   3132    +63% +0.500  (+2)
10.375   9.875  10.125   8254    +78% +0.125  (+1)
60.250  59.250  60.000  14542    +72% +1.000  (+2)
 5.250   4.875   5.000    419    +56% -0.125  (-2)
43.875  42.875  43.875   3167    +64% +1.000  (+2)
```

```
13.125  12.625  12.875    749  +109% -0.250  (-2)
 6.625   6.250   6.625   4054  +116% +0.375  (+6)
48.750  47.375  48.750  38735   +86% +2.750  (+6)
39.125  38.750  38.750   7346  +177% -0.625  (-2)
 9.875   9.000   9.625   2347  +220% +0.250  (+3)
12.750  12.375  12.500    514  +147% -0.125  (-1)
31.250  30.750  31.250  18245   +56% +0.500  (+2)
38.375  38.000  38.000   6532   +66% -0.500  (-1)
 8.375   8.125   8.250   3921  +653% +0.000  (+0)
16.250  15.625  15.625  21744   +73% -0.500  (-3)
 9.125   8.875   9.125    475  +303% +0.250  (+3)
60.125  58.000  59.500   3527   +55% +1.500  (+3)
30.500  30.000  30.250   1188  +101% +0.375  (+1)
23.750  23.000  23.250   9514   +64% +0.000  (+0)
23.250  23.000  23.125    646  +873% +0.250  (+1)
29.375  28.875  29.250  11287   +56% +0.000  (+0)
30.125  29.750  30.000   2273  +106% +0.250  (+1)
57.375  56.125  57.250   7054   +69% +0.375  (+1)
38.000  37.000  37.375   5831   +87% -0.750  (-2)
50.875  49.875  50.750   3467   +63% +1.250  (+3)
51.875  51.375  51.750  10027   +60% -0.062  (-0)
*****
exa           E    10^18      10^36      10^54
peta          P    10^15      10^30      10^45
tera          T    10^12      10^24      10^36
giga          G    10^9       10^18      10^27
mega          M    10^6       10^12      10^18
hectokilo     hk   10^5       10^10      10^15
myria         ma   10^4       10^8       10^12
kilo          k    10^3       10^6       10^9
hecto         h    10^2       10^4       10^6
basic unit    -    1 meter,   1 meter^2  1 meter^3
                   1 gram,
                   1 liter
deci          d    10^-1      10^-2      10^-3
centi         c    10^-2      10^-4      10^-6
milli         m    10^-3      10^-6      10^-9
decimilli     dm   10^-4      10^-8      10^-12
```

```
centimilli   cm  10^-5      10^-10     10^-15
micro         u  10^-6      10^-12     10^-18
nano          n  10^-9      10^-18     10^-27
pico          p  10^-12     10^-24     10^-36
femto         f  10^-15     10^-30     10^-45
atto          a  10^-18     10^-36     10^-54
*****
28.750  28.500  28.750   5985    +99% +0.250  (+1)
34.875  33.250  34.875   2916    +68% +1.625  (+5)
33.625  33.250  33.500   2913    +82% -0.125  (-0)
 5.875   5.500   5.875    529    +55% +0.250  (+4)
46.750  44.625  46.750   1761    +55% +2.375  (+5)
 8.250   7.750   8.000   4987   +160% +0.125  (+2)
46.875  44.875  45.250  15515   +175% -1.500  (-3)
59.250  58.750  59.000   8571   +138% -0.500  (-1)
 2.625   2.375   2.500   1902   +153% -0.125  (-5)
56.250  55.500  56.250   1279    +44% +0.750  (+1)
51.375  50.625  51.250  12806    -15% +0.875  (+2)
74.500  73.875  74.125   6323    +27% +0.375  (+1)
 1.812   1.500   1.688   2264    +13% +0.125  (+8)
22.375  21.875  22.125    522    +46% -0.250  (-1)
29.375  28.875  29.250  11287    +56% +0.000  (+0)
41.375  39.500  40.250    468   +180% -0.500  (-1)
31.625  30.625  31.250   9569    +87% -0.125  (-0)
31.750  30.750  31.250   2501   +605% +0.000  (+0)
19.625  19.500  19.625    101   +111% +0.000  (+0)
10.500  10.000  10.500    288    +90% +0.062  (+1)
35.000  34.250  34.375   2141   +259% -1.625  (-5)
 8.000   7.250   7.562    696    +58% -0.438  (-5)
10.750  10.375  10.500    255   +132% -0.250  (-2)
```

Appendix E

Calculating the Effective Width for Fitt’s Law

Fitts' law requires W to be a one-dimensional term. In the performance experiments, W needed to be derived from a two-dimensional button. It was decided to use a method described by MacKenzie (MacKenzie, 1992) which the span of the rectangle across the direction of the approach vector is used.

