Predictive Text-Entry in Immersive Environments

Barry McCaul

Bachelor of Science in Computer Applications

A dissertation submitted in fulfilment of the

requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

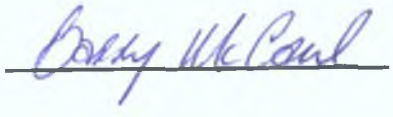
School of School of Computing

Supervisor: Dr. Alistair Sutherland

January, 2005

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed 

Student ID 95619054

Date January, 2005

Acknowledgments

This thesis could not have been completed without the support of many people, whom I am indebted to for their constant patience and encouragement throughout the past few years

I sincerely thank my supervisor Dr Alistair Sutherland for his support and guidance

To my parents, who stoically accepted my return home to student life Mam, you can have your car back now

To my fellow postgrad students, and those lucky few who have escaped before me Dave, Noel, Neil, Dalen, Mary, Mick, Niall, Michelle, John, and Karl to name but a few! Never before has so much coffee been drank by so few

To John, without whose encouragement I would never have reached this stage Thank you so much

To Roy, Brian, Pamela, Rory, and Celine who ensured that my time away from college, albeit fleeting, was very well spent

Finally, to Claire, whose patience, understanding, and love never ceased to astound me I love you, and I hope I can return the favour this year

Abstract

Virtual Reality (VR) has progressed significantly since its conception, enabling previously impossible applications such as virtual prototyping, telepresence, and augmented reality However, text-entry remains a difficult problem for immersive environments (Bowman *et al* , 2001b, Mine *et al* , 1997) Wearing a head-mounted display (HMD) and datagloves affords a wealth of new interaction techniques However, users no longer have access to traditional input devices such as a keyboard Although VR allows for more natural interfaces, there is still a need for simple, yet effective, data-entry techniques Examples include communicating in a collaborative environment, accessing system commands, or leaving an annotation for a designer in an architectural walkthrough (Bowman *et al* , 2001b)

This thesis presents the design, implementation, and evaluation of a predictive text-entry technique for immersive environments which combines 5DT datagloves, a graphically represented keyboard, and a predictive spelling paradigm It evaluates the fundamental factors affecting the use of such a technique These include keyboard layout, prediction accuracy, gesture recognition, and interaction techniques Finally, it details the results of user experiments, and provides a set of recommendations for the future use of such a technique in immersive environments

Contents

List of Figures

List of Tables

Chapter 1

Text-entry in immersive environments

1.1 Introduction and motivation

Text-entry in immersive environments has received minimal research in comparison to more traditional interaction techniques such as selection, manipulation and travel. There are two central reasons for this: firstly, virtual reality (VR) traditionally offers *natural* interaction with objects (Rather then issuing a command to delete a virtual object, a user simply picks it up and puts it in the virtual bin), secondly, usable and effective text-entry techniques are difficult to design and implement (Bowman *et al.*, 2004). Nevertheless, we believe that text-entry is an important feature which is significant for wider adoption of VR as a technology. Command-line entry remains the user interface technique of choice for computer *power users* (Jacob, 2000), allowing users to express complex possibilities rapidly and offering significant task efficiency through the use of scripting. Similarly, in VR, typed commands offer the possibility for powerful interaction, through the use of natural language interfaces (Kelleher, 2003). Aside from issuing system commands, there are other scenarios where text-entry is valuable in virtual environments. Collaborative work is a typical activity performed in

VR Communication is possible through speech However, the ability to take notes within the environment offers users persistent data which is less likely to be forgotten after a conversation has ended Similarly, users touring virtual architectural walkthroughs can annotate their environment, leaving details about what changes should be made without having to disengage from the environment

The following chapter reviews the previous text-entry research in immersive environments It proposes an alternative text-entry technique, and details its design and implementation

1.2 Text-entry techniques in immersive environments

Previous techniques for text-entry in immersive environments are varied They include speech, chorded keyboards, pen and tablet techniques, and various gloved techniques Each tries to overcome the inherent difficulty a designer faces when trying to design a text-input technique for a system without access to a standard keyboard

1 2 1 Speech

Speech input offers what might seem like the most ideal solution to the natural interface offered by VR and was first used by Bolt (1980) in his famous "Put That There" system over 20 years ago However, although considerable advances have been made in speech recognition in the intervening years – with vocabulary sizes increasing from 1000 words to 230,000 (Keenan, 2002) – speech is not ideal for several reasons Typically, speech recognition systems require significant training for accurate dictation Even with extensive training, they suffer in noisy environments, and are inefficient for text-editing and manipulation (Schneiderman, 2000) Despite 20 years worth of advances, speech input is not commonly used for 2D user interfaces, as users are sensitive to the lack of privacy, the perception of bothering others around them, and feel an unease or awkwardness talking to

machines (Bowman *et al*, 2004)

Although many of these problems will no doubt be addressed, with future systems being user-independent, requiring little or no training, a more fundamental problem exists which was recognised by Schneiderman (2000) His experiments found that users had difficulty in completing tasks which required both the use of speech and memory In a word processing experiment, users were required to memorise an equation, issue a voice command to "page down", and then type the memorised equation Schneiderman found that users repeatedly scrolled back to review the equation, as speaking the commands appeared to interfere with their retention Schneiderman argued that this was because both activities required the use of limited short-term or acoustic memory Therefore, because physical activity is handled by another part of the brain, it is easier to think and type, than it is to think and talk

1 2 2 Pen and tablet techniques

The Virtual Notepad (Poupyrev *et al*, 1998a) draws from the ubiquitous everyday activity of writing It combines a spatially tracked pressure-sensitive graphics tablet, pen, and handwriting recognition software Users are able to take notes and annotate documents without being forced to disengage from VR in order to use a keyboard The system is activated by bringing the pen close to the tablet, whereby the user is presented with a virtual notepad and pen As the user writes on the physical tablet, the virtual pen writes on the virtual notepad Handwriting is both simple and intuitive, thus users require very little time to adjust to the system However, the system suffers from latency problems, and so there is considerable lag between the user writing and the visual representation on the notepad

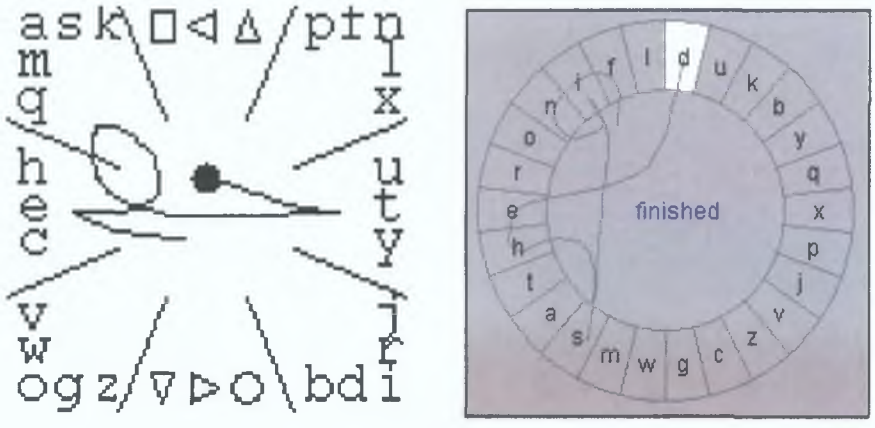
With handwriting recognition disabled, the Virtual Notepad produces *digital ink*, allowing users to take notes, which can be read at a later stage However, the digital ink is not machine readable, which means that it cannot be used to

Figure 1.1: Virtual Notepad (From Poupyrev *et al.*, 1998a)

issue commands, or be searched at a later date. Alternatively, the system can be used with recognition enabled. However, character recognition is typically inaccurate unless specialised character sets such as *Graffiti* (PalmOne Inc., 2004) are employed. While accurate, these slow text-entry, requiring users to learn a specialised alphabet which must be entered separately, one character at a time.

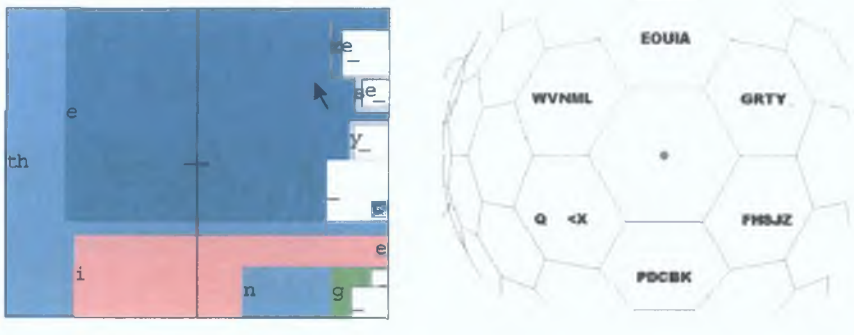
The use of a pen and tablet in VR allows for many techniques, which, while designed for use on small mobile devices, are nevertheless equally applicable to VR. Such techniques include the standard soft virtual keyboards, found on most PDAs, as well as continuous stroke techniques such as Quikwriting (Perlin, 1998) and Cirrin (Mankoff and Abowd, 1998) (Figure 1.2a and 1.2b). Adaptive techniques, such as Dasher (Ward *et al.*, 2000) and Hex (Williamson and Murray-Smith, 2003), which use probabilistic data based on linguistic models to alter the user interface depending on text typed are also suitable candidates for pen and tablet interfaces (Figure 1.3a and 1.3b).

The largest drawback of pen and tablet techniques is that in order to see the virtual tablet users must hold it up in front of them, which is tiring with prolonged use. The users are also forced to have a pen and tablet constantly in



(a) (b)

Figure 1.2: Quikwriting (a) and Cirrin (b) (From Perlin, 1998 and Mankoff and Abowd, 1998)



(a) (b)

Figure 1.3: Dasher (a) and Hex (b) (From Ward *et al.*, 2000 and Williamson and Murray-Smith, 2003)

their hand, which limits the number of alternative interaction methods they can avail of within the system

1 2 3 Chorded keyboards

Chorded keyboards such as the TwiddlerII (2004) allow a user to enter text with just one hand Their reduced size, allowing them to fit comfortably in one hand, necessitates the use of fewer physical keys the TwiddlerII has just 18 keys However, by pressing several of the 18 buttons simultaneously users can emulate the 101 keys on a standard keyboard Although expert users can type at speeds of up to 60 WPM, beginners are forced to learn a complicated alphabet of chords before they begin to type As with the pen and tablet technique, most chording keyboards require a user to carry an extra device Exceptions to this, Thumbcode (Pratt, 1998) and Chording Glove (Rosenberg, 1998) offer similar text-entry techniques with the buttons incorporated into a glove which is worn by the user

1 2 4 Gloved techniques

Datagloves are commonly used for interaction in immersive environments, allowing the wearer to interact in a natural way with virtual objects A text-entry technique which uses datagloves is thus likely to be easily incorporated into any immersive environment

Sign language is one option for gloved data entry Grimes (1983) secured a patent on a dataglove which was described as a man-machine interface "for translating discrete hand positions into electrical signals representing alpha-numeric characters" Grimes's "Digital Data Entry Glove" incorporated sensors which measured both finger flexion and contacts at key positions as well as hand orientation The glove was never put into actual use, or made commercially available (Sturman and Zeltzer, 1994) Furthermore, gesture recognition was hard-coded into Grimes's glove in order to recognise sign language Other more flexible sign

language recognition techniques have been developed since Grimes's Digital Data Entry Glove Krammer's *Talking Glove* project (Kramer and Leifer, 1989), was designed to convert American Sign Language finger spelling into synthesised voice with a early version of the CyberGlove Kadous (1995) used a Mattel PowerGlove to recognise 95 different Australian Sign Language signs with 80 percent accuracy Finally, Fels and Hinton (1993) created a system which recognised 66 root words with 5 possible suffixes Although these systems were not designed specifically for immersive environments, they do allow for the conversion of hand motion into text However, each of these methods has the same fundamental problem, each requires significant training, as a signing language must be learnt before it can be used

An alternative to learning a new signing language is to draw from the most common data-entry device, the keyboard Kitty (Mehring *et al* , 2004) attempts to transfer the information gained using standard keyboards to gloves by mapping contact points on the gloves to a regular QWERTY keyboard layout To type, fingers are pressed against the various contact points on the thumb, which indicate the desired row Thus, the letters *Q*, *A*, and *Z*, which are normally struck by the little finger on a regular keyboard, are *typed* on the glove by pressing the little finger on three contact points on the thumb which correspond to the top, middle, and bottom row of the keyboard respectively However, with no visual representation of the keyboard offered to users, the system is only suitable for touch-typists The Finger-Joint Gesture (FJG) glove (Goldstein and Chincolle, 1999) uses a similar technique to Thumbcode (Pratt, 1998), with three keys on each fingers which are pressed by the thumb Although designed primarily for numeric entry, the FJG design emulates the keyboard of a mobile phone and thus could be used to mimic any of the text-entry methods used on modern mobile phones

A similar technique to Kitty, but one which provides better visual feedback, is the pinch keyboard (Bowman *et al* , 2001b) It uses FakeSpace Pinch Gloves

combined with two six degree-of-freedom (DOF) trackers, with a visual representation of a keyboard also provided Based on the ubiquitous QWERTY keyboard layout that users are familiar with, the system requires users to pinch the thumb and any finger to represent a key press for that finger Thus a pinch between thumb and index of the left hand would correspond to F if the home row were active Active rows (top, home and bottom) are selected by moving the hands closer or further away from the user's body Inner keys (G & H on home row) are selected by rotating the hand inward Special gestures are provided for space, delete, etc Visual feedback is provided which displays the currently active keys Although easy to use, typing was slow for beginners, 3 minutes for sentences of 6-8 words With extensive practice the designers of the system were able to reduce this to 45 seconds or roughly 12 to 15 words-per-minute (WPM) Interestingly, although most users tested were comfortable touch-typists, there was little transfer of this knowledge to the pinch keyboard and considerable time was spent searching for the correct key

Finally, Evans *et al* (1999) suggested an alternative virtual keyboard technique, VType Rather then using 3D trackers to indicate the desired row, VType mapped the fingers of two 5DT datagloves to the keys they would strike on a QWERTY keyboard, and used a disambiguation algorithm to predict the intended keys at the end of each sentence We believe the predictive text-entry in immersive environments, suggested by Evans *et al* (1999), although showing promise, is limited in its current form VType is designed for use in immersive environments However, in practice, keys are indicated visually by attaching paper keys to the gloves VType provides no method for the correction of incorrect predictions, nor any method for correcting human typing errors Instead, Evans *et al* focus on the readability of sentences that contained errors Our work builds upon the technique outlined by Evans *et al* It explores the practical issues in using such a system and attempts to augment VType in several key areas

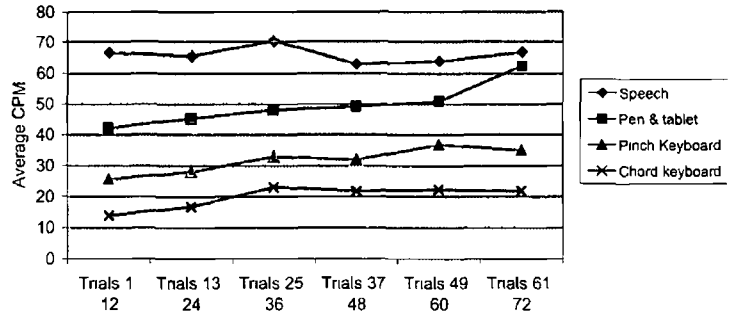


Figure 1 4 Comparison of CPM for four immersive text-entry techniques speech, a chording keyboard, a pinch keyboard and a soft keyboard (From Bowman *et al* , 2002)

1 2 5 Empirical comparisons of immersive text-entry techniques

Bowman *et al* (2002) compared four immersive text-entry techniques speech, a chording keyboard, a pinch keyboard, and a soft keyboard Voice input was achieved using a *wizard of oz* technique, users spoke one letter at a time into a microphone, and a hidden evaluator listened to the users utterances and typed the correct letter on a keyboard A TwiddlerII (2004) was used for the chording keyboard The soft keyboard was implemented with a tracked pen and tablet The experiment measured both the text-entry rate achieved with each technique, as well as the subject comfort experienced using each technique

The results, which measured characters-per-minute (CPM), showed speech to be the most efficient technique, followed by the soft keyboard and the pinch keyboard, with the chording keyboard proving the slowest of all techniques (Figure 1 4) However, based on subjective ratings by users, no technique offered clear advantages Although fastest, speech seemed tedious The soft keyboard resulted in a high degree of neck and arm strain (Figure 1 5) Finally, the TwiddlerII was considered the least appropriate for immersive environments However, this was most likely due to the fact that users were unfamiliar with the required chords