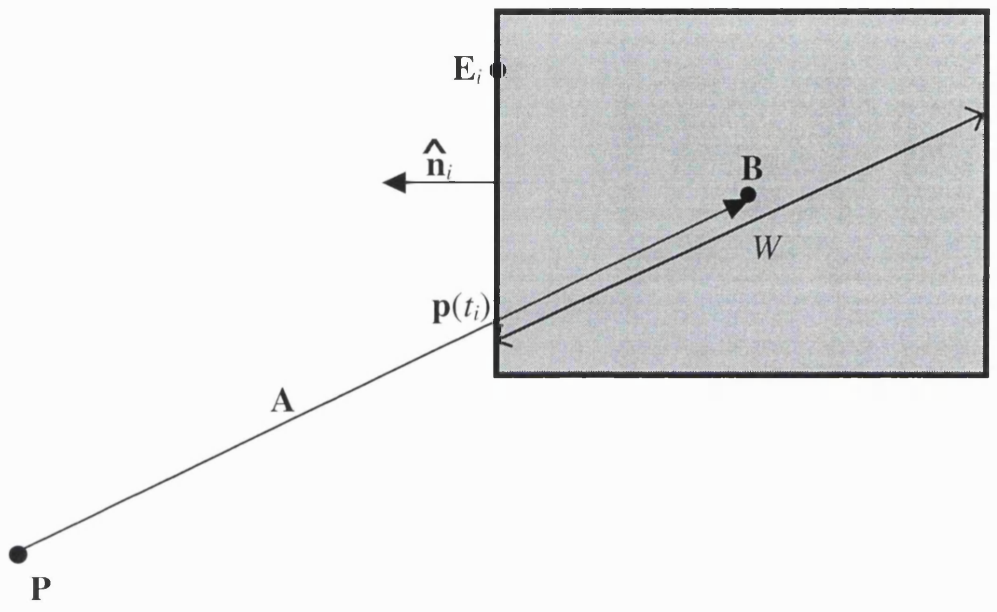


Figure E.1: The button and the approach vector

The position of the pointer at the start of the task is point $\mathbf{P}$. The centre of the button is point $\mathbf{B}$. The approach vector $\mathbf{A}$ is defined as:

$$\mathbf{A} = \mathbf{B} - \mathbf{P} \tag{E.1}$$

The point $\mathbf{p}$ is the point where the vector $\mathbf{A}$ intersects the rectangle. The point $\mathbf{p}$ can be found by using the Cyrus-Beck clipping algorithm (Foley et al., 1990), which is described as follows:

$\mathbf{p}$ is a function of the parameter t such that:

$$\mathbf{p}(t) = \mathbf{P} + t\mathbf{A}. \tag{E.2}$$

$\mathbf{E}_i$ is any arbitrary point on edge i and $\hat{\mathbf{n}}_i$ is the outward normal vector for that edge. This yields:

$$\hat{\mathbf{n}}_i \cdot (\mathbf{p}(t_i) - \mathbf{E}_i) = 0 \tag{E.3}$$

where t_i is the value of the parameter t where it intersects edge i. This can be solved for t_i by:

$$\hat{\mathbf{n}}_i \cdot (\mathbf{P} + t_i\mathbf{A} - \mathbf{E}_i) \quad = \quad 0 \tag{E.4}$$

$$\hat{\mathbf{n}}_i \cdot (\mathbf{E}_i - \mathbf{P}) \quad = \quad \hat{\mathbf{n}}_i \cdot t_i\mathbf{A} \tag{E.5}$$

and finally,

$$t_i = \frac{\hat{\mathbf{n}}_i \cdot (\mathbf{E}_i - \mathbf{P})}{\hat{\mathbf{n}}_i \cdot \mathbf{A}} \tag{E.6}$$

$\mathbf{p}$ is then chosen from the smallest defined value of t_i between 0 and 1, yielding:

$$W = 2|\mathbf{B} - \mathbf{p}| \tag{E.7}$$

Appendix F

Deriving the Neural Network

The neural network used in the Biofeedback Pointer is simple perceptron which uses the EMG spectra as input, and outputs a two-dimensional position vector. In this appendix we will show the derivation of the equations used to calculate the weights for the neural network.

F.1 Definitions

The input data S is a set of N frequency spectra

$$\mathsf{S} = (\mathrm{s}_0 \ldots \mathrm{s}_{N-1}) \tag{F.1}$$

Each frequency spectrum is an array consisting of B bins.

$$\mathrm{s}_i = (s_{i,0} \ldots s_{i,B-1}) \tag{F.2}$$

In order facilitate manipulation of the data, it can be converted to the linear array $\mathbf{a}$ of $N + B + 1$ points so that

$$\mathbf{a} = (1, s_{0,0} \ldots s_{0,B-1}, s_{1,0} \ldots s_{i,j} \ldots . s_{N-1,B-1}) \tag{F.3}$$

The leading 1 in $\mathbf{a}$ is to allow for an offset value for the weights.

There is a set of test points $(\mathbf{x}, \mathbf{y})$ where (x_i, y_i) is the ith test point associated with the ith input data $\mathbf{a}_i$. There is a set of weights $\mathbf{w}_x$ and $\mathbf{w}_y$ which, when applied to each $\mathbf{a}_i$ yield a set of points (x'_i, y'_i) which are as close as possible to original points. In other words:

$$x_i \approx x'_i = \mathbf{w}_x^{\mathsf{T}} \mathbf{a}_i \quad \textit{or} \quad \mathrm{x} \approx \mathrm{x}' = \mathbf{w}_x^{\mathsf{T}} \mathsf{a} \tag{F.4}$$

$$y_i \approx y'_i = \mathbf{w}_y^{\mathsf{T}} \mathbf{a}_i \quad \textit{or} \quad \mathrm{y} \approx \mathrm{y}' = \mathbf{w}_y^{\mathsf{T}} \mathsf{a} \tag{F.5}$$

where a is the matrix generated from all $\mathbf{a}_i$.

The errors ϵ_x and ϵ_y are defined by

$$\epsilon_x = \mathrm{x} - \mathrm{x}' \tag{F.6}$$

$$\epsilon_y = \mathrm{y} - \mathrm{y}' \tag{F.7}$$

In order to find the best value for the weights we want to minimise the errors. We can solve exactly for the weights which minimise the errors by a method very similar to a linear least squares fit.

F.2 Derivation

Since the equations for the error are equivalent and independent, we can examine the x case to derive both equations. Dropping the x subscripts for now, we find the error squared is:

$$\epsilon^2 = (\mathrm{x} - \mathrm{x}')^2 \tag{F.8}$$

$$= (\mathrm{x} - \mathrm{w}^{\mathsf{T}} \mathsf{a})^2 \tag{F.9}$$

$$= x^2 + \mathrm{w}^{\mathsf{T}} (\mathsf{a}\mathsf{a}^{\mathsf{T}}) \mathrm{w} - 2(\mathrm{x}\mathsf{a}^{\mathsf{T}}) \mathrm{w} \tag{F.10}$$

We then define $\mathsf{R} = \mathsf{a}\mathsf{a}^{\mathrm{T}}$ and $\mathbf{q} = \mathbf{x}\mathsf{a}^{\mathrm{T}}$. Note that R is independent of x. Replacing the terms in the Equation F.10 we get:

$$\epsilon^2 = x^2 + \mathbf{w}^{\mathrm{T}}\mathsf{R}\mathbf{w} - 2\mathbf{q}\mathbf{w} = g(\mathbf{w}) \tag{F.11}$$

Where g is a function of $\mathbf{w}$ which we need to minimise to find the best values for $\mathbf{w}$. This is done by:

$$\sum_{i=0}^{N+B} \frac{\partial g(w_i)}{\partial w_i} = 2\mathsf{R}\mathbf{w} - 2\mathbf{q} \tag{F.12}$$

We define the $\mathbf{w}^*$ as the array which makes Equation F.12 equal to zero.

$$\begin{aligned} 2\mathsf{R}\mathbf{w}^* - 2\mathbf{q} &= 0 && \text{(F.13)} \\ \mathsf{R}\mathbf{w}^* &= \mathbf{q} && \text{(F.14)} \\ \mathbf{w}^* &= \mathsf{R}^{-1}\mathbf{q} && \text{(F.15)} \end{aligned}$$

Which for both x and y becomes:

$$\begin{aligned} \mathbf{w}_x^* &= \mathsf{R}^{-1}\mathbf{q}_x && \text{(F.16)} \\ \mathbf{w}_y^* &= \mathsf{R}^{-1}\mathbf{q}_y && \text{(F.17)} \end{aligned}$$

Solving these equations is easily done by well established methods. The Singular Value Decomposition method described in Press et al. (1992) was used for the Biofeedback Pointer.

Appendix G

Technical Specifications for the Biofeedback Pointer

G.1 The Physiological Amplifiers

The four physiological amplifiers are custom-made and designed specifically for reading the EMG. Each one is contains the amplification hardware (see Section 3.3.3) and is encased in a hard epoxy resin, approximately 4.5cm long and 1.5cm wide. On the bottom side of each physiological amplifiers are two connectors spaced 2.54cm apart. The specifications for these amplifiers is detailed in Table G.1.

Table G.1: Physiological Amplifier specifications

Input Impedance	10GΩ
Input Offset Limit	$\pm$350mV
Overall Gain	$\times$1000
Noise	5μV p-p
Frequency Response	16Hz - 1.6kHz
Common Mode Rejection	$>$ 100dB
Current Protection Limit	$<$ 500μA
Supply Voltage	$\pm$4.5V
Amplifier Current	3mA

G.2 The Analogue-Digital Converter

The analogue-digital converter used was a Pico ADC-11 (Pico Technology LTD, n.d.). The specifications are described in Table G.2 and the pin connections to the EMG amplifier are described in Table G.3.