Thomas *et al* (1997) evaluated three text-entry techniques for wearable computing a Kordic keyboard, a forearm keyboard, and a soft keyboard The Kordic

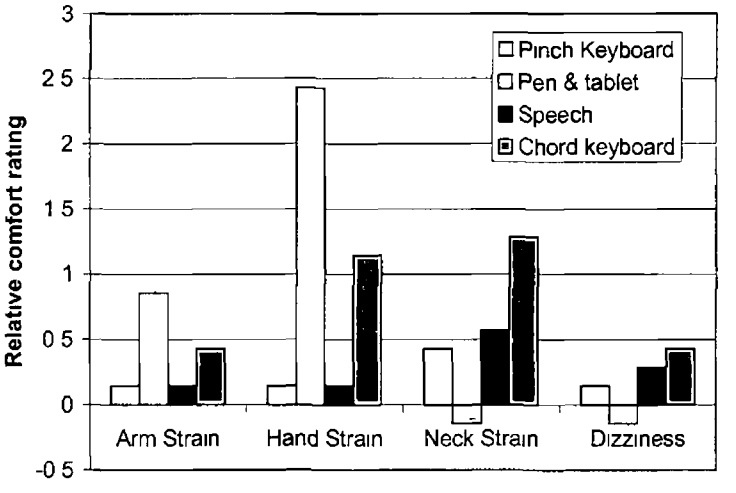


Figure 1 5 Comparison of comfort ratings for four immersive text-entry techniques speech, a chordmg keyboard, a pinch keyboard and a soft keyboard (From Bowman *et al* , 2002)

keyboard was a chord keyboard similar to the TwiddlerII (2004) The soft keyboard was accessed using a belt mounted mouse Finally, the forearm keyboard accessed using the users dominant hand, and was attached to the non-dominant arm While the test was designed to evaluated text-entry techniques for mobile, wearable computers, both the Kordic keyboard and soft keyboard could be used in conjunction with a HMD as they do not require direct sight to be used Although the forearm keyboard could potentially be used for touch-typing, where users did not look at the keys, it was not used in this manner for the experiments

Thomas's (1997) study found that the highest text-entry rate was achieved with the forearm keyboard This was followed by the soft keyboard, with the lowest speed being recorded with the Kordic keyboard (Figure 1 6) Steady gains were made over the course of the experiments with all techniques However, with over 5 hours training, the text-entry rate of the Kordic keyboard remained below 25 CPM (5 WPM) The speed of the soft keyboard was attributed to the poor performance of the belt mounted mouse, which users found tiring and uncomfortable to interact with

Finally, Osawa (2002) conducted an experiment to contrast speech, a 3D vir-

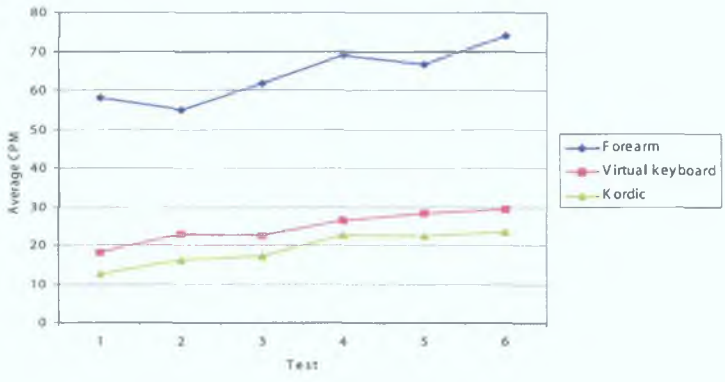


Figure 1.6: Comparison of CPM for three immersive text-entry techniques: a forearm mounted keyboard, a virtual keyboard, and a Kordic keyboard (From Thomas *et al.*, 1997).

(a) (b)

Figure 1.7: Virtual 3D keyboard (a) and Speech and gesture interface (b) (From Osawa and Sugimoto, 2002)

tual keyboard (Figure 1 7a), and a technique that combined of speech and gesture (Figure 1 7b) Osawa's 3D virtual keyboard widget was designed for interaction with a spatially-tracked dataglove Users pressed virtual keys in the same manner as they would on a standard keyboard Speech input was preformed using the recognition engine provided with Microsoft Office XP The combined speech and gesture technique, involved selecting correct utterances from predictions made by the system as a user spoke Suggested words were presented in 3D lists and could be ignored, selected or combined to create sentences

The results of the experiments showed no statistically significant effect of input technique on text-entry rate The virtual keyboard had the fastest average speed, followed by the speech and gesture, and finally the speech only technique A subjective questionnaire revealed that of the three, the speech and gesture, and virtual keyboard techniques were preferred, with the virtual keyboard deemed the most appropriate for precise text-entry in immersive environments

1 2 6 Review and discussion

Though desirable, no elegant solution exists for text-entry in immersive environments Speech is unsuitable for many of the same reasons it is not used as a 2D technique Chording techniques require considerable training, and are frustrating for beginners Pen and tablet techniques are fast, yet tiring with prolonged use, and limit alternative interaction possibilities Glove-based techniques, offer the most versatile solution They incorporate existing input hardware and do not require additional devices to be carried by the user The pinch keyboard (Bowman *et al*, 2001b) is perhaps the most elegant of these, and has many positive elements It offers users a visual representation of the keyboard, an intuitive interface, and is quickly learnt by beginners However, the large movements necessary for row selection, and the lack of muscle memory transfer, result in slow text-entry rates VType (Evans *et al*, 1999) provides an interaction style that is closer to the typing motion of a regular keyboard, and is thus more likely to

Figure 1.8: Proposed solution

allow muscle memory transfer. However, the lack of a virtual keyboard, and the ability to correct predictions, considerably limit its usefulness. An ideal gloved solution, is one which combines aspects of both techniques.

1.3 Proposed technique

We propose the following technique: when text-entry is required in a virtual environment the user is presented with a graphical representation of a keyboard, with each finger mapping to a column of keys. To type, the user simply flexes the relevant finger to select the corresponding column. After a sequence of finger flexes, a dictionary is consulted and the user is presented with the predicted word. Users may rotate through alternative matching words to indicate the desired word if the initial prediction is incorrect.

Having defined the basic technique, the crux of our research will focus on the main factors affecting its use: prediction accuracy, keyboard layouts, gesture recognition, and interaction techniques. Some of these areas are interdependent; the interaction techniques used will depend on the gestures which can be recognised; and the accuracy of the predictions will depend, as well as the keyboard layout employed. The design of the keyboard and the interaction techniques used

will depend on the gestures which we can recognise and the accuracy of the prediction system Finally, the prediction accuracy of the system will be depend on the language model used, but will also be affected by the keyboard layout employed

1 4 Design methodology

The design of our text-entry technique, follows the iterative evolution model common in many design methodologies, and is influenced by (Preece *et al* , 1993, Gabbard *et al* , 1999, Johnson, 1992, Bowman, 1999)

An initial evaluation identified keyboard-layout, prediction-accuracy, and interaction techniques as fundamental factors affecting the use of the system This was followed by a prototype design Through iterative, formative evaluation this evolving prototype helped to give a greater insight into the system requirements Informal *hallway* user testing identified user factors that simple task analysis might not have otherwise identified Gesture recognition difficulties became apparent as an essential factor affecting the use of the system Minor elements, such as the potential benefits of word completion were also examined at this stage These tests were used throughout the design process to verify or disprove potential improvements to the system From this iterative evolution, four key factors were identified as central to the use of predictive virtual keyboards prediction accuracy, keyboard layouts, gesture recognition, and interaction techniques These four fundamental factors are examined in greater detail in Chapters 3 through 6

Finally, the overall merit and usability of the system, and effects of various keyboard layouts, visual aids and interaction techniques are evaluated in a larger summative evaluation The results of this evaluation lead to quantitative performance results, as well as a set of guidelines for the future use of predictive text-entry

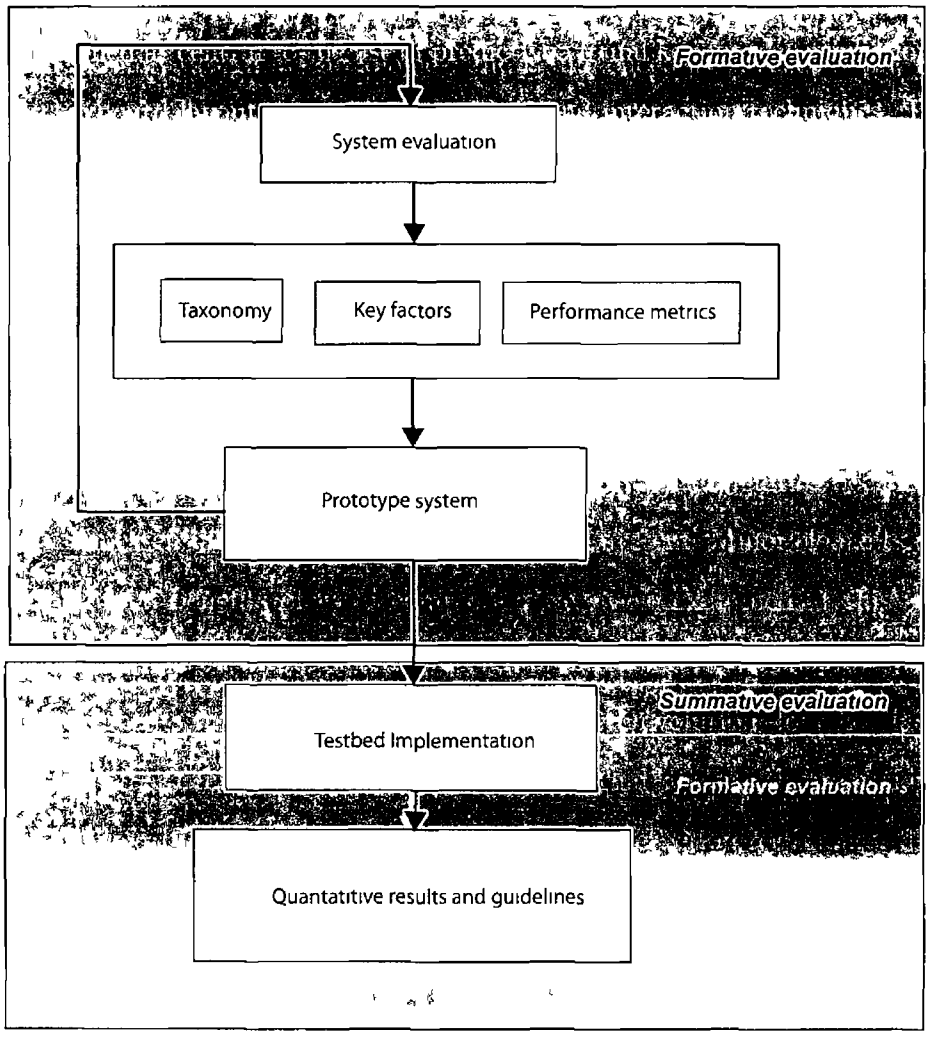


Figure 1 9 Design methodology

1.5 Thesis outline

In this chapter we have proposed a predictive text-entry technique for immersive environments which combines 5DT datagloves, a graphically represented keyboard, and a predictive spelling paradigm We have identified the fundamental factors affecting the use of such a technique keyboard layout, prediction accuracy, gesture recognition, and interaction techniques

In Chapter 2 we review previous work in the four core areas central to the use of predictive text-entry in immersive environments In Chapter 3 we will

examine the design of optimised keyboards for ambiguous text-entry, developing a keyboard designed to minimise ambiguity while typing Chapter 4 will explore the prediction accuracy of ambiguous keyboards, in particular we will examine the effects of language modelling on accuracy with standard and optimised keyboards In Chapter 5 we will tackle the problem of gesture recognition suitable for predictive text-entry We will contrast the gesture recognition techniques suitable for identifying key-press postures We will focus on the problem of sympathetic bending, and the associated recognition errors Chapter 6 will discuss the design of interaction techniques suitable for immersive text-entry with ambiguous keyboards Ultimately, any user interface must be subject to user testing, which can be used both for the identification of design problems, and to compare and contrast potential designs Chapter 7 discusses the results of both formative and summative evaluations conducted

Although examined separately in the following chapters, many of these core issues are interlinked Chapter 4, which focuses on prediction accuracy, will be influenced by the accuracy of the optimised keyboards created in Chapter 3 Interaction techniques, designed in Chapter 6, will be influenced by both the gesture recognition capabilities and prediction accuracy

Finally, Chapter 8 combines the results of Chapters 3 to 7 to provide a detailed methodology for predictive text-entry in immersive environments This methodology summarises the results of our experiments, and provides a set of recommendations for the use of ambiguous text-entry in immersive environments

Chapter 2

Predictive text-entry in immersive environments: theory and problems

2.1 Introduction

In this chapter, we review the four core areas central to the use of predictive text-entry in immersive environments. Section 2.2 will review prediction accuracy, and the techniques employed to resolve the inherent ambiguity caused by placing multiple letters on one key. Section 2.3 examines the design of keyboard layouts, the reasons for their optimisation, and techniques employed to do so. Section 2.4 discusses gesture recognition, the types of gestures relevant to virtual typing, and the relevant techniques used to recognise them. Finally, Section 2.5 considers interaction in immersive environments, focusing on those suitable for interaction with virtual keyboards.

2.2 Prediction accuracy of ambiguous keyboards

Shannon (1951) reports experiments to calculate the entropy of the English language Entropy is a measure of the average information content of an message source If plain text is optimally compressed into binary digits, the entropy of the language is the average number of binary digits required per letter of the original text (Ward, 2001) Shannon conducted experiments in which users were asked to predict the next letters of a sentence, only proceeding past each letter when they guessed correctly An example result looked as follows

$$T_1 H_1 E_1 R_5 E_{1\,-5}\, I_2 S_{1\,-1}\, N_2 O_{1\,-1}\, R_{15} E_1 V_{17} E_1 R_1 S_1 E_{2\,-1}\, O_3 N_{2\,-1}\, A_3$$

The subscript represents the number of guesses needed, and _ represents a space Through these experiments, he showed that the entropy of the English language was roughly 1 3 bits per character In doing so he demonstrated the redundancy which exists within the English language In an experiment, subjects were able to use this redundancy to predict the next letter with high accuracy, subjects were able to guess 79 of 102 letters on their first guess Language prediction systems attempt to harness this redundancy to aid them in accurate prediction during text-entry

2 2 1 Word prediction and word completion

A distinction is made at this point between the terms word *prediction* and word *completion* Word *completion* refers to the technique whereby complete words are offered to users as they type based on the text that they have entered until that point, thus potentially saving the user several key-stokes In systems with unambiguous keyboards, this technique is often also commonly referred to as word-*prediction* However, this can lead to confusion as word *prediction* is also the term used when text-entry is performed on an ambiguous keyboard As

the user types on an ambiguous keyboard there is an inherent uncertainty as to the intended letter or word typed The letter or word offered by the system is essentially a guess, or a prediction of the user's intention This is the meaning we refer to when discussing word prediction throughout this thesis Thus, *predicted* words refer to the current interpretation(s) of the ambiguous sequence typed by the user, while *complete* words refer to words offered by the system, while the user is typing, which are longer then the current *predicted* word

2 2 2 Ambiguous text-entry

The mapping of multiple letters to one key in order to reduce the number of keys that are needed is a widely used technique It was initially suggested independently by Glaser (1981), and Johnson and Hagstad (1981) as a communication aid for people with speech impairments It has since been adapted for text-entry in many conditions where a standard keyboard is not feasible or desirable

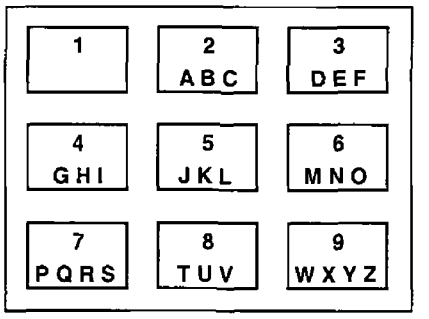


Figure 2 1 Telephone keypad

Mapping multiple letters to each key leads to an inherent ambiguity when each key is pressed There are various methods for resolving this ambiguity, most of which can be classed as either letter-level or word-level disambiguation methods

2 2 3 Letter-level disambiguation

Letter-level disambiguation typically requires the user to press 2 keys per letter Although there are several variations, all typically require 1 key-press to select a

group of letters, and a second key-press to uniquely identify the required letter For example, the letter *C* might be selected on the keyboard in Figure 2 1 by pressing the *2* key to select the letter grouping *ABC*, followed by the *3* key to indicate the 3rd letter in the grouping An alternative technique, often referred to as *Multitap*, requires users to scroll through the key groups to select the required letter, thus the letter *C* on the keyboard in Figure 2 1 would be selected by three presses of the *2* key