Table G.2: Specifications for the Pico ADC-11

Resolution	10 bits
Number of input channels	11
Input voltage range	0–2.5V
Maximum sampling rate	max 18k samples per second
Repeatability	±1 at 25°C
Input overvoltage protection	±30V
Input impedance	$> 1\text{M}\Omega$
Digital output voltage	typically 3–5V, depending on computer and load
Digital output impedance	approx. 1–2kΩ depending on type of computer
Input collector	25 way female D-type
Output collector	25 way male D-type (connects to PC printer port)

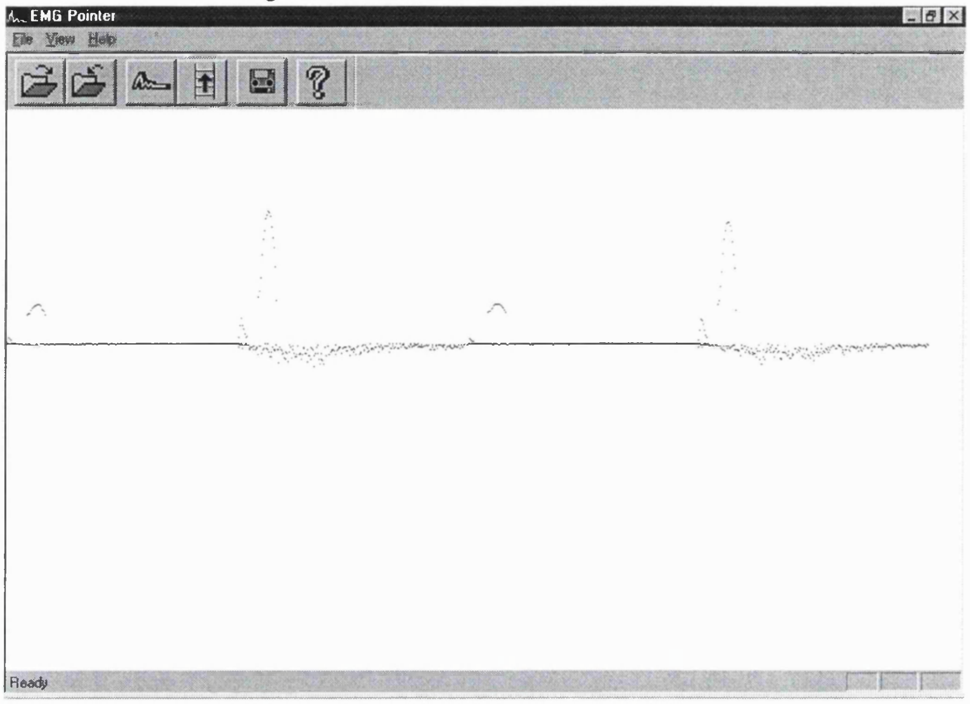
Table G.3: Output pin connections for the ADC-11

Pin	Function	Use
1	Digital output / voltage output	unused
2	Signal ground	ground
3	Analogue input channel 1	EMG 1
4	Analogue input channel 2	EMG 2
5	Analogue input channel 3	EMG 3
6	Analogue input channel 4	EMG 4
7	Analogue input channel 5	unused
8	Analogue input channel 6	
9	Analogue input channel 7	
10	Analogue input channel 8	
11	Analogue input channel 9	
12	Analogue input channel 10	
13	Analogue input channel 11	
14–25	Unused	

Appendix H

The Biofeedback Pointer Software

Figure H.1: The Biofeedback Pointer Software



H.1 Introduction

The windows application "EMG Pointer" was written to provide an easy to use graphic interface to the Biofeedback Pointer. The primary use of the application is to read the Analogue/Digital converter (ADC) and convert the data into pointer motion. In addition the application can train the neural network, run the performance test experiment, and perform several diagnostic functions.

The application is a 32 bit Windows 95 application written in Microsoft Visual C++ on a Pentium 133MHz desktop computer.

H.2 Using the Application

The application is operated by either buttons, pull-down menus, or hotkeys. Information is either displayed in the main window, status bar, a pop-up window, or saved to disk as a binary data file.

H.2.1 Button control

There are two sets of buttons which control the application. The first is the standard Windows 95 set of window control buttons. The upper right corner contains buttons which iconify, maximise and close the application. The upper left corner contain the application's icon which is designed to resemble the frequency spectrum of an EMG. Pressing it will display a pull-down list of standard window control operations. The second set of buttons control software specific functions and are detailed here.

Retrieve a saved neural network file The weights of the neural network can be saved as a binary file, for analysis or to be used later. This button reads the file `data.nnt` and stores the weights in a new neural network. If the EMG data is being read, the pointer control will start immediately.

Save a neural network in a file This saves the weights of the neural network in a file called `data.nnt`.

Start or stop reading EMG data This starts or stops the application reading the ADC on the parallel port (Section H.3). The data will be displayed in the main window (Section H.2.4).

If the pointer has been trained, or the neural net retrieved from a file, the application will take immediate control over the pointer. When pressed again the application will stop controlling the pointer.

Note that the mouse still works when the application has control of the pointer. In this case, either device can be used to control the pointer.

Train the neural network When this is pressed, the neural network training will start immediately. The pointer moves to the centre of the screen and performs several motion which the user must follow (see Section 3.3.4 for exact details). After the data is collected the programme pauses a short time while the neural network is calculated (Appendix F). The network is calculated eight times, with time delays ranging from zero to 448ms. The weights with the least error are used in the neural network. Pointer control begins immediately upon termination of the calculations. It is possible that some the matrixes are unsolvable. If this occurs, no network is generated, and no pointer control occurs. The network must be trained again. In practice this is an extremely rare occurrence, but it can happen. If this button is pressed again while training, the training stops immediately without calculating the network.

Start/stop recording EMGs When pressed, this causes all EMGs to be saved in the file `emg.dat` until the button is pressed again. Every 64ms, the 4 reduced spectra of 8 points are saved. This is a diagnostic tool for saving data to be analysed by external programmes.

Display application information This pops up a window displaying the application's name, author and copyright information.

H.2.2 Menus

There are three pull-down menus which can be used to control the application.

File	Exit	—	Exits the application
View	Toolbar	—	Hides or displays the buttons.
	Statusbar	—	Hides or displays the status bar at the bottom on the window.
	Start/End measurements	—	Starts or stops reading the EMG data, as above.
	Train Pointer	—	Trains the pointer as described above.
	Performance Test	—	Starts the performance test which is described in Section 5.2.3. There is no button which performs this function. this is to avoid problem with subjects accidently pressing it during an experiment.
Help	About EMG Pointer	—	Pops up the information window as above.

H.2.3 Hotkeys

There are three hotkeys used by the application. Hotkeys are necessary since the application takes control over the pointer, and there must be a way to operate the commands with just keyboard commands.

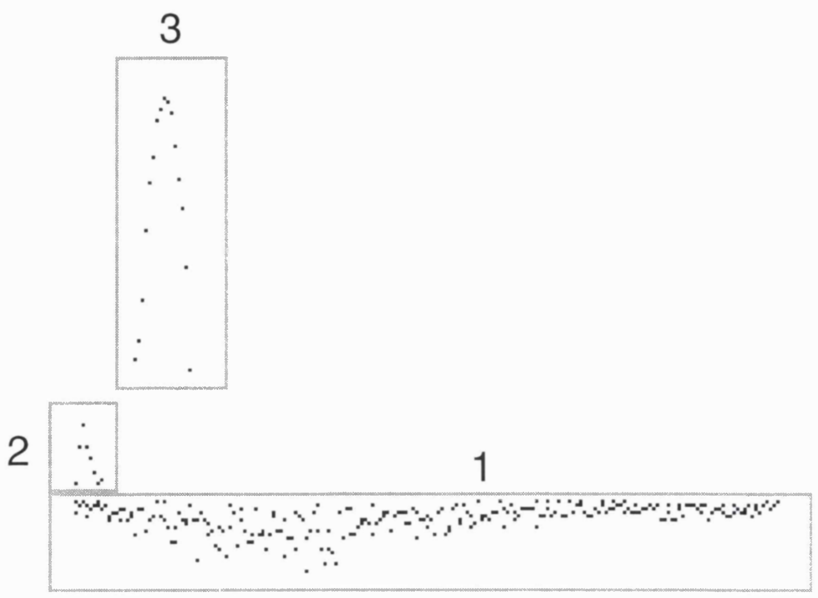
`<Control>-P` Start the performance test

`<Control>-S` Start / Stop reading EMG data

`<Control>-T` Start /Stop training

H.2.4 Main Display Area

Figure H.2: The real-time displayed plot for one EMG signal



The display contains the output of all four EMG amplifiers. Each of the signals is displayed in three ways, as shown in Figure H.2. These are all used for diagnostic purposes, and have no effect on the running of the application. These are displayed primarily to let the operator know if anything has gone wrong. Problems such as broken equipment, loose electrodes, weak batteries, and the like are fairly evident just by observing the display.

The first graph is a plot of the entire power spectrum. This is generated by the square of the absolute value of the fast fourier transform (FFT) of the EMG signal. This is displayed downward so as to not interfere with the other two graphs. This is used to determine the strength of the signal, which is especially useful when attaching the electrodes.

The second graph is the spectrum, rebinned down to 8 points. These are the values fed into the neural network.

The third graph is a plot of the integral of the spectrum over time. The 16 values correspond to the last 1.024 seconds. This plot shows that the wrist was recently extended and then relaxed. The horizontal and vertical offset is to enable the plot to be clearly differentiated from the others.

H.2.5 Status Bar

The status bar displays the current state of the application to the user. In the lower right it displays the shift states of the device (this has no effect on the application). The lower left displays text information to the user such as the purpose of a button or comments like "Pointer training will start immediately".

H.3 Internal Processes

When the EMG reading is started, a regular timer event is registered. This tells the window manager to call the `TimeProc` function every millisecond. This function reads the ADC and stores the value in an array. In addition, every 64th call, a copy is made of the entire array, and a message is sent to the operating system to tell it to update the pointer. Updating is done in a separate procedure with a copy of the raw data to allow the millisecond timing to run in parallel without any interference .

When the window manager receives the message to update the pointer it calls the `showData` function. This function calculates the square of the absolute value of the complex FFT of the data. The DC components of the power spectra are removed to avoid problems with signal drift. The spectra are then rebinned down to 8 points each. The full spectra, rebinned spectra, and the sum over each spectrum are then displayed in the main window.

If the pointer is being trained, the position and the 4 reduced spectra are saved for later calculations. If the pointer has been trained successfully, the 4 reduced spectra are fed into the neural network to receive the raw (x, y) position. This value is scaled by Equation 3.9 and added to the current position of the pointer.

This process is repeated until the data reading is turned off.

References

3M Health Care Customer Helpline (1997), Personal communication. Phone number: +1-800-228-3957.