2 2 4 Word-level disambiguation

Word-level or dictionary-based disambiguation was originally suggested by Witten (1982) Due to the entropy of the English language, Witten found that if a dictionary of 24,500 words was encoded such that each letter in a word was represented by the key number on which is resided on a telephone keypad[1], only 2000 words had identical sequences Thus, the majority of words can be expressed unambiguously with only 1 key-press per letter when the sequence of keys entered is compared against a dictionary of valid words *Clashing* is used to describe the phenomenon of two or more words mapping to the identical sequence The *clash-count* for a dictionary refers to the number of words within the dictionary that clash for a given keyboard layout The practical use of word-level disambiguation requires that users be afforded the ability to choose between words if there is more then one match for a given set of key-presses One option for this, is to offer the most likely word to the user, with the option of choosing the next most likely alternative should the first prediction be incorrect Dictionary based disambiguation was suggested as a mobile phone SMS text-entry method by both Dunlop and Crossan (2000) and engineers at Tegic Communications (D L Grover and Kuschler, 1998) independently

[1]Witten assigned the missing letters Q & Z to the *1* key On modern phones they are assigned to the *7* and *9* keys respectively

2 2 5 Prediction accuracy of letter-level disambiguation systems

Letter-level disambiguation was improved by Foulds *et al* (1987), who used letter-frequency statistics to predict the desired letter, allowing the user to correct the prediction at any point Their system was similar to Multitap, but rather than rotate through letters in the order in which they appeared on the keys, Foulds *et al* suggested the use of fourth-order (quad-gram) transitional probabilities to disambiguate each keystroke based on the statistical relationship of the ambiguous characters to the previously typed words As a user presses a key, a quad-gram table is consulted, which provides the statistical probability of a character based on the 3 previously typed letters The characters on the pressed key are ranked and offered to the user according to there likelihood If the first character offered is incorrect, the user hits the key again and is offered the next most likely character This method was later suggested by MacKenzie *et al* (2001) as a text-entry technique for mobile phones *Letterwise* Although more efficient in terms of the average number of keystrokes required per character (KSPC), with a KSPC value of 1 15 compared to 2 0342 for Multitap, Letterwise requires constant attention during typing With Multitap, the letter *B* is always entered with two presses of the *2* key, requiring no user attention In comparison, with Letterwise the letter *B* may appear after 1, 2, or 3 key-presses depending on context, forcing users to pay close attention during typing

2 2 6 Prediction accuracy of word-level disambiguation systems

Unlike letter-level disambiguation, word-level disambiguation relies on a dictionary As the dictionary increases, so does the percentage of words known, but the chance of an error increases too However, like letter-level disambiguation, improvements in accuracy can be achieved

The use of word frequency is the most obvious method of increasing prediction accuracy, and is the method employed on most mobile phones (T9, 2004) Ana-

lysing the corpora – large collections of text and speech – from which dictionaries are created, we can store the frequencies with which words occur Then, when a sequence of key-presses matches more then one word in a given dictionary, the list of possible words offered are ranked according to their frequency

Although this improves accuracy, it fails to make full use of the contextual information available during typing Various techniques have been proposed to help augment the accuracy of word-level accuracy These can be broken down into 3 fundamental groups syntactical, statistical, and context-sensitive

Syntactical techniques

Syntactical techniques try to use syntax to help predict the most likely word Words in the current sentence are tagged with parts-of-speech (POS) tags These tags detail the lexical type of each word, differentiating them into classes such as noun, verb, pronoun, adverb etc Using this information, a parser then attempts to parse the current sentence based on a set grammar This grammar contains a set of rules, or productions, each of which expresses the ways that syntactic categories of the language can be grouped and ordered together (Jurafsky and Martin, 2000) Based on this information, ambiguous words can be ranked according to how well they would fit into the currently parsed sentence Work in this area is often aimed at word completion (Guenthner *et al*, 1993, Beck *et al*, 2004), where the systems are attempting to aid in the sorting of lists of potential complete words However, the techniques are equally applicable to word prediction Some of the problems faced when using syntactic techniques include trying to parse ungrammatical sentences, dealing with ambiguous words which have various possible tags, and the problem of tagging any new words entered into the dictionary Despite these problems, syntactic information is useful and is often used in combination with other techniques to augment the overall accuracy (Rua and Skiena, 1994, Hasan, 2003, Fazly and Hirst, 2003) Interestingly, as noted by Boissière and Vigouroux (2003), most of these projects are French or German,

which are highly inflected languages English, which is less inflected, is more suited to statistical techniques, such as N-gram language modelling

Statistical techniques

Language modelling is the term used to describe the use of statistical knowledge gained from analysing corpora to assign probabilities to words N-gram models use the previous $N-1$ words, to help predict the next, or N^{th}, word Thus, given an N-gram language model, the probability $P(w_n \mid h)$ of the N^{th} word w_n, given a history of words h can be written as $P(w_n \mid w_1, w_2 \quad w_{n-1})$ In the simplest case, that of a 1-gram or uni-gram model, the probability of a word is based solely on its frequency within a corpus For $N > 1$ there are various techniques which can be used The simplest of these being the *maximum likelihood estimation* (MLE) If we define $c(w_1 \quad w_n)$ to be the count of a specific N-gram in our corpus, then we can define the probability of a bi-gram (2-gram) as follows

$$P_{MLE}(w_n \mid w_{n-1}) = \frac{c(w_{n-1}w_n)}{c(w_{n-1})}$$

More generally, we can define the probability of for an N-gram model as

$$P_{MLE}(w_n \mid w_{n-N+1}^{n-1}) = \frac{c(w_{n-N+1}^{n-1}w_n)}{c(w_{n-N+1}^{n-1})}$$

Discounting Although simple, the MLE technique is not ideal As no corpus can contain every possible N-gram, the MLE technique overestimates the probability of the N-grams it has been trained on and underestimates the probability of the N-grams that were not seen in the training corpus To counter these effects, various discounting or smoothing algorithms can be used, including *Add-One*, *Witten-Bell* (Witten and Bell, 1991) and *Good-Turing* (Good, 1953) These attempt to lower the overall probability of N-grams which have been seen, in order to increase the probability of N-grams which haven't

One widely used discounting strategy, and the one used in our tests, is Good-Turing discounting This was first described by Good (1953), who credited Turing with the original idea The Good-Turing method attempts to re-estimate probabilities of N-grams with zero or low counts, by looking at N-grams with higher counts It defines N_c as the number of N-grams which occur c times This is referred to as the frequency of frequency c It defines c^* as the smoothed count of N-grams which occur c times, N_c , by looking at the count of N_{c+1}

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

Thus an estimate for the count of N-grams which never occurred N_0 is achieved by looking at the count of N-grams which occurred once, N_1 The same is done to estimate the count for all N-grams of low counts below a threshold k, above which the actual count value is considered accurate enough Thus, $c^* = c$ for $c > k$ The full equation (From Jurafsky and Martin, 2000) is written as follows

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, \text{for } 1 \leq c \leq k$$

Katz (1987) recommends a value of 5 for k Using this new estimation of N-gram counts, the discounted probability $\tilde{P}$ of a bi-gram occurring can now be written as

$$\tilde{P}(w_n \mid w_{n-1}) = \frac{c^*(w_{n-1}w_n)}{c(w_{n-1})}$$

Backoff Although discounting can be used to estimate the probabilities of N-grams we have not seen, another valuable source of information is lower-order N-grams If we have seen no occurrence of a specific tri-gram $w_{n-2}w_{n-1}w_n$ during training, then we can estimate its probability based on the probability of the bi-gram $w_{n-1}w_n$, $P(w_n \mid w_{n-1})$ If no occurrences of the bi-gram $w_{n-1}w_n$ have

been seen, then we can estimate the probability based simply on the probability of the uni-gram probability $P(w_n)$

The backoff N-gram model is a non-linear method introduced by Katz (1987) If we have a non-zero probability for an N-gram – because it has been seen before in our corpus – we use it to determine probability If we haven't seen an N-gram before, we *back off* to a lower-order N-gram Formally

$$\begin{aligned}\hat{P}(w_n \mid w_{n-N+1}^{n-1}) = \; & \tilde{P}(w_n \mid w_{n-N+1}^{n-1}) \\ & +\theta(P(w_n \mid w_{n-N+1}^{n-1}))\alpha(\hat{P}(w_n \mid w_{n-N+2}^{n-1})\end{aligned}$$

where

$$\theta(x) = \begin{cases} 1, \text{ if } x = 0 \\ 0, \text{ otherwise} \end{cases}$$

Here θ indicates the binary function whereby the lower-order model is chosen if the higher-order model has zero probability α is the normalisation factor, which is used to ensure that the lower-order model only receives a fraction of the remaining probability of the discounted higher-order model This is to ensure that the overall probability cannot be greater then 1 For an N-1-gram it is computed by subtracting from 1 the total discounted probability mass for all N-grams starting with that context, and is then distributed to all N-1-grams This is then normalised by the total probability of all N-1-grams that begin some N-gram Formally

$$\alpha(w_{n-N+1}^{n-1}) = \frac{1 - \Sigma_{w_n\, c(w_{n-N+1}^{n})>0}\tilde{P}(w_n \mid w_{n-N+1}^{n-1})}{1 - \Sigma_{w_n\, c(w_{n-N+1}^{n})>0}\tilde{P}(w_n \mid w_{n-N+2}^{n-1})}$$

The non-linear tri-gram model, which we shall use during the course of our experiments, can be represented as follows

$$\hat{P}(w_n \mid w_{n-2}w_{n-1}) = \begin{cases} \tilde{P}(w_n \mid w_{n-2}w_{n-1}), & \text{if } c(w_{n-2}w_{n-1}w_n) > 0 \\ \alpha(w_{n-2}^{n-1})\tilde{P}(w_n \mid w_{n-1}), & \text{if } c(w_{n-2}w_{n-1}w_n) = 0 \\ & \text{and } c(w_{n-1}w_n) > 0 \\ \alpha(w_{n-1})\tilde{P}(w_n), & \text{otherwise} \end{cases} \quad (2\,1)$$

Context-sensitive

As the name suggests, context-sensitive techniques attempt to use context to aid in the prediction of more suitable words (Hawes and Kelleher, 2004, Stocky *et al*, 2004, Guenthner *et al*, 1993) On a mobile phone, the words *act*, *cat* and *bat* have the same key-sequence, 228 Many mobile phones use the T9 prediction engine (T9, 2004), which uses a uni-gram language model Statistically, the word *act* occurs more often in the English language, therefore it is offered first Thus, if the sentence "Yesterday a stray dog chased my " is entered followed by the key-sequence 228, the word *act* is offered because it is most likely word statistically However, given the *context*, clearly *cat* is more likely to be the word intended by the user Context-sensitive systems, use databases of related words, which can be used to help re-order the offered words In the above example, the word *cat* would be related to both the words *chased* and *dog* (Hawes and Kelleher, 2004), and thus would be offered before the alternatives The accuracy of such systems depends on the strength and breadth of the underlying context database they use Both Stocky *et al* (2004) and Hawes and Kelleher (2004) perform well in the context of weddings and cooking, because their databases have been trained on large corpora on these subjects

Hybrid-systems

The techniques we have discussed are are not mutually exclusive They can be used in parallel (Guenthner *et al*, 1993, Hasan, 2003, Beck *et al*, 2004) Hasan

(2003) uses a tri-gram language model combined with lightweight dependency analyses which discards unlikely syntactic sentence structures A similar technique is employed by Beck *et al* (2004) who use grammar rules to correct the order of words offered by the statistical model Rua and Skiena (1994) use a tri-gram language model of both words and POS tags, combined with grammar rules Analysing complete sentences, they use the Viterbi algorithm to calculate the most likely words and thus the most likely intended sentence Finally, Guenthner *et al* (1993) combine syntactic knowledge with context-sensitive information

2 2 7 Open questions

Having discussed the theory of word prediction and word completion, we will now look at the practical questions that impact on the design of an ambiguous text-entry technique using these approaches There are several metrics that we can use to judge both word prediction and word completion systems clash count, percentage of words guessed correctly and percentage of words known In the following sections we will introduce each of these metrics and discuss how keyboard layout, language size and language model affect a word prediction or word completion system's score relative to each of these metrics

Word prediction

The core issue for any word prediction system is *prediction accuracy* There are several metrics that can be used to measure this clash count, percentage of words guessed correctly, percentage of word known

The clash count of a word prediction system is the number of words in the system's dictionary that share their input sequence with at least one other word in the system's dictionary This is of interest because the number of clashing words likely during text-entry affects the interaction technique we choose Witten (1982) noted that only 2000 words (8%) of a 24,500 word dictionary clashed when entered on a telephone keypad In comparison, tests on the T9 system (D L Grover and

Kuschler, 1998) carried out by Silfverberg *et al* (2000) found that 8437 words (95%) of its 9025 word dictionary were offered correctly first time when sorted by frequency (no detail is given of the exact clash count) Increasing the dictionary size will result in an increased clash count Test by Klarlund and Riley (2003) showed that on average every word in their 463,000 word dictionary clashed once on a telephone keypad For an ambiguous keyboard system, an important point in relation to clash count is that the distribution of letters to keys has a direct affect on the number of words which clash for a given dictionary

Although the clash count of a keyboard may be high, the language model employed by a system may compensate for this by predicting the correct word with high accuracy Thus the *percentage of words guessed correctly* is another useful metric for testing prediction accuracy Klarlund and Riley (2003) found that by using a tri-gram language model with their 463,000 word dictionary the percentage words guessed correctly was over 97% for a QWERTY keyboard and 98% for an alphabetic keyboard The analysis of a system's percentage of words guessed correctly can be refined by examining the percentage of words offered correctly on the first guess, second guess, and so on However, the interaction necessary with the system to choose the second or subsequent guess determines the importance of the initial prediction For example, due to their limited screen space mobile phones only offer one prediction, thus requiring a user to scroll through the alternative predictions As a result, for these systems the first word predicted is extremely important to the user In contrast, immersive environments, which have no screen space limitations, can show more then one prediction simultaneously If any of these predictions can be quickly selected from the list of candidates then the importance of the initial prediction accuracy is reduced

Many word prediction systems, including the one developed in this thesis, use dictionary-based disambiguation For these systems a key factor in their ability to predict a word is having the desired word in their dictionary Consequently, the percentage of words known during use is therefore another useful metric for

judging these systems However, the larger the dictionary, the more words that are likely to clash Consequently, this metric is often at odds with the clash count metric However, as the experiments by Klarlund and Riley (2003) indicate this may not be at odds with overall accuracy

Word completion

As we have mentioned, many of the systems we have discussed employ language models in order to facilitate word completion on *un*ambiguous keyboards (Stocky *et al*, 2004, Fazly and Hirst, 2003, Darragh, 1989) Word completion is also possible on ambiguous keyboards However, it will naturally be less accurate compared to similar systems with unambiguous keyboards This is because word completion systems rely on the letters of a word which have been typed thus far to guide the prediction of a possible complete words With an ambiguous keyboard, the letters typed thus far are not known, but rather the set of ambiguous keys which have been pressed, on which several letters reside Nevertheless, despite the decreased accuracy, using word completion may increase the typing speed When looking at word completion for ambiguous keyboards, we are interested in the effects of keyboard layout, dictionary size and language model order on completion accuracy, but also the effects of word list size, and word selection techniques As with word prediction, word completion has several metrics with which to evaluate its accuracy

One measure of the usefulness of a word completion system is the *percentage of characters saved* This refers to the number of characters saved through the use of the word completion system, relative to the total number of characters typed One of the key factors affecting the percentage of characters saved is the number of words offered at any stage The more words the completion system offers, the more likely it is that the word being typed can be completed by selecting it However, several practicalities limit the typical size of word lists offered Firstly, and most obviously, there is little point in offering 100 complete words, as a user

must first scan though and locate a word before choosing it The total time taken to do this is unlikely to be shorter then the time taken to simply type the remaining letters Secondly, available screen size can affect the potential words offered With virtual keyboards and previously typed words already taking up screen space, there are practical limitations to the number of words that can be fit in the remaining space Finally, the technique employed to choose the complete word from the list of offered words will affect the word list size If a word list must be cycled through iteratively, the effectiveness of larger word lists is reduced The time spent reaching words further down the list will negate the potential time saved completing the word However, if any word offered can be chosen directly from a list, irrespective of its position, then larger word lists have increased potential Garay-Vitoria and Gonzalez-Abascal (1997) examine the effects of word list size on the percentage of characters saved on unambiguous keyboards They show that savings of over 55% and 60% can be achieved with word lists of 5 and 10 respectively Lesher *et al* (1999) demonstrate the positive effects of increased training text size and N-gram order on the efficiency of completions with 10 words They found savings of 54% could be achieved with tri-gram models based on 3 million word texts As with Garay-Vitoria and Gonzalez-Abascal (1997), Lesher's experiments were with word lists where any of the 10 words could be selected explicitly This was done by pressing one of 10 keys, each of which corresponded to a complete word Our experiments will examine the effects of varying word list size, with various completion methods, and examine if the positive effects of N-gram order and training size still hold with ambiguous keyboards

2 3 Optimising keyboard layouts

The first keyboard layout was patented in 1868 by Sholes, Glidden, and Soulé This keyboard layout was alphabetic However, it was superseded by the familiar

QWERTY layout in 1878 (Noyes, 1983) The locations of letters and numbers on keys has been a matter of research, theory, debate, contest and patent applications ever since (Lewis *et al*, 1997) Despite the common belief the the QWERTY layout is sub-optimum even its strongest competitor, the Dvorak simplified keyboard layout (DSK), has failed to displace it It was officially recognised in 1971 by the International Standards Organisation as the standard layout, and remains the de facto computer keyboard layout However, data input on small, reduced and virtual keyboards, such as those found on mobile phones or personal digital assistants (PDAs), has renewed interest in alternative keyboard layouts

In the following section we will discuss the keyboard layouts on both regular and ambiguous keyboards To avoid any confusion, we shall refer to ordinary keyboards, where each key maps to one letter, as explicit keyboards

2 3 1 Optimising conventional explicit keyboards

It is widely accepted that the QWERTY keyboard layout is sub-optimal[2] (Lewis *et al*, 1997) The workloads assigned to each hand and each finger are questioned, as is the amount of movement needed between rows Figure 2 2 shows a breakdown of these values There is a bias toward use of the left hand, and the work distribution of the fingers is especially uneven for the right hand Over 52% of keys struck are on the top row, which requires movement away from the middle, or "home" row

The most widely known keyboard which attempted to address these issues was the DSK layout Patented by August Dvorak in 1936, it was designed to enable simple, rhythmic, rapid movements, in contrast to the erratic motions needed on a QWERTY layout (Lewis *et al*, 1997) The design of Dvorak's keyboard (Figure 2 3) was based on a statistical analysis on common English letter pairs

[2]There is considerable doubt over the reasoning behind the design of the QWERTY layout There is a common misconception that it was chosen over the alphabetic to confuse and slow down typists, thus reducing jamming Noyes (1983) discusses some of the more plausible theories, the most widely accepted of which, is that commonly occurring letter combinations – such as 'qu' – were separated in order to reduce jamming

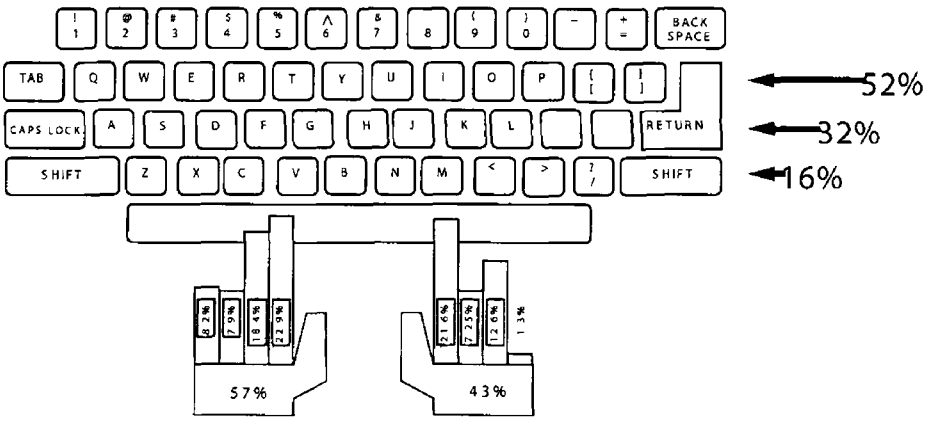


Figure 2 2 Finger and row distribution on a QWERTY keyboard layout (From Brown, 1992)

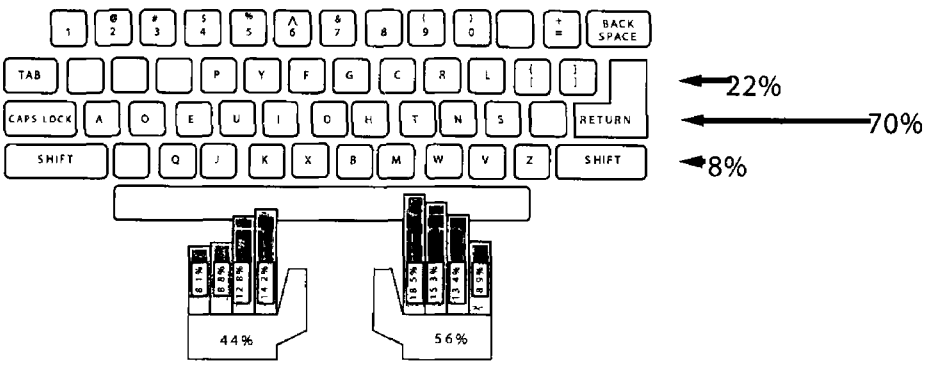


Figure 2 3 Finger and row distribution on a DSK layout (From Brown, 1992)

and attempted to maximise the use of the *home* row, give the strong right hand more work and consequently be less tiring (Light and Anderson, 1993) Almost 70% of typing is performed on the home row, with a larger workload given to the right hand, and fingers are assigned proportional amounts of work Vowels and frequently-used consonants were placed on opposite halves of the keyboard to enable quick, two handed typing of common sequences Keying sequences with alternative hands was shown to be faster then for same-hand entry by Kinkead (1975), who measured the inter key-stroke time (bi-action) of users of a standard keyboard (Figure 2 4)

Although it is accepted that Dvorak's keyboard is indeed superior to the QWERTY layout, there is doubt as to the extent Reported values range from

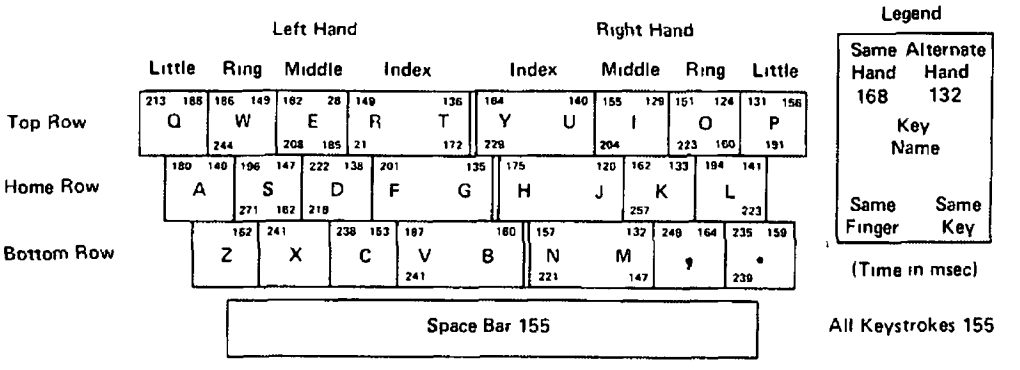


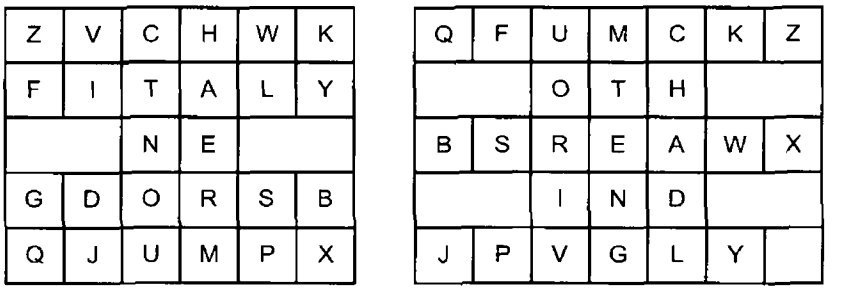
Figure 2 4 Inter-stroke typing times (ms) (From Kinkead, 1975) Times reflect the average time taken to hit a key depending on the hand and finger used to type the previous key

2 3% to an unlikely 50% (Lewis *et al* , 1997) Further improvements to Dvorak's keyboard were offered by Light and Anderson (1993), whose keyboard was created using simulated annealing However, none of the improved designs for physical keyboard layouts provide enough of an improvement to justify the switch from the QWERTY layout, due to the retraining necessary QWERTY remains the king of traditional keyboard layouts, and seems likely to remain so

2 3 2 Optimising virtual explicit keyboards

Virtual, or soft keyboards are keyboards which are displayed on a screen, and exist solely through software Although the dominance of the QWERTY keyboard layout for traditional keyboard layouts is accepted, the optimal design of alternative virtual or soft keyboards for small mobile devices has received considerable interest (MacKenzie and Zhang, 1999, Zhai *et al* , 2000, Textware Solutions, 1998) There are several reasons for this Firstly, by their nature, soft or virtual keyboards can be easily changed or adapted Also, 10 finger touch-typing skills learnt on a regular keyboard do not transfer to on-screen stylus tapping (Zhai *et al* , 2000), and finally, the QWERTY layout itself is even less optimal a design when tapping with a stylus, due to its elongated shape

The Fitaly keyboard (Figure 2 5a) designed by engineers at Textware Soluti-



(a) (b)

Figure 2 5 Fitaly and Opti keyboard layouts

ons (Textware Solutions, 1998), was one of the first optimised keyboard layouts to be designed and sold commercially for PDAs The design was based on the frequency of letters in the English language Letters with the highest frequency were placed closer to the centre of the keyboard, with less frequent letters being relegated to the edge The layout was created by hand and boasts WPM speeds of 41 95, compared to 30 for that of a standard QWERTY When designing the OPTI keyboard (Figure 2 5) MacKenzie and Zhang (1999) used Fitts' Law[3] to predict the tap time for key pairs, and then mapped the shortest tap times to the most common letter-pairs (digraphs) in English It was developed through trial and error Empirical experiments revealed that the OPTI layout resulted in a higher text-entry rate then conventional the QWERTY layout after 4 hours of practice Lewis (1999) created a keyboard using similar techniques, but suggested that users unlikely to reach expert status might benefit from an alphabetic keyboard, which would take advantage of the users pre-existing knowledge of the alphabet

In contrast to these, Zhai *et al* (2000) used computerised quantitative design techniques to search for an optimal keyboard layout The resulting *Metropolis*

[3]Fitts' Law (1954, 1964) quantifies the index of difficulty ID of a movement task based on the distance or amplitude to move A and the width or tolerance of the region within which the move terminates W, where $ID = log_2(\frac{2A}{W})$ MacKenzie (1995) provides a detailed analysis of its implications for movement time prediction in HCI

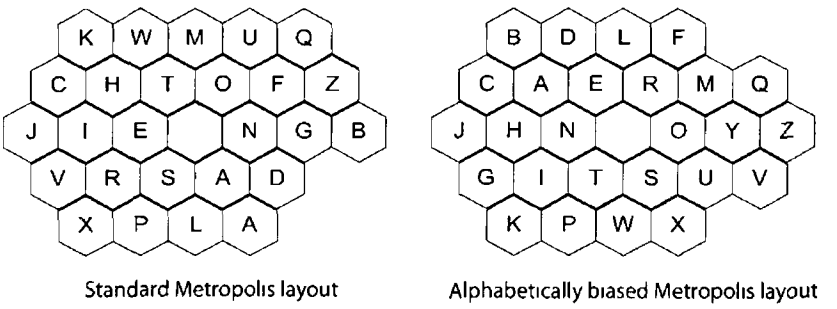


Figure 2 6 Metropolis keyboards original and alphabetically biased

keyboard (Figure 2 6a) was faster then both the OPTI and Fitaly layouts Like MacKenzie and Zhang (1999), they used Fitts' law and digram frequencies to estimate the *cost* of an arbitrary keyboard layout They then used the Metropolis algorithm – a Monte Carlo method, widely used to search for minimum energy states – to search for a layout which offered the lowest cost, thus maximising text-entry speed

Smith and Zhai (2001) later adapted their original keyboard layout (Figure 2 6b) which was designed for expert typists, to facilitate easier learning for novice users They hypothesised that alphabetically biased keyboards would be easier to learn, as the search time would be reduced They tested novice users with both a standard and alphabetically biased layouts and found that beginners were 9% faster on an alphabetically biased keyboard This was significant, as it contradicted earlier research for traditional keyboards by Norman and Fisher (1982), which found that alphabetic keyboards were not easy to use

Lesher and Moulton (2000) created an optimised keyboard based on letter N-grams and an elliptic travel cost function defined by Levine and Goodenough-Trepagnier (1990) However, no user tests were performed

Finally, Hughes *et al* (2002) used empirical bi-action tables to create and assess optimal keyboard layouts Here bi-action is defined as the physical motion from one key to the next Rather then using a human performance model, such as

	K	G	I	C	Z
	F	N	T	H	W
Q	O	S		A	Y
J	U	R	E	D	V
	P	M	L	B	X

Figure 2 7 Optimised layout based on bi-action tables

Fitts' law, Hughes *et al* (2002) recorded 5 participants as they made all possible bi-actions for a keyboard, and measured the time for each motion The average time taken by each user to perform each bi-action was then stored in a bi-action table They combined this information with letter digraph frequencies to create an optimised keyboard, which had a predicted expert speed of 65 26 WPM (Figure 2 7)

2 3 3 Reduced key keyboards

Although many virtual keyboards simply reduce the physical size of each key in order to offer smaller keyboards suitable for mobile text-entry, another option is to reduce the number of keys offered These reduced keyboards can offer larger key sizes, but to do so must sacrifice the 1-to-1 mapping of letters to keys offered by traditional keyboards One alternative, which allows for unambiguous typing, is the use of chords – where several keys are pressed simultaneously to select one letter

As well as the Twiddler II, which we have previously discussed (Section 1 2 3), there exist several chord variations GKOS (2000) is a 6 key system, which uses various combinations of keys, similar to the Twiddler II In contrast, the half-QWERTY keyboard (Matias *et al*, 1994), and FrogPad (1999) behave closer to traditional keyboards, employing shift keys which traditional keyboard users are familiar with The FrogPad keyboard is optimised so that the most common 15 letters typed are the easiest to type The half-QWERTY as the name implies,

copies the QWERTY keyboard layout The half-QWERTY mirrors the keys struck by the non-active hand, so the left hand little finger strikes the Q key as normal to select Q, but the P key – usually struck with the right little finger – is selected by pressing and holding the spacebar and then striking the Q key This layout takes advantage of the knowledge already gained from typing on a standard keyboard, and subjects can achieve 50% of their standard keyboard rate after 8 hours training

2 3 4 Optimising ambiguous keyboards

In contrast to chording keyboards, ambiguous keyboards, as we have already discussed in Section 2 2 2, require only one press to select each key, relying instead on a disambiguation engine to identify the intended letter Their accuracy is dependent on a number of factors including the disambiguation engine, keyboard layout used, and the interaction techniques employed The accuracy improvements offered by an accurate disambiguation engine, taking advantage of the entropy of the English language, have been discussed However, another factor affecting the accuracy is the keyboard layout and key-count Effective ambiguous keyboards have been implemented with as few as 4 keys (Harbusch and Kuhn, 2003, Evreinova *et al*, 2004) Reducing the key-count will naturally increase ambiguity and have a negative effect on the accuracy of prediction Such severely reduced keyboards are typically used where input capabilities are extremely limited, for example where physical disability limits motion (Kuhn and Garbe, 2001, Hansen *et al*, 2003)

For any given key-count, the keyboard layout also effects accuracy The optimisation of ambiguous layouts is dependent on the disambiguation technique used, and the interaction techniques possible

For letter-level disambiguation on phone-pad keyboards, several layouts have been suggested Levine *et al* (1985, 1986) were the first to recognise the potential for increased prediction accuracy if the layout was changed to reduce ambiguity

They used a genetic algorithm to minimise the ambiguity for letter bi-gram based prediction In contrast, Foulds *et al* (1987) analysed the error rates of the characters on different keys and identified the main characters which reduced the prediction accuracy By switching these keys, they created an alternative layout which further improved accuracy Their TOC keyboard, named after the 3 characters which were moved, predicted the correct character with an accuracy of 90 8% Foulds *et al* (1987) contrasted their optimised layout with those of Levine *et al* (1986) and found similar performance