Abernethy, C. & Hodes, D. (1987), "Ergonomically determined pointing device (mouse) design", *Behaviour and Information Technology* **6**(3), 311–314.

Andrews, D. (1994), "Apple, Cirque Unveil Trackball Alternative", *Byte* .

Balakrishnan, R. & MacKenzie, I. S. (1997), Performance Differences in the Fingers, Wrist, and Forearm in Computer Input Control, *in* "Proceedings of the CHI '97 Conference on Human Factors in Computing Systems", SIGCHI, New York: ACM, pp.303–310.

Bass, L. (1996a), "Boeing Wearable Computer Workshop: Human Computer Interaction Breakout Session", Internet document. Online at: http://www.cs.cmu.edu/People/wearable/boeing/hci.html.

Bass, L. (1996b), Is there a wearable computer in your future?, *in* "Engineering for Human computer interaction / Proceedings of the IFIP working conference", Vol. VIII, pp.3–16.

British Standards (1993), "BS5724-1 Medical Electrical Equipment, Part 1: General Requirements for safety, 1. Collateral standard: Safety requirements for medical electrical systems".

Burdea, G. & Coiffet, P. (1994), *Virtual Reality Technology*, John Wiley & Sons, chapter Chapter 2: Virtual Reality Tools, pp.15–79.

Buxton, W. (1995), Chapter 7: Touch, Gesture & Marking, *in* R. M. Baecker, J. Grudin, W. Buxton & S. Greenberg (eds.), "Readings in Human Computer Interaction: Toward the Year 2000", Morgan Kaufmann Publishers, San Francsco.

Buxton, W. & Myers, B. A. (1986), A study in two-handed input, *in* "Proceedings of CHI '86", pp.321–326.

Card, S., English, W. & Burr, B. (1978), "Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT", *Ergonomics* **21**, 601–613.

Carroll, D. (1984), *Biofeedback in Practice*, Longman.

Chang, G.-C., Kang, W.-J., Luh, J.-J., Cheng, C.-K., Lai, J. S., Chen, J.-J. J. & Kuo, T.-S. (1996), "Real time implementation of electromyogram pattern recognition as a control command of man-machine interface", *Medical Engineering Physics* **18**(7), 529–537.

Charlton, R. (1997), Mobile computing 1997 Worldwide Forecast Update, Market Statistics, Dataquest. Dataquest Research Report.

Cioni, R., Giannini, F., Paradiso, C., Battistini, N., Denoth, F., Navona, C. & Starita, A. (1988), "Differences between surface EMG in male and female subjects evidenced by automatic analysis", *Electroencephalography and Clinical Neurophysiology* **70**, 306–312.

Denning, D. E. (1982), *Cryptography and Data Security*, Addison-Welsey.

Dragon Systems (1998), "Dragon - Naturally Speaking Personal", Internet document. Online at: http://www.dragonsys.com/products/nat-speaking.html.

Epps, B. (1986), Comparison of six cursor control devices based on Fitts' law models, *in* "Proceedings of the Human Factors Society 30th Annual Meeting", pp.327–331.

Fels, S. & Hinton, G. (1993), "Glove-Talk: A Neural Network Interface Between a Data-Glove and a Speech Synthesiser", *IEEE Transactions on Neural Networks* **4**(1), 2–8.

Fels, S. & Hinton, G. (1995), Glove-TalkII: An adaptive Gesture-to-formant interface, *in* "Proceedings of Computer Human Interaction 1995 (SIGCHI95)", ACM, Denver, CO, pp.456–463.

Foley, J. D., van Dam, A., Feiner, S. K. & Hughes, J. F. (1990), *Computer Graphics, Principles and Practice, Second Edition*, Addison-Wesley, Reading, Massachusetts.

Freeman, J. A. & Skapura, D. M. (1991), *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley.

Geddes, L. A. (1972), *Electrodes and the Measurement of Bioelectric Events*, John Wiley & Sons.

Gips, J. & Oliveri, P. (1996), EagleEyes: An eye control system for people with disabilities, *in* "The Eleventh International Conference on Technology and Persons with Disabilities", Boston College, Los Angeles.

Goldberg, D. & Richardson, C. (1993), Touch-typing with a stylus, *in* "INTERCHI'93 Conference on Human Factors in Computer Systems", ACM, New York, pp.80–86.

Gopher, D. & Raij, D. (1988), "Typing With a Two-Hand Chord keyboard: Will the QWERTY Become Obsolete?", *IEE Transactions on Systems, Man, and Cybernetics* **18**(4), 601–609.

Greenstein, J. S. & Arnaut, L. Y. (1988), Input Devices, *in* M. Helander (ed.), "Handbook of Human-Computer Interaction", Elsevier Science Publishers B.V. (North-Holland), New York, NY, chapter 22, pp.495–519.

Grimes, G. (1983), "Digital Data Entry Glove Interface Device", Patent: US 4,414,537.

Guggenbuehl, U. & Krueger, H. (1991), Is Feedback Necessary when Using a Keyboard?, *in* M. J. Smith & G. Salvendy (eds.), "Proceedings of the Third International Conference on Human-Computer Interaction", Vol. 1 of *Work with Computers: Organizational, Management, Stress and Health Aspects;Interface – Displays and Controls*, NEC Corporation, Elsevier Science Publishers B.V. (North-Holland), New York, NY, pp.113–117.

Hartwig, H. (1978), "Tastur mit Kontakthandschuh zur manuellen Eingabe von Zeichen bei elektrischen bzw. elektronishen Gerüten", Patent: DE 27 18 415.

Hertz, J., Krogh, A. & Palmer, R. G. (1991), *Introduction to the Theory of Neural Computing*, Vol. 1 of *Computation and Neural Systems*, Addison-Wesley, Redwood City, CA, USA.

Hiraiwa, A., Uchida, N. & Shimohara, K. (1993), EMG Pattern Recognition by Neural Networks for Prosthetic Finger Control, *in* "Selected Papers from the IFAC/IFIP/IMACS Symposium 1993", pp.73–79. Artificial Intelligence in Real-Time Control 1992.

Holands, R. (1996), *The Virtual Reality Homebrewer's Handbook*, John Wiley & Sons, England, chapter 6: Data Gloves, pp.200–211.

Hunt, A. (1997), ""Java Speech API: A White Paper", Internet document. Online at: http://java.sun.com/marketing/collateral/speech.html.

I. Scott MacKenzie, A. S. & Buxton, W. (1991), A comparison of input devices in elemental pointing and dragging tasks, *in* "Proceedings of the CHI '91 Conference on Human Factors in Computing Systems", SIGCHI, New York: ACM, pp.161–166.

Ilg, R. (1987), "Ergonomic Keyboard Design", *Behaviour and Information Technology* **6**(3), 303–309.

International Organization for Standardization (1994), "ISO 9995-3 Information Technology - Keyboard Layouts for Text and Office Systems - Complementary Layouts of the Alphanumeric Zone of the Alphanumeric Section".

Jastrzembski, M. (1997), "Wearable Computer Systems at Carnegie Mellon University", Internet document. Online at: http://www.cs.cmu.edu/People/wearable/.

Johnston, R. D. (1996), "Are speech recognisers still 98% accurate, or has the time come to repeal 'Hyde's law?"', *British Telecom Technology Journal* **14**(1), 165–176.

Kabbash, P., Buxton, W. & Sellen, A. (1994), Two-Handed Input in a Compound Task, *in* "Proceedings of CHI '94", pp.417–423.

Kabbash, P., MacKenzie, I. S. & Buxton, W. (1993), Human performance using computer input devices in the preferred and non-preferred hands, *in* "Proceedings of InterCHI '93", pp.474–481.

Kermani, M. Z. & Badie, K. (1990), An Intelligent Strategy for Motion Interpretation in an EMG-Controlled Prosthesis: An Application of AI to Biomedical Engineering, *in* F. Gardin & G. Mauri (eds.), "Computational Intelligence, II Proceedings of the International Symposium 'Computational Intelligence 89"', Biomedical Engineering lab. Amirkabir University of Technology. Tehran, Iran, Elsevier Science Publishers B.V. (North-Holland).

Kirschenbaum, A., Friedman, Z. & Melnik, A. (1986), "Performance of Disabled People on a Chordic Keyboard", *Human Factors* **28**(2), 187–194.

Konheim, A. G. (1981), *Cryptogyraphy, a Primer*, John Wiley & Sons.

Kramer, J., Lindener, P. & George, W. (1991), "Communication System for Deaf, Deaf-Blind, or Non-Vocal Individuals Using Instrumented Glove", Patent: US 5,047,952.

Kroemer, K. N. E. (1972), "Human Engineering the Keyboard", *Human Factors* **14**(1), 51–63.

Lusted, H. S. & Knapp, R. B. (1996), "Controlling computers with bioelectric signals", *Scientific American* .

MacKenzie, I. S. & Chang, L. (1997), A performance comparison of two handwriting recognizers, Sumbitted for publication. Online at http://www.cis.uoguelph.ca/ mac/IWC2.html.

MacKenzie, I. S. & Oniszczak, A. (1998), A Comparison of Three Selection Techniques for Touchpads, *in* "Proceedings of the CHI '98 Conference on Human Factors in Computing systems", University of Guelph, ACM, New York, NY.

MacKenzie, I. S. & Zhang, S. (1997), The Immediate Usability of Graffiti, *in* "Proceedings of Graphics Interface '97", Canadian Information Processing Society, Toronto, pp.129–137.

MacKenzie, I. S., Soukoreff, R. W. & Zhang, S. (1997), Text entry using soft keyboards, Sumbitted for publication.

MacKenzie, S. (1992), "Fitts' Law as a Research and Design Tool in human-computer Interaction", *Human-Computer Interaction* **7**(1), 91–139.

Matias, E., MacKenzie, I. S. & Buxton, W. (1996), A wearable computer for use in microgravity space and other non-desktop environments, *in* "Companion of the CHI '96 Conference on Human Factors in Computing Systems", SIGCHI, ACM, New York, pp.69–70.