Slight improvement of the TOC keyboard was achieved by Lesher (1998), who used a confusability matrix to guide an *n-opt* algorithm Lesher's confusability matrix was created by simulating text-entry for a sample text T_s with a prediction algorithm P_x For every character in the sample text, the prediction algorithm ranks the possible letters For every letter β which is ranked higher than the intended letter α, the value in the confusability matrix at position $C(\alpha, \beta)$ is incremented Thus, for a given character pairing α, β, their mutual confusability $C_m(\alpha, \beta)$ can be calculated as $C(\alpha, \beta) + C(\beta, \alpha)$ Lesher created an optimal keyboard by minimising the mutual confusability of characters on a keyboard layout

An optimised telephone key-pad layout for word-level disambiguation was created by Oommen *et al* (1991) Optimal keyboard designs were achieved by attempting to minimise the number of words with identical key sequence mappings For any given dictionary, each word must map to a key-sequence With increased dictionary size, the potential for 2 or more words to map to the same key-sequence increases Oommen *et al* (1991) attempted to find a keyboard which reduced clashing by ranking keyboards according to the number of individual sequences they created for a set dictionary, thereby minimising possible clashing and, ultimately, ambiguity when typing Using a stochastic automaton they reduced the average number of clashes for their 1067 word dictionary to just 5 7

Similarly, (Bahuman *et al* , 2000, Kuhn and Garbe, 2001, Harbusch and Kuhn, 2003, Garbe, 2000), used ambiguous keyboards that were optimised to reduce the word clashing However, in contrast to the keyboard of Oommen *et al* (1991), which is optimised for users who could select any one of the ambiguous keys at a time, those of (Bahuman *et al* , 2000, Harbusch and Kuhn, 2003, Garbe, 2000, Kuhn and Garbe, 2001) are designed for motor impaired users, where input options are severely restricted These systems highlight each key in rotation, and users then press a key to indicate that the highlighted key is the one intended Optimisation of these keyboards therefore, is focussed on choosing a layout which would require as little cycling as possible, but also provides as few clashes as possible (as these also have to be rotated through) The variation in priorities highlights the importance of the analysing the planned interaction technique when optimising ambiguous keyboard layouts

2 3 5 Optimising keyboard layouts discussion

Although the ultimate goal of most optimal keyboards is an increase in effective throughput, or WPM, the design of an optimal keyboard begins with deciding what exactly needs to be optimised There is no definitive optimum keyboard layout, but rather, keyboard layouts which have been optimised with regard to a certain performance characteristic The optimisation of the traditional explicit keyboard layout for 10 fingered typing, has been the subject of considerable research (Noyes, 1983, Lewis *et al* , 1997) Here the optimisation considered was the reduction of the travel time of each of the fingers In contrast, reduced virtual keyboards for PDAs must be designed to minimise the travel time of just a single stylus

Optimisation of ambiguous keyboards is usually focussed on choosing the optimum arrangement of letters on keys The ultimate aim is to reduce the ambiguity and, as a consequence, the need to correct incorrect predictions Here, the optimal keyboard is closely linked to the prediction algorithm used and the

interaction possible with the system Thus, layouts optimised for letter-level prediction will be sub-optimal if used with word-level prediction and vice-versa

While (Foulds *et al*, 1987, Textware Solutions, 1998, MacKenzie and Zhang, 1999) chose to optimise keyboards by hand, (Garbe, 2000, Light and Anderson, 1993, Zhai *et al*, 2000, Lesher *et al*, 1998) have attempted to use search techniques such as *n-opt* and simulated annealing, combined with models of human performance and linguistic knowledge, to search for optimised layouts

In Chapter 3, we will examine the creation of keyboards optimised for word-level prediction for immersive text-entry

2 4 Gesture recognition

Human-computer input devices can generally be categorised into two groups firstly there are those that monitor explicit, unambiguous actions, such as the depressing of a mouse button or a key on a keyboard, or the tapping of a stylus on a touch screen button, secondly there are passive devices, which constantly observe or attend the user Examples include microphones, video cameras, position trackers and data gloves, which monitor the speech, gaze, or position of the user The passive nature of these devices leads to the problem of interpretation of the user's actions, which are often ambiguous Pattern recognition is the identification of patterns in large datasets, it can be used to attempt to identify and classify patterns from a stream of data provided by a passive device, and is a non-trivial task (Harling, 1993) Gesture recognition may be viewed as a problem of pattern recognition, in which the patterns to be classified are instances of input from posture sensors (Watson, 1993) In the following section we will look at the application of pattern recognition techniques to gesture recognition and will review the most common recognition algorithms currently used, particularly those suitable for virtual keyboards

2 4 1 Applications of gesture recognition

Gesture recognition is used for variety of various applications, from sign language recognition (Kadous, 1995, Wu and Sutherland, 2001) to being used to control television on occasions when the remote control just can't be found (Freeman and Weissman, 1994) It is particularly popular in immersive VR applications, where gloves are mapped to virtual representations of the hand, which can be used to manipulate objects in the world Gestures are typically used for selection and manipulation of objects, as well as travel These can take the form of natural intuitive gestures, such as pointing or grasping motions, as might be used in the real world (Mine, 1995), or by mapping predefined gestures or postures to desired actions, such as the selecting from a menu (Bowman *et al*, 2001b)

Gesture recognition has also been applied to communication, where various attempts have been made to recognise sign language, from finger spelling to full dynamic hand motions (Shamaie and Sutherland, 2003, Starner and Pentland, 1995) In contrast, Fels and Hinton (1993) mapped hand gestures to 10 control parameters, which allowed the hand to act as an artificial vocal tract and produce speech in real time Finally, text-entry in immersive environments was achieved by Bowman *et al* (2001b) and Evans *et al* (1999), and involved the recognition of key-press gestures, which were mapped to virtual keyboard keys

2 4 2 Datagloves

In order for gestures to be recognised, they must first be measured Raw data, corresponding to the position and orientation of the hand, must be collected to which algorithms can be applied and gestures recognised One method by which this is achieved is through instrumented gloves, or datagloves, which measure and relay data detailing values such as finger flexion and abduction (See Figure 2 8) These gloves are often augmented with 3D positional trackers which relay hand position and orientation data Unlike data from computer vision, which must

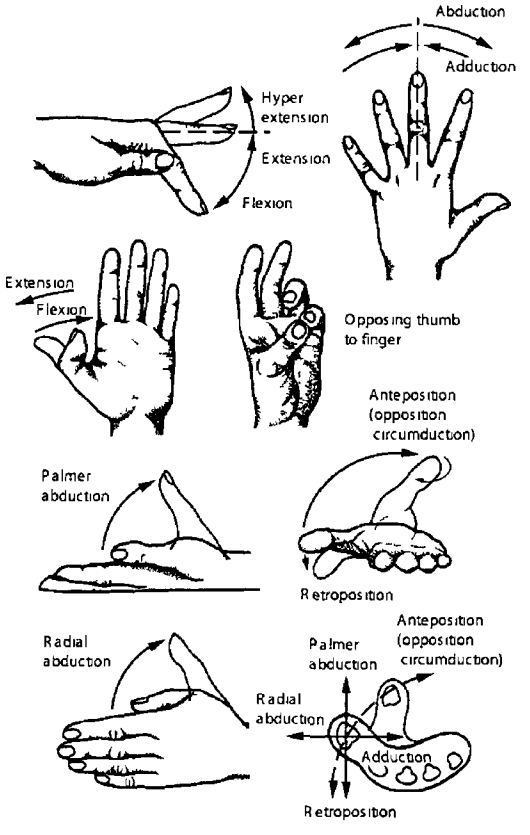


Figure 2 8 Finger motions of the hand (adapted from Sturman, 1992)

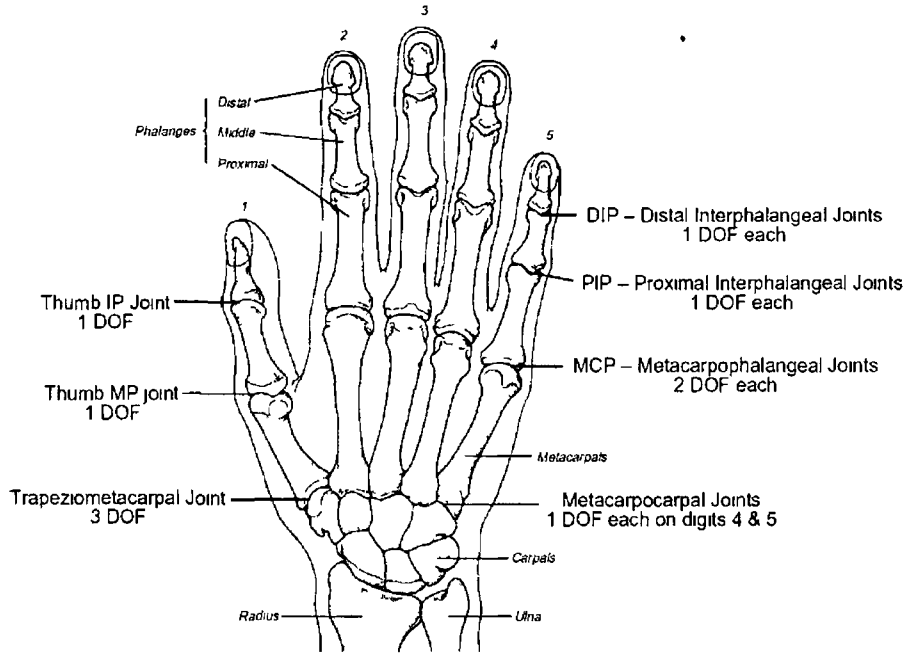


Figure 2 9 Bones and joints of the hand (adapted from Sturman, 1992)

be pre-processed before gesture recognition can take place, information recorded from dataglove and trackers can be analysed directly

A brief history One of the first gloves to be described in literature is the "Sayre" glove (Defanti and Sandin, 1977) The design was based on an idea by Rich Sayre of the University of Chicago (Sturman, 1992) Flexible tubes with a single light source at one end and a photocell at the other were attached to each of the fingers of the gloves As the fingers were bent, the amount of light reaching the photocell decreased evenly, this allowed for accurate measurement of finger flexion The glove could measure the metacarpophalangeal joints for the four fingers and thumb, along with the proximal interphalangeal joints on the index and middle fingers, for a total of 7-DOF (LaViola, 1999)

Grimes (1983) secured a patent on a dataglove which describes a man-machine interface "for translating discrete hand positions into electrical signals representing alpha-numeric characters" Grimes's glove incorporated sensors which mea-

sured both finger flexion and contacts at key positions as well as hand orientation Gesture recognition was hard-coded into Grimes's glove, mapping various hand positions letters of the alphabet

The Z-Glove and Dataglove, developed by Zimmermann *et al* (1987), were considerably more versatile The gloves were made commercially available through VPL Technologies Flexion was measured using Zimmerman's patented optical flex sensor (Zimmerman, 1985) Similar to the Sayre glove, a photocell measured the direct and reflected light through a flexible tube attached to the fingers Up to 15 of these flex sensors were attached to the glove which allowed the flexion at the MCP, PIP joints to be measured, as well as the abduction of the fingers (See Figure 2 9 for hand joints) Tactile feedback was provided by piezoceramic benders, which were mounted underneath each finger This produced a tingling or numbing sensation in the fingertips On the Z-Glove, tracking was achieved with ultrasonic transducers attached to either side of the metacarpal, which allowed roll and yaw to be determined when a direct line of sight was available The Dataglove used a 3SPACE magnetic tracking system which allowed for 6-DOF tracking

Initially developed as a controller for the Utah/MIT Dexterous Hand, the Dexterous HandMaster (DHM) was an exoskeleton-like device worn over the fingers and hand (Sturman and Zeltzer, 1994) The technology was licenced and sold by Exos, Inc The glove measured 20-DOF of the hand, four for each finger, and four for the thumb It was accurate to within 1^o of flexion (Sturman, 1992)

Mattel introduced a low cost glove for the Nintendo in 1989, which used resistive-ink flex sensors which registered the overall flexion of the thumb and four fingers Acoustic trackers were used to accurately locate the glove within one-fourth of an inch (Sturman and Zeltzer, 1994) Mattel stopped producing the glove after only 2 or 3 years However, the gloves low cost meant it was used regularly by VR researchers, who reverse-engineered it in order to connect it to PC serial ports

Current input technology There are four datagloves currently available on the market, which vary greatly in cost and accuracy The most recent addition to the market is the p5 from Essential Reality (2003) Retailing at under $70, the glove is considerably cheaper then anything else available on the market and has been designed primarily for the computer gaming market Proprietary flex sensors measure overall finger and thumb flexion, while 6-DOF positional tracking is provided by 2 cameras Although no formal studies have been carried out on the glove, personal experience revealed a high level of hysteresis during flexion and extension of the fingers However, given its low cost, and freely available software development kit, the p5 is likely to replace Mattel's PowerGlove as a low cost VR input device

At the opposite end of the market is the CyberGlove Originally developed by Kramer for his "Talking Glove" project (Kramer and Leifer, 1989), it is sold commercially by Immersion Corporation (2004) (formerly Virtual Technologies) and is currently the glove of choice for VR research The glove is available in 2 models, with 18 and 22 sensors respectively The 18 sensor model measures the flexion of the MCP and PIP of the 4 fingers and thumb, thumb opposition, abduction/adduction between the fingers, radial and palmer abduction, and wrist flexion and abduction The 22 sensor model also measures the flexion of the DIP on each of the 4 fingers Evaluations of the glove by Kessler *et al* (1995) and LaViola (1999) suggest it is accurate to within one degree of flexion Perhaps the only negative aspect of the glove is its price, which is in excess of $10,000

Pinch Gloves, offer an alternative approach for gesture recognition Unlike traditional datagloves, they do not have flex sensors, but instead have electrical sensors in each finger tip When two or more of these sensors come in contact a signal is returned indicating that contact or *pinching* has occurred The advantage of the gloves is that contact is binary and unambiguous, greatly simplifying gesture recognition Gloves are sold in pairs, creating a large potential gesture set However, the disadvantage of the gloves is that, without flex sensors, it is

difficult to provide accurate virtual representation of the hands Nevertheless, pinch gloves are well suited to situations where simple gestures are sufficient, and, when combined with 3D tracking devices, can be used for a variety of interaction techniques (Bowman *et al*, 2001b) Pinch Gloves are made available commercially from Fakespace Inc (2004) and retail for roughly $2000

Finally, Fifth Dimension Technologies (5DT) produce datagloves with 5 and 16 flex sensors The 5 sensor model, often referred to simply as the *5th Glove*, uses proprietary fiber optic based flexor technology to measure overall finger and thumb flexion The 16 sensor model has 2 of these sensors per finger, and also measures finger abduction Both models are fitted with a 2-axis tilt sensor, which measures 120 degrees of pitch and roll We have used the 5 sensor model throughout the course of our experiments Measurement of the flexion of each finger is achieved by altering the fiber optic cable at 2 key points per finger, which when bent affect the light received by the opto-electronics (See Figure 2 10) The effect is that the glove can sense the overall flexion of each finger, but cannot differentiate between flexion points The 5 sensor model retails at $500 while the 16 sensor model retails for $4000 (Fifth Dimension Technologies, 2004)

2 4 3 Typing movements

Interaction with a virtual keyboard involves accurate recognition of users' finger motions (Figure 2 8) Of primary interest is the flexion and extension of the fingers and the abduction of the thumb, as users *press* the virtual keys of the keyboard This involves recognition of flexion at the MCP and PIP joints of the fingers, and the trapeziometacarpal of the thumb

2 4 4 Gesture and posture recognition

Throughout the text we will make a distinction between postures and gestures

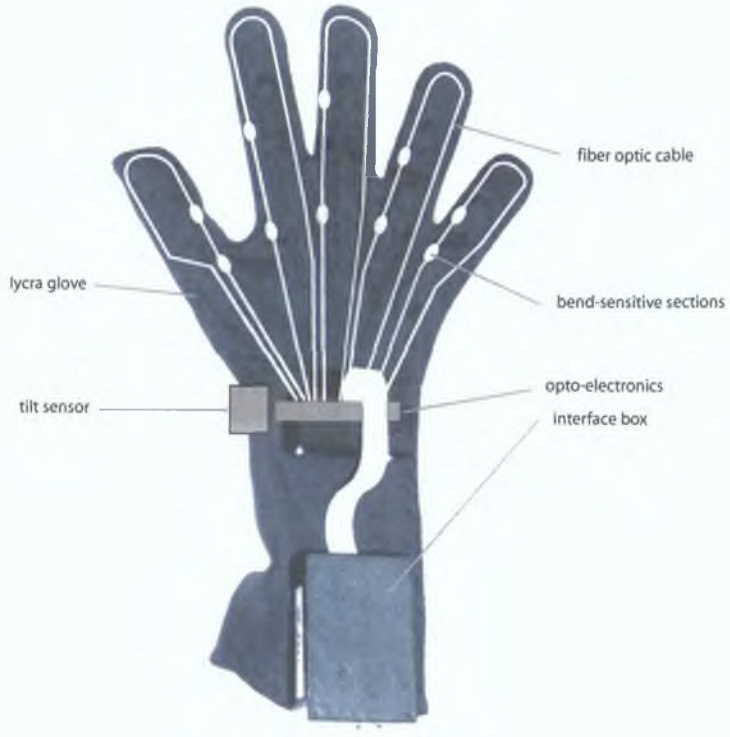


Figure 2.10: The 5DT dataglove, or *5th glove*.

Gesture recognition Gestures, or dynamic gestures, are dynamic motions, which may be measured over time in 3D space Gestures may also be measured relative to other parts of the body Examples would be the motion we make when waving good-bye Gesture recognition represents the attempt to understand what the user is doing, and should not mistaken for gesture interpretation, which attempts to combine knowledge of what the user is doing, with context to try understand the users intention A waving motion for example, would have a different interpretation if the user was in an immersive environment standing in front of a dirty window, where this motion would more likely be an attempt to clean the window, rather then wave good-bye to it

Posture recognition Posture, or static gesture recognition, does not measure hand motion over time or in 3D space Thus, postures traditionally correspond only to the measurement of the hand orientation and finger flexion at a particular moment in time (See Figure 2 8 for a description of finger movements) Examples of postures include the OK sign or a clenched fist (See Figure 2 11 for examples of postures)

The recognition of key-presses for text-entry in immersive environments can be classified as posture recognition, as we are only interested in the position of each finger, and not the position of the hand in 3D space However, we will show that the accurate of measurement of postures is increased if they, like dynamic gestures, are measured over time Thus gesture recognition needed for text-entry might be described most accurately as dynamic posture recognition

2 4 5 Simple versus complex posture recognition

At this point a distinction should be made between what Sturman (1992) referred to as simple and complex posture recognition Many of the postures we commonly make are comprised of fully flexed or extended fingers, these can be considered simple postures (Figure 2 11) However, there are many postures, which involve

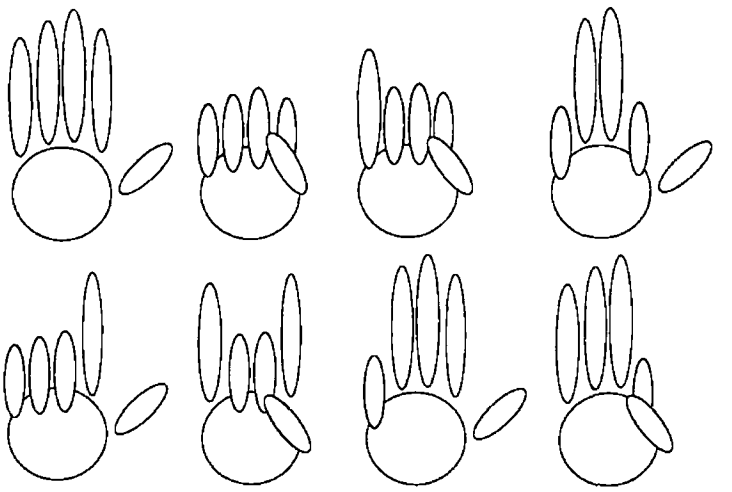


Figure 2 11 *Simple postures* These are some of the 32 possible postures possible using only flexed or extended fingers Sturman noted that not all postures are achievable or comfortable However, sufficient practical postures exist to satisfy most applications

only partially flexed fingers Many examples of these gestures can be found in sign language Considerable research has been conducted in an attempt to recognise such complex postures (Liang and Ouhyoung, 1996, Harling, 1993, Kadous, 1995) These systems typically have large posture sets, require considerable training, and are often user dependent The recognition of key-press postures for virtual typing can be classified as simple posture recognition, which, in contrast to complex posture recognition, has only a small posture set and should ideally be accurate while maintaining user independence Thus, in reviewing related work, this thesis focuses primarily on simple posture recognition, and only mentions complex gesture recognition for completeness For a more comprehensive review of both static and dynamic gesture recognition techniques, see Watson (1993) and LaViola (1999)

2 4 6 Recognition errors

Errors in recognition can generally be classified as either false positive, false negative, or misclassification errors False positive recognition errors occur where

the system recognises a gesture when one was not intended by the user False negative recognition errors occur where the system doesn't recognise a gesture when one is intended by the user Finally, misclassification errors occur when the system recognises the user's attempt to create a gesture, but misinterprets the user's intention and mis-classifies the gesture as an alternative one

2 4 7 Factors affecting recognition

Sympathetic bending One of the hindrances to accurate gesture recognition for virtual typing is *sympathetic bending*, or *enslavement* This refers to the tendency of fingers to move in sympathy with others that are bent Zatsiorky *et al* (2000) define enslavement as "the involuntary force production by fingers not involved in a force production task" The most common example of this, is the propensity of the ring finger to bend when the little finger is bent Although the exact cause is unknown, sympathetic bending is believed to be caused by a combination of factors, including the mechanical coupling of tendons (See Figure 2 12), and neural interconnections among structures controlling flexor muscles in the hand (Zatsiorky *et al*, 2000) Unfortunately, not only does the severity of sympathetic bending vary greatly between users, but the fingers which bend in sympathy can also vary Thus, although the majority of users will have some level of sympathetic bending of the little and ring fingers, others might have strong sympathetic bending of the index finger, ring finger or both when the middle finger is bent This variation between users greatly hinders the creation of a user independent system which requires little or no training

The segmentation problem The segmentation problem refers to the problem of trying to distinguish at what stage a gesture has begun, and at what stage it has ended While forming a posture, we may inadvertently create another posture, which can cause the intended one to be misclassified For example, creating a fist posture, if the index finger is moved slightly slower then the other fingers, a point

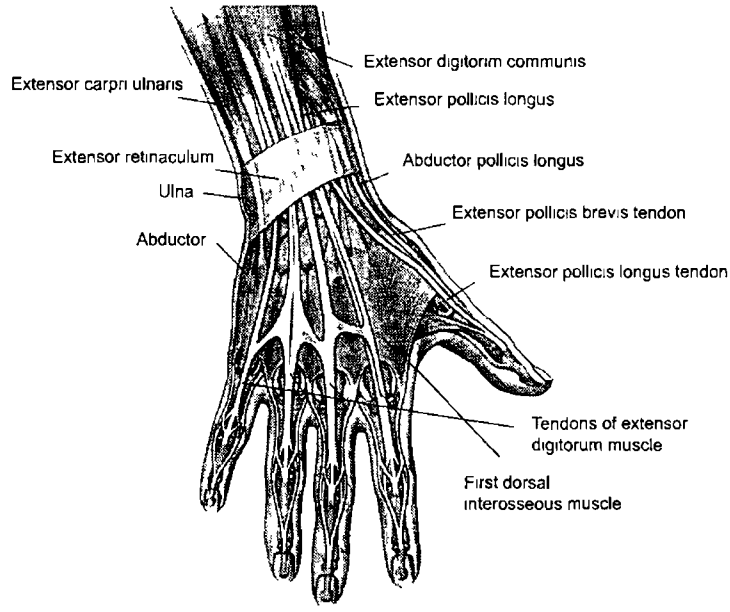


Figure 2 12 Muscles of the hand (from Sturman, 1992)

gesture might be recognised Thus, the problem becomes one of identifying which portions of a gesture are transitions (sections were the user is in the process of making a gesture), and which are the final intended postures

Glove-based problems Finally, the glove used to sense gestures can have a significant impact on recognition We have used the 5 sensor model throughout the course of our experiments As previously mentioned, measurement of the flexion of each finger is achieved by altering the fiber optic cable at 2 key points, which when bent affect the light received by the opto-electronics (See Figure 2 10) The effect is that the glove can sense the overall flexion of each finger, but cannot differentiate between flexion points Because flexion is only sensed at two specific points per finger, natural variations in hand size can lead to reduced sensitivity of the glove This occurs when the user's flexion at the MCP and PIP joints do not map accurately to the flexion sensitive points on the glove for one or more fingers As a result of these common mis-mappings, the glove sensitivity is often less then desirable, and requires significant finger flexion before the glove

will register any movement In practice, the glove will usually map to at least one of the finger joints This will be visible during initial calibration as users bend their fingers at each joint, but that can require the user to remember which joint they must flex for each finger, which is far from ideal Instead, users often simply over-pronounce each flexion in an attempt to ensure one of the sensors is flexed, which can in turn exaggerate sympathetic bending Another problem with the sensor technology used by the 5DT data glove, is that it cannot distinguish between standard finger flexion and hyper-extension Consequently, the glove will often indicate that a finger is flexed when in fact it is hyper-extended

2 4 8 Pattern recognition applied to gesture recognition

Pattern recognition has applications ranging from medicine, to robotics and military systems Examples of pattern recognition systems include character recognition, fingerprint identification, minefield detection (Jean Laurent, 1997) Pattern recognition may be summarised as the categorisation of input data into identifiable classes, via the extraction of significant features or attributes of the data, from a background of irrelevant details Determining what are *significant features*, it a non-trivial task, and will vary from case to case However, in general, significant features are values which should be similar for objects belonging to the same class, and distinct for objects in different classes

These features form a *feature vector* which uniquely identifies an object or pattern These features may then be mapped to a feature space, where objects from the same class cluster together It is the role of the *classifier* to divide this feature space into distinct regions which identify the boundaries of these clusters, and identify the class to which a feature vector belongs Typically, the classifier is trained on a set of feature vectors with a known classification, which it then uses to identify new unclassified vectors

Gesture recognition may be viewed as a problem of pattern recognition, in which the patterns to be classified are instances of input from glove sensors

2 4 9 Previous work

Pioneering work in this area is usually credited to Grimes (1983) Although gesture recognition was hard-coded into Grimes's glove, many more flexible systems have since been explored

Template matching

Template matching is probably one of the simplest methods for posture recognition, and is well documented (Sturman, 1992, Watson, 1993, LaViola, 1999) Essentially, the similarity between input data and predefined templates is measured The input is then classified as the gesture or posture which it most closely resembles In practice, there is a similarity threshold, which the input data must fall within, otherwise no gesture is classified Various techniques are used to define the similarity metric used to classify gestures

Early pioneers in this area, VPL's posture recognition work used a simple table lookup technique to define templates (Zimmermann *et al*, 1987) Each table entry corresponded to a posture that defined a range of valid values for each sensor (Figure 2 13) If the sensor values all fell within the valid ranges defined in any one table entry, then the corresponding posture was recognised Each table entry also contained hysteresis values, which widened the posture range once it had been recognised This allowed users to hold the posture comfortably, and prevented the accidental recognition of extra postures due to small fluctuations in sensor readings

VPL's technique was simple, easy to implement and flexible However, in practice, the sensor range in each table entry must be quite wide, up to 30% of the total range (Sturman, 1992) This is due to a combination of glove inaccuracies, and natural variances in user performance Sturman (1992) reported that with over 10 table entries an overlap occurred, which caused more then one gesture to be recognised for many postures

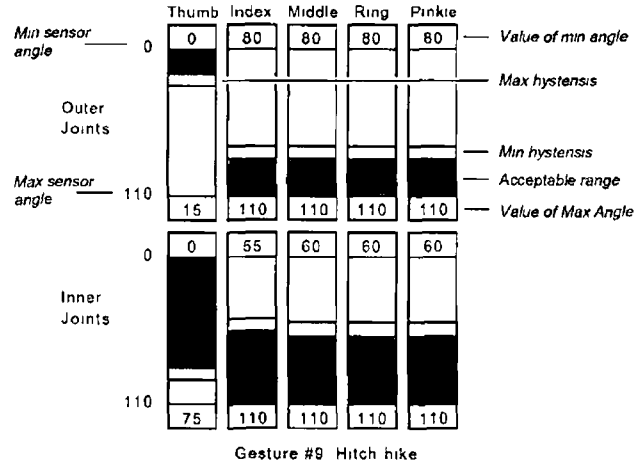


Figure 2 13 *VPL gesture editor* A typical table entry for a gesture The valid sensor range is in dark grey, and the hysteresis range is in light grey (From Sturman, 1992)

Sturman (1992) proposed a simplified version of this technique Recognising that 90% of gestures used a combination of fully flexed or fully extended fingers, he suggested that threshold values could be hard coded into any system to recognise if a finger was flexed or extended By normalising the flex values between 0 0 and 1 0, flexion was detected if the flex value exceeded 0 8, while extension was detected if the flex value fell below 0 2 Thus, a point gesture was recognised if the flex values of the little, ring, and middle fingers were above 0 8 and the index finger below 0 2 As can be seen from Example 2 1, fingers which were not central to the gesture (in this case the thumb, which could be either flexed or extended) could simply be ignored

```
/* flex[d][j]  normalized flex value for digit d  joint j */
if ( flex[index ][MCP] < 0 2 &&
     flex[middle][MCP] > 0 8 &&
     flex[ring  ][MCP] > 0 8 &&
     flex[pinkie][MCP} > 0 8 ) {
  posture_recognized = POINTING
}
```

Example 2 1 Code to recognise point (From Sturman (1992))

Evans *et al* (1999) proposed a further simplified version of Sturman's tech-

nique in their recognition system, which was designed to recognise key-press gestures (henceforth referred to as the *maximum flexion technique*) They had no lower threshold for flexion, a key-press gesture was recognised if its corresponding finger flexion exceeded a minimum threshold They recognised that sympathetic bending could cause multiple gestures to be recognised, in which case the finger with the greatest flexion was selected as the intended one Evans *et al* reported recognition accuracy of 99% with their maximum flexion technique However, although accurate, their technique dramatically reduces the possible gesture set While Sturman's method allows for 32 (2^5) possible gestures, the maximum flexion technique reduces this to only a small subset of 5 gestures - individual flexion of each of the fingers and the thumb

An alternative similarity measurement, suggested by Kramer and Leifer (1989) and Newby (1993), is the Euclidean distance between the current input and each posture template, where the closest posture is accepted if it is within a threshold distance This system can have several variations Instance-based learning may be used, whereby the system is initially trained on a set of example gestures Using the K-Nearest Neighbour algorithm, the distance is measured between the current input and all training postures, and the K closest postures are returned The current input is then classified as the posture with the majority of the K closest postures A less computationally intensive alternative to the K-Nearest Neighbour is to simply calculate the average values for each training posture and then measure the Euclidean distance between these and the current posture

Neural nets and hidden Markov models

Neural nets offer an alternative to standard template matching techniques, and are widely used for various complex posture and gesture recognition techniques (Fels and Hinton, 1993, Sandberg, 1997) Using a VPL dataglove and a Polhemus tracker for data acquisition, Fels was able to recognise 66 different hand postures, each of which was assigned to different words This was achieved with 5 feed-

forward neural networks trained using back-propagation Fels reported an error rate of 6% Of these, 1% were misclassification, and 5% false negative errors As is common with neural nets, the system must be re-trained for every user Sandberg (1997) reported posture recognition accuracy of 93 5% on a set of 14 postures, attributing most errors to false positive recognition of gestures before they were fully formed (transition errors)

Hidden Markov models (HMM), are popular in speech recognition, and have been applied to gesture recognition for both vision- and glove-based systems (Liang and Ouhyoung, 1996, Starner and Pentland, 1995) Like neural nets, HMMs must be trained on a large set of training data However, the typical reported accuracy is usually about 90% (LaViola, 1999) If one HMM is used to recognise all gestures, it must be retrained if new gestures are added However, this can be avoided by having a HMM for each gesture (Liang and Ouhyoung, 1996), in which case only a new HMM must be trained when adding a gesture

Both NNs and HMMs require extensive training, which can involve much trial and error, with little guarantee of good results Both techniques are suitable for larger complicated gesture sets, where overlapping gestures make standard template matching unfeasible However, the training time required, and the trial and error needed to find optimal designs, are unnecessary for simple posture recognition such as is needed for key-press recognition, where template matching is preferred

2 4 10 Conclusions

Gesture recognition is required to identify key-press motions created by users as they press virtual keys Ideally, any gesture recognition technique employed should be user-independent and require no training However, variations in hand size and dexterity lead to problems of gesture segmentation and sympathetic bending, which cause false positive, false negative, and misclassification errors

In Chapter 5 we will examine the effect of these errors on the text-entry task

We will test effectiveness of the template matching techniques reviewed in Section 2 4 9 Finally, based on the results of these tests we will recommend a hybrid technique suitable for text-entry with virtual keyboards