McCall, T. (1997), "Dataquest says half-year results show worldwide handheld market is no longer a pocket-sized industry: 3Com's PalmPilot Extends its market share in standard handheld market", Press Release.

Miah, T., Carter, C., Thorpe, A., Baldwin, A. & Ashby, S. (1998), "Wearable computers – an application of BT's mobile video system for the construction industry", *British Telecom Technology Journal* **16**(1), 191–199.

Microsoft Corporation (1995), "BAT IBM-PC Quick Reference Guide", Manual.

Microsoft Corporation (1997), "Introduction to the Published Research on the Microsoft Natural Keyboard". Online at http://www.eu.microsoft.com/products/hardware/netkeybd/intro.htm.

Mithal, A. K. & Douglas, S. A. (1996), Differences in movement Microstructure of the mouse and the finger-controlled isometric joystick, *in* "CHI '96 Proceedings: Conference on Human Factors in Computing Systems: Common Ground", SIGCHI, ACM, Vancouver, BC, Canada.

Moore, K. L. (1985), *Clinically Oriented Anatomy*, 2nd edition, Williams & Wilkins, Baltimore, MD, USA.

Noyes, J. (1983), "The QWERTY Keyboard: A Review", *International Journal of Man-Machine Studies* **18**(3), 265–281.

Palastanga, N., Field, D. & Soames, R. (1990), *Anatomy and Human Movement*, 2nd edition, Butterworth-Heinemann Ltd, Oxford, UK.

Pico Technology LTD (n.d.), *Pico ADC-11 User Manual*. Version 2.2.

Potosnak, K. M. (1988), Keys and Keyboards, *in* M. Helander (ed.), "Handbook of Human-Computer Interaction", Elsevier Science Publishers B.V. (North-Holland), New York, NY, chapter 21, pp.475–494.

Press, W., Teukolsky, S., Vetterling, W. & Flannery, B. (1992), *Numerical Recipies in C: The Art of Scientific Computing*, Cambridge University Press, chapter 2: Solution of Linear Algebraic Equations.

Rempel, D., Honan, M., Serina, E. & Tal, R. (1996), Wrist Postures While Typing On A Standard And Split Keyboard, *in* A. C. Bittner & P. C. Champney (eds.), "Advances in Industrial Ergonomics and Safety VII", Ergonomics Laboratory, University of California, Taylor & Francis, San Francisco.

Roberts, S. (1995), "Chord Keyboards", *Microship Status* (80). Online at http://www.microship.com/gopher-data/Microship_Status/Microship_Status_950131.

Rosenberg, R. (1997), "Keyboard Glove", Patent: GB 2 305 714 A.

Roy, D., Sawhney, N., Schmandt, C. & Pentland, A. (1997), Wearable Audio Computing: A Survey of Interaction Techniques, Technical Report 434, Perceptual Computing.

Scahill, F., Talintyre, J. E., Johnson, S. H., Bass, A. E., Lear, J. A., Franklin, D. J. & Lee, P. R. (1996), "Speech recognition – making it work for real", *British Telecom Technology Journal* **14**(1), 151–164.

Sears, A., Revis, D., Swatski, J., Crittenden, R. & Shneiderman, B. (1993), "Investigating Touchscreen Typing: The Effect of Keyboard Size on Typing Speed", *Behaviour and Information Technology* **12**(1), 17–22.

Seberry, J. & Pieprzyk, J. (1988), *Cryptography: An Introduction to Computer Security*, Advances in computer science, Prentice Hall.

Seibel, R. (1962), "Performance on a Five Finger Chord Keyboard", *Journal of Applied Psychology* **46**(3), 165–169.

Siegal, J. & Bauer, M. (1997), A Feild Usability Evaluation of a Wearable System, *in* "First International Symposium on Wearable Computers", Institute of Electrical and Electronics Engineers.

Soukoreff, R. W. & MacKenzie, I. S. (1995), "Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard", *Behaviour and Information Technology* **14**(6), 370–379.

Thompson, R. F. & Patterson, M. M. (1974), *Part B: Electroencephalography and Human Brain Potentials*, Vol. 1: Bioelectric Recording Techniques of *Methods in Physiological Psychology*, John Wiley & Sons.

Tittiranonda, P., Burastero, S., Wei, E. & Rempel, D. (1996), Three Month Clinical Prospective Intervention Study Using Four Keyboards, *in* "Marconi Input Device Research Conference", Interdisciplinary Ergonomics Program, Lawrence Livermore National Laboratory.

US Robotics (1996), "The Palm Pilot Connected Organizer", White Paper. Online at http://palmpilot.3com.com/products/ppwhitep.pdf.

Virtual Technologies Inc (1997), "VTI FAQ: GesturePlus^{TM}", Internet FAQ. Online at http://www.virtex.com/FAQ_gestureplus.html.

Yankelovich, N. (1994), Talking vs. Taking: Speech Access to Remote Computers, *in* "CHI '94 Conference Companion, 1994 ACM Conference on Human Factors in Computing Systems", ACM, Boston MA, pp.275–276.

Zimmerman, T. & Lanier, J. (1987), "Computer Data Entry and Manipulation Apparatus and Method", Patent: EU 0 211 984 A1.