2.5 Interaction in immersive environments

2 5 1 Introduction

This section will review the design of interaction techniques for immersive environments, specifically, those suitable for interaction with our virtual keyboard An interaction technique is the means through which a user achieves a certain desired task It is achieved via a user interface, which combines hardware and software It maps the information from the input device to an action within the system, which may then be represented visually through an output device An interaction technique may be as simple as clicking a mouse button, or as complex as a series of gestures (Bowman, 1999, Kettner, 1995, Bowman *et al* , 2004)

Early research in immersive environments focussed on the capabilities of hardware, with little consideration for interaction techniques However, as immersive environments matured the importance of effective interaction techniques has received more recognition, and is the subject of considerable research (Hand, 1997, Bowman, 1999, Bowman *et al* , 2004, Mine, 1996) Again hardware played a large part in the focus of early research Work by Bier (1987) focussed on the use of 2-DOF input devices such as a mouse, in 3D environments Using abstract cursors called *skitters* and *jacks*, Bier's work was typical of early work which attempted to perform 6-DOF tasks, controlling 3D position and orientation, by limiting the DOF at any one point The arrival of datagloves, and 3D trackers such as the VPL dataglove and Polhemus tracker, provided true *integrated* 6-DOF trackers, allowing users to simultaneously control 3D position and orientation

2 5 2 Universal tasks

Most virtual environments require four basic universal tasks *selection*, *manipulation*, *travel*, and *system control* (Bowman, 1999) This classification is not the only accepted taxonomy for interaction techniques Way-finding, is sometimes distinguished from travel (Bowman *et al* , 2004) Travel can be achieved through the selection and manipulation of a camera which represents the user's *eye* to the virtual environment (Hand, 1997) Similarly system control can be achieved through the manipulation of objects representing the environment (Hand, 1997)

Interaction in immersive environments will often involve selection of an object, followed by some manipulation of that object However, when interacting with a virtual keyboard, the process of selection alone is central, as no manipulation will be performed We are only interested in choosing or selecting an ambiguous key, and do not need to perform any further action (such as translation, rotation, etc) upon it Thus, the act of selection alone is the goal The remainder of this section will focus on selection techniques within immersive environments

2 5 3 Isomorphism in VR

It can be argued that one of the benefits of VR is that it affords natural interaction The isomorphic mapping of our hands to virtual counterparts allows us to interact with virtual objects as we would in real life However, this direct one to one mapping has drawbacks which can limit the potential of VR For example, objects which are beyond arms length cannot be reached Non-isomorphic methods of interaction, although potentially reducing realism, provide the advantage of greater control Thus, for example, movement can be achieved through the use of gestures, rather then physical walking Distant objects can be selected with laser pointers emanating from the finger (Mine, 1995) or by squeezing their 2D projection (Pierce *et al* , 1997) As we shall see, both techniques can co-exist, with isomorphic interaction augmented by seamless switching to non-isomorphic

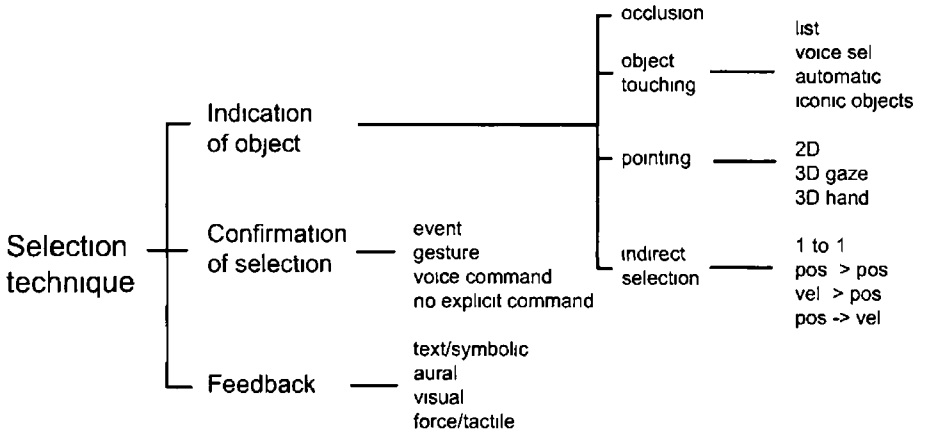


Figure 2 14 Taxonomy of selection (From Bowman *et al*, 2004)

techniques when necessary (for example the Go-go technique (Poupyrev *et al*, 1996))

2 5 4 Taxonomy of selection

Taxonomy typically refers to the science and methodology of classifying organisms based on physical and other similarities However, the same process of sub-categorisation can be applied to interaction in immersive environments By breaking a required task into sub-categories, and these in turn into sub-sub-categories, we can identify the core components required to perform a task (Bowman *et al*, 1999) A taxonomy can also be used to describe a specific instance of decomposition The power of such taxonomies is that a variety of interaction techniques can be created to complete the desired task, by varying the combinations of core components used Bowman *et al* (1999) define a taxonomy for selection in immersive environments (Figure 2 14), which we shall use as the basis for our own taxonomy of ambiguous text entry in Chapter 6

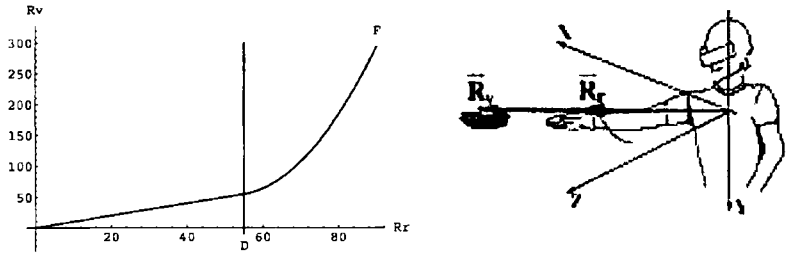


Figure 2 15 Go-go interaction technique (From Poupyrev *et al*, 1996)

2 5 5 3D selection techniques in immersive environments

As mentioned in Section 2 5 3, although the direct 1-to-1 mapping of a physical hand to a virtual representation offers the most *natural* method of selecting objects, as it closely resembles real life, various alternative techniques have been proposed for selection of objects in immersive environments These can generally be classified as either virtual hand, or pointing techniques

Go-go technique One of the simplest extensions of the classic virtual hand technique, is the go-go technique (Poupyrev *et al*, 1996) The go-go technique tackles the problem of selecting objects beyond the user's reach by creating a non-linear mapping between the user's hand and its virtual representation Within a threshold distance D the mapping of real to virtual hand is 1-to-1 However, once the hand moves beyond the threshold, the length of the arm $r_{virtual}$ is calculated according to the function $r_{virtual} = r_{real} + \alpha(r_{real} - D)^2$ (Figure 2 15) This allows seamless movement from the standard virtual hand, to an augmented version

Pointing techniques A variety of pointing techniques have been suggested for immersive environments Laser pointer, or ray-casting techniques (Mine, 1995, Poupyrev *et al*, 1998b) allow users to select objects with a ray emanating from their hand or index finger (Figure 2 16)

One of the problems with a standard ray casting technique, however, is that

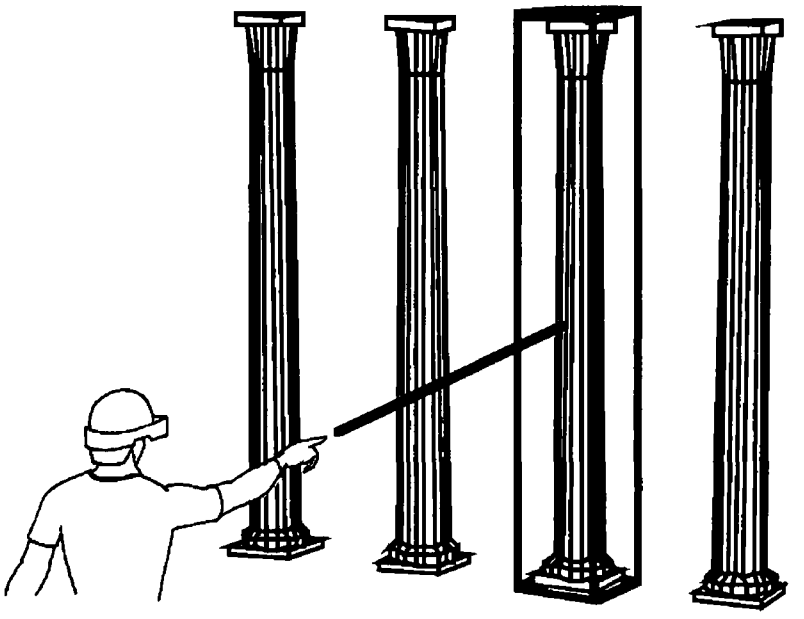


Figure 2 16 Selecting a distant object by pointing (From Mine, 1995)

as the distance of an object increases, small variations in the angle of the hand correspond to large movements at the far end of the ray Thus, selection of smaller distant objects is quite difficult An alternative to standard ray-casting, is the use of a torch metaphor Here, the ray has a cone like shape The conic shape allows for a larger catchment area when selecting, which facilities selection of distant objects However, with a larger catchment area the system may have to disambiguate between closely positioned objects which may be selected simultaneously

2 5 6 2D selection in immersive environments

Interestingly, although early interaction technique research focussed on the use of 2D devices to control 6-DOF movement in 3D (Bier, 1987), many interaction techniques now perform the reverse, limiting the DOF of 6-DOF input devices to just 2D

It should not be surprising that 2D interaction techniques are used in 3D Users, intimately familiar with 2D desktop interaction, are comfortable using

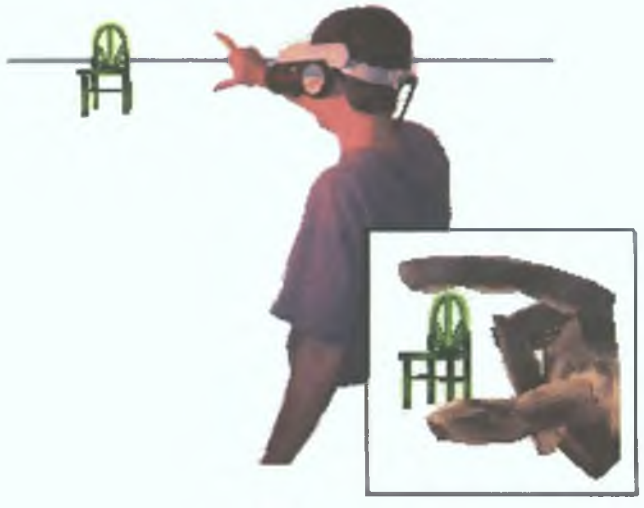


Figure 2.17: Head crusher technique (From Pierce *et al.*, 1997)

such techniques. Also, it has been noted that 3D interaction is difficult. The lag and resolution of sensing technology, and often the lack of stereoscopic vision to aid depth perception, mean that interaction in a virtual world is not as easy as in real life. Finally, and perhaps more importantly, there is little point in using 6-DOF input techniques, for tasks which may require only 2-DOF.

Pierce *et al.* (1997) describe a selection technique, which reduces the selection of objects to a 2D problem. Using their *head crusher* technique, objects are selected by positioning its 2D representation on the image plane between the thumb and forefinger (Figure 2.17). This is similar to the *crosshair*, method suggested by Mine (1995). Variations on the same theme include the lifting the desired object with the palm, or framing it with both hands (Pierce *et al.*, 1997).

2.5.7 Menu selection in immersive environments

Selection using menus is another 2D technique borrowed from the standard desktop metaphor. Again, the benefits include reduced DOF, and previous user experience. Menu selection is most commonly used in immersive environments for sys-

tem control such as mode changes However, it can be used for selection where other techniques may not be suitable Selection of objects which may be occluded or hidden, or choosing an object yet to be created from a list of potentials being just some of the possibilities

2D menus can be represented in various ways in 3D environments The simplest of these, is to overlay the menu in 2D on top of the user's image plane, allowing interaction using the *crosshair* described by Mine (1995)

Another alternative is to create menus which float inside the immersive environment, relative to the world rather then the user A useful technique is to map these to a paddle or tablet, which can be held in the non-dominant hand, while items are selected using either pointing techniques such as ray-casting, or by simply touching the tablet if it is touch sensitive The benefits of this technique is that, when attached to a tablet, the menu can be easily be moved out of view when it is not being used The downside is that the user must carry a tablet, which may restrict alternative interaction techniques When not attached to tablets, the drawback is that 3D ray-casting can be needed to perform what is essentially a 2D task, and users may have to first manoeuvre into an appropriate position to view or interact with it

Shaw and Green (1994) provide an alternative, simpler approach Their *ring menu* was designed in recognition of the fact that in many situations selection can be reduced to just 1-DOF Menu items were selected by rotating the ring until the desired item was in focus, and then issuing a select command (Figure 2 18) By reducing the DOF to just 1, ignoring up to 5-DOF of the user, 1-DOF menus ease the task of selection, allowing the user to concentrate on the accurate movement of just 1-DOF (contrast this to the 6-DOF, precision control necessary to select objects using ray-casting techniques)

Tulip menus (Three-Up, Labels in Palm) offer an effective technique for direct selection of items from a menu A simple approach to selecting objects from a list or menu, is to attach or assign each one to a finger Selection is then achieved

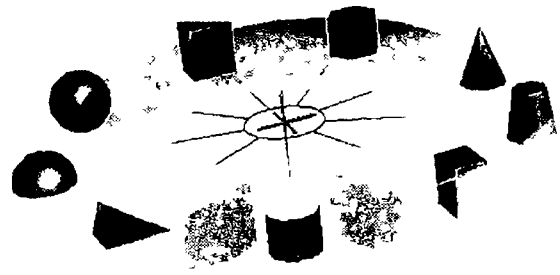


Figure 2 18 JDCAD's 1-DOF Ring Menu (From Hand, 1997)

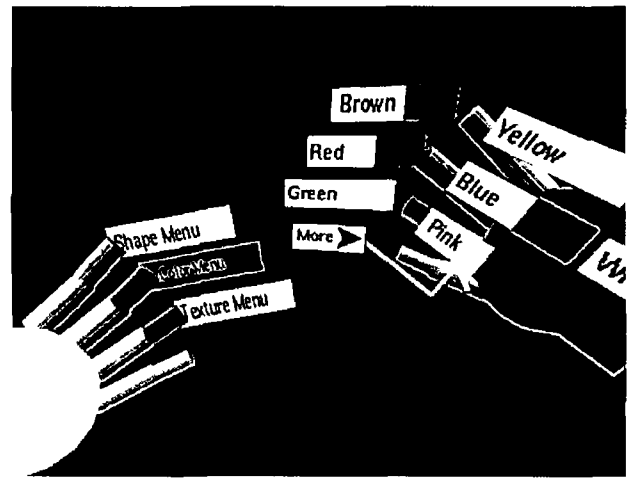


Figure 2 19 Tulip menu (From Bowman and Wingrave, 2001)

by flexing (or pinching in the case of (Bowman and Wingrave, 2001)) the finger with the desired menu item In case of menu lists greater then 4 (the system was originally designed for pinch gloves, where thumbs are required to pinch each finger), a *more* menu item can be assigned to the little finger

2 5 8 Miscellaneous selection techniques

Gesture provides a useful, if somewhat limited technique for selection Tulip menus offer one atypical example of this, where pinch gestures can select menu items In this situation, there is a clear visual mapping between the gesture and the object This is not usually the case, and users must remember the

meaning of each gesture Gesture is perhaps most powerful as a method of system control, where gestures are used to select commands, rather then as a technique for selecting objects Gestures can be used to select modes, or actions to perform upon objects The power of gestures is their direct 1-to-1 mapping The drawback of gestures is that users must learn and remember gestures, as there is typically no visual feedback of possible gestures Also, gestures can be misclassified, or misinterpreted by the system, either through badly formed gestures, or through accidental creation

Speech offers an attractive selection technique Like gesture, it provides a natural interaction with the environment and can have a direct one-to-one mapping However, as with gestures, the user must learn which commands can and cannot by understood by the system False recognition of background noise, or unintended utterances can also have unwanted effects

2 5 9 Conclusion

The key interaction tasks in immersive environments are selection, manipulation, travel and system control Interaction with a virtual keyboard can be viewed primarily as a selection problem Various selection techniques exist in immersive environments, from natural, full 6-DOF direct selection, to restricted 1-DOF menu selection Creating a taxonomy of selection offers a useful insight into the possible selection techniques

In Chapter 6 we will create a taxonomy for interaction with ambiguous keyboards Having created our taxonomy, we will map a set of possible gestures possible with the 5DT data glove, to produce possible interaction techniques for ambiguous text-entry

2 6 Summary

In this chapter we have reviewed the four main areas central to the use of ambiguous keyboards in immersive environments keyboard layout, prediction accuracy, gesture recognition and interaction techniques In the following chapters we will examine each area in greater detail In Chapter 3, we will examine the design of optimised keyboards for ambiguous text-entry In Chapter 4 we will explore the prediction accuracy of ambiguous keyboards, in particular we will examine the effects of language modelling on accuracy with standard and optimised keyboards In Chapter 5 we will tackle the problem of gesture recognition suitable for predictive text-entry Specifically, we will address the problem of sympathetic bending, and the associated recognition errors Finally, Chapter 6 will discuss the design of interaction techniques suitable for immersive text-entry with ambiguous keyboards Ultimately, any user interface must be subject to user testing, which can be used both for the identification of any design problems, and to compare and contrast potential designs Chapter 7 discusses the results of both formative and summative evaluations conducted

Although examined separately in the following chapters, many of these core issues are interlinked Chapter 4, which focuses on prediction accuracy, will be influenced by the accuracy of the optimised keyboards created in Chapter 3 Interaction techniques, designed in Chapter 6, will be influenced by both the gesture recognition capabilities, and prediction accuracy

Finally, Chapter 8 combines the results of all 5 chapters to provide a detailed methodology for predictive text-entry in immersive environments This methodology summarises the results of our experiments, and provides a set of recommendations for the use of ambiguous text-entry in immersive environments

Chapter 3

Optimisation of ambiguous keyboard layouts

3 1 Introduction

In Chapter 2, we discussed the previous research on keyboard layout optimisation The aim of most keyboard layout optimisation is to increase WPM Other aims include increased comfort and learnability For ambiguous keyboard layouts, increasing WPM is typically tackled by attempting to reduce ambiguity, and the need to resolve incorrect predictions

In this chapter we will focus on the optimisation of virtual ambiguous keyboards accessed using datagloves This will be tackled in two separate stages Firstly, we will examine the optimisation of dictionary-based ambiguous keyboard layouts This is typically performed by attempting to minimise the number of words with identical key-stroke mappings We will argue that any optimisation of keyboard layouts should take characteristics of word frequencies into account and we will perform experiments to show that keyboards which are designed accordingly perform significantly better then those which are not We will contrast keyboards designed to minimise clashing, as suggested by Oommen *et al* (1991), with keyboards optimised using uni-gram and bi-gram statistics Secondly, we

will optimise the key-to-finger mapping of ambiguous keys based on bi-action data calculated from user testing We will combine this data with bi-gram tables to produce keyboards optimised for expert users, and also to predict potential expert speed

3 2 Optimising dictionary-based ambiguous keyboards

As discussed in Chapter 2, optimisation of ambiguous keyboards is highly dependent on the interaction possible with the ambiguous keyboard Although reducing clashing is important, it is not always the most important factor This is evident in (Bahuman *et al*, 2000, Harbusch and Kuhn, 2003, Garbe, 2000, Kuhn and Garbe, 2001), where keyboard layouts were optimised for motor-impaired users Here users could not select each key independently, each ambiguous key was highlighted in rotation, with users pressing a single key to indicate that the highlighted key is the one intended Optimisation of these keyboards, therefore, is focussed on choosing a layout which would require as little cycling as possible, but also provides as few clashes as possible (as these also have to be rotated through)

In contrast, the ambiguous keyboard we propose for use in immersive environments allows the user to access each ambiguous key at any point, by flexing the matching finger Accordingly, the optimisation of our ambiguous keyboard layout is focussed solely on reducing clashing, which in turn reduces the user intervention needed when the system predicts the incorrect word

Similar work, optimising a telephone key-pad layout for word-level disambiguation, is tackled by Oommen *et al* (1991) Optimal keyboard designs are achieved by attempting to minimise the number of words with identical key sequence mappings For any given dictionary, each word must map to a sequence of key strokes With increased dictionary size, the potential for two or more words to map to the same sequence increases Oommen *et al* (1991) attempt to find a

keyboard which reduces clashing by ranking keyboards according to the number of individual sequences they create for a set dictionary Using a stochastic learning automata, they reduce the average number of clashes for their 1067 word dictionary to just 5 7

Their method may be formally described as

A is a finite alphabet and H, the finite dictionary, is a subset of the words over A^* Let $\boldsymbol{K}$ be a subset of the integers where, with no loss of generality, $\boldsymbol{K} = \{1, 2, \quad , K\}$ For all $i \in \boldsymbol{K}$, a set $C_i \subset A$ is associated, where the C_i are mutually exclusive and collectively exhaustive Here $\boldsymbol{K}$ represents the key count, while C_i represents the letters allocated to a single key

Let $\Pi = \{C_i\}$, which corresponds to a keyboard layout Thus the sequence of keys needed to enter enter a word X, $\Pi(X)$, will be the string Y, where if $X = x_1, x_2, \quad , x_N \in H$ then $Y = y_1, y_2, \quad , y_N$ where $y_i \in \boldsymbol{K}$ and $x_i \in C_{y_i}$ Also for of ease of expression we shall define H^Π to be the mapping of H due to Π Thus

$$H^\Pi = \{\Pi(X) \,|\, X \in H\} \tag{3 1}$$

Since more than one $X \in H$ may map to the same encoding string, the cardinalities of both H and H^Π need not be the same, such that $\left|H^\Pi\right| \leq |H|$ Thus the ambiguity of any keyboard layout, Λ, is

$$\Lambda = |H| - \left|H^\Pi\right| \tag{3 2}$$

Oommen *et al* suggest that an optimal keyboard design is one which minimises this ambiguity Λ for a given dictionary H However, by rating the fitness of a given keyboard according to this ambiguity value, the optimisation method employed fails to consider the frequency with which words in their dictionary occur in natural use, and thus underestimates the probability of clashing when the keyboards are used in practice

3 2 1 Word frequency characteristics and keyboard layout evaluation

The use of ambiguous keyboards for text-entry necessitates a method for disambiguating words when a collision occurs This is usually performed by a disambiguation engine, with the user ultimately completing the disambiguation process The method we shall focus on, and that is currently employed on most mobile phones, is as follows if clashing occurs, the words are ranked according to frequency, with the most frequent offered first The user rotates through alternatives if the first word offered is not their desired word

For this method of interaction, keyboards designed using Oommen's technique are sub-optimal For example, consider the dictionary of the most common 1000 words in the Brown Corpus If, as suggested by Oommen *et al*, keyboards are ranked according to how many individual sequences they have, a keyboard with 998 individual sequences would be ranked near optimal With only 4 words from 1000 clashing, one might assume that a user employing such a keyboard would be faced with ambiguous words with a probability of only 0 004 If however the 4 words that clashed happened to be the pairs {*the, and*} and {*of, to*}, the most common four words in the dictionary, whose combined frequencies represent over 20% of our 1000 word dictionary, then the user would in fact be presented with ambiguous words with a probability of 0 2, or once in every five words Although this example is clearly a worst-case scenario, it highlights the dangers of ignoring word frequency when considering alternative keyboard layouts

We argue that by simply ranking keyboard layouts according to the number of individual sequences it has for a given dictionary, the characteristics of the dictionary are ignored, ultimately leading to the creation of inferior keyboards This is due to the nature of word frequency and the phenomenon known as Zipf's law (Zipf, 1935) Zipf's law is a well-established empirical generalisation about word frequency which says that frequency is inversely proportional to rank

Plotting frequency against rank results in a doubly exponential curve (Gazdar, 1996) We further contend that a method of ranking keyboards which takes word frequency into account should lead to the creation of superior keyboard layouts

3 2 2 Evaluation using word frequency

We suggest that when evaluating keyboard layouts, both the frequency of words in the dictionary being used and the disambiguation process should be considered when attempting to create an optimal keyboard

Consider the previously described disambiguation algorithm, where clashing words are ranked according to their frequency In this scenario the user need only intervene if the word offered is incorrect Consider to clashing words X_1 and X_2, with probability $p(X_1)$ and $p(X_2)$ respectively, where $p(X_1) > p(X_2)$ The disambiguation algorithm is statistically likely to offer the correct word with a probability of $p(X_1)$ and the incorrect word with a probability of $p(X_2)$ Therefore the predicted accuracy of any given keyboard layout is the sum of the probabilities of unambiguous words, combined with the sum of the probabilities of the most frequent word in each of the clashing sets We can describe the technique for calculating this predicted accuracy more formally as

$p \quad H \rightarrow [0,1]$ where $p(X)$ = relative probability of X occurring in natural language

For each word X_0 we can determine the clash count $\delta(X_0)$ as

$$\delta(X_0) = |\{X \,|\Pi(X) = \Pi(X_0), X, X_0 \in H, X \neq X_o\}|$$

Let Q be the set of all clashing words, such that $Q = \{X \mid \delta(X) > 0, X \in H\}$, and G be the set of their corresponding sequences Thus $G = \{\Pi(X) \mid X \in Q\}$ where $G = \{Y_1, Y_2, \quad , Y_d\}$ and $d = |G|$

Then let Q_i define the set of words that map to a specific sequence Y_i for $i = 1 \quad d$

$$Q_i = \{X \mid \Pi(X) = Y_i, X \in Q, Y_i \in G\}$$

Thus the predicted accuracy P of any given keyboard may be calculated as the sum of the probabilities of unambiguous words (with a clash count of zero, thus $\delta(X) = 0$), plus the sum of the probabilites of the words in each clashing group Q_i with the highest probabilites

$$P = \sum_{X \in H} p(X)\sigma(X) + \sum_{i=1}^{d} \max_{\{X \in Q_i\}}(p(X))$$

where (3 3)

$$\sigma(X) = \begin{cases} 1 \text{ if } \delta(X) = 0 \\ 0 \text{ otherwise} \end{cases}$$

As an example consider a small dictionary of words and their relative probabilities, {*the* (0 3), *of* (0 15), *and* (0 15), *to* (0 1), *a* (0 1), *in* (0 1), *is* (0 1)} For a given keyboard, our dictionary might map to the following sequences {*324, 12, 324, 31, 3, 12, 12*} respectively In this case the words *the*, *of*, *and*, *in*, and *is* are all words which clash, and are thus part of Q The set G would consist of {*324* , *12*} With Q_1 = {*the*, *and*} and Q_2 = {*of*, *in*, *is*} The predicted accuracy of the keyboard for this dictionary would then be calculated as follows the sum of the probabilities of words which don't clash (*to, a*), plus the sum of the word with the highest probability in Q_i (*the*), and that in Q_2 (*of*), which gives us 0 65 This means that, during the course of typing, the first word offered by our disambiguation system is likely to be correct 65% of the time

We contend that using this alternative method for rating keyboards, we can rate a keyboard's performance with higher accuracy and thus create significantly

better keyboards[1]

3 2 3 Bi-gram prediction and word co-occurrence

As discussed in Chapter 2, language modelling may be used to aid the prediction of ambiguous words based on their probability in a large corpus We will examine the benefits of higher-order language models in greater detail in Chapter 4, however, their use in the design in optimal keyboards would seem a natural progression from word frequency Word frequencies only indicate the likelihood of a word occurring in our text, whereas bi-gram probabilities tell us the probability of a word occurring given a previous word Thus, given a clash, a frequently seen word may be ranked lower then a less frequent one if the less frequent word is more likely to appear after the previous word If two words with high frequency clash, then ranking a keyboard with Equation 3 3, may be inaccurate, as a bi-gram language model may predict the correct word with higher accuracy then suggested by Equation 3 3 This is because the *co-occurrence* of the words might be low The two words might rarely be seen after similar words, and are thus unlikely to be confused by a bi-gram prediction system *Confusion probability* P_c is used to describe the estimate of the probability that w_1 and w_1' may be found in the same context (Dagan *et al*, 1999), and can be calculated as follows

$$P_c(w_1' \mid w_1) = \sum_{w2} \frac{P(w_1 \mid w_2)P(w_1' \mid w_2)P(w_2)}{P(w_1)}$$

Applying this concept to our evaluation technique, a more accurate method of predicting keyboard performance, is to consider how likely two words are to occur in the same sentence, and from this calculate how likely the language model is to predict incorrectly If we consider the bi-gram pairs "go home" and "be good", where the words *good* and *home* clash for an alphabetic keyboard, with

[1]This technique was also suggested independently by Cardinal and Langerman (2004) However, the benefit of using such a technique over alternatives was not examined

example probabilities of each shown in Example 3 1, a bi-gram language model will predict the correct word for both our pairs However, should the bi-gram "be home", or 'go good" by typed by a user, the bi-gram language model will predict the wrong word This is likely to occur with a probability of $P(\text{good} \mid \text{go})P(\text{go})$, and $P(\text{home} \mid \text{be})P(\text{be})$ respectively

```
P(go) = 0 1
P(be) = 0 2
P(home | go) = 0 2
P(home | be) = 0 1
P(good | go) = 0 01
P(good | be) = 0 2
```

Example 3 1 Sample bi-gram probabilies

Therefore, a more accurate estimate of the cost of a pair clashing can be found by summing the probabilities of all instances where the language model would predict the wrong word For any clashing pair, w_1, w_1', the probability that the language model will predict incorrectly is

$$P_w(w_1, w_1') = \sum_{w_2} \min(P(w_1 \mid w_2)P(w_2), P(w_1' \mid w_2)P(w_2)) \tag{3 4}$$

This equation allows us to create a confusion matrix, which, for each word pair, contains the probability our language model will predict incorrectly should they clash If we define A as the set of all pairs of words in Q_i, $A = \{(X, Y) \mid X \in Q_i, Y \in Q_i\}$, then, for any given keyboard layout, we can predict the likely performance as follows

$$P = 1 - \frac{1}{2}\sum_{i=1}^{d} \sum_{\{A \in Q_i\}} P_w(X, Y) \tag{3 5}$$

3 2 4 Optimised keyboard layout creation

Once an evaluation metric has been decided, the optimisation of a keyboard becomes a NP-complete search problem (Oommen *et al*, 1991) Various search heuristic algorithms have been employed to search for optimised keyboard lay-

outs simulated annealing, n-opt, genetic, and metropolis algorithms to name a few (Zhai *et al* , 2000, Light and Anderson, 1993, Lesher and Moulton, 2000) During the course of our experiments we used simulated annealing Van Laarhoven and Aarts (1987) to create optimal keyboards However, any of the previously mentioned optimisation techniques would likely have sufficed N-opt, as described in (Lesher and Moulton, 2000), was also tested giving near identical results

We also explored the use of a slightly adapted version of simulated annealing for our search algorithm, which could be called guided simulated annealing With standard simulated annealing, the keys to be switched are typically chosen at random Our adapted technique attempted to guide the system by choosing the key which was causing the most ambiguity, and switching it with a random key This was inspired by Oommen *et al* (1991), who used a similar, but slightly more elaborate system

For the sequence-based system, the adapted version guides the selection of the letter to be changed, by choosing the letter which has the most clashes For the sequence based evaluation metric, for every set of clashing words, the letters which caused the clash are *penalised* A running count is kept of the clashes each letter causes Thus, for every pair of clashing words, 1 is added to the clash count of any letter which do not match For example, if two words, *cake* and *late* clash, the letters c and l clash, as do the letters k and t, thus each of these would be punished The letters a and e remain unpunished The letter which has been penalised the most, and thus causes the most clashes in the dictionary is chosen as the letter which should be changed

For the statistical evaluation metric, for each clashing pair, the clashing letters in the word with the lowest probability are penalised according to that probability Thus, if *cake* and *late* clash, and each have a probability of 0 08 and 0 02 respectively, then only the letters l and t are penalised, with 0 02 each Again, the letter which has been penalised the most is chosen as the letter which should be changed In this case, this will not necessarily be the letter which causes the

most clashes, but rather it is the letter among less likely clashing words which is statistically most likely to clash

Our adapted or guided simulated annealing technique was out-performed by standard simulated annealing for both statistically based and sequence based keyboards It is likely the large early gains in performance made by the guided technique caused the search get *stuck* in a local minimum, In contrast, the standard technique although making slower initial progress, performed better in the long run Given the performance of the guided statistical and sequence based techniques, a guided technique was not attempted for the co-occurrence method

3 2 5 Experiment

Having defined three separate metrics with which to estimate the potential accuracy of keyboard layouts, we conducted an empirical evaluation of each metric to examine the performance of resulting keyboards during actual use This was achieved by creating separate keyboard layouts using each of the three estimation metrics to predict performance, and then testing the actual performance of resulting layouts on a large text corpus We used the simulated annealing algorithm to create 90 optimised keyboard layouts, with each of the evaluation metrics used to estimate the performance of 30 keyboards

The algorithms used to implement Equation 3 1,3 3 and 3 5, can be seen in Algorithm 3 2 , 3 3 and 3 4 respectively

```
{ Algorithm to calculate the sequence count for a keyboard layout }
BEGIN
 calculate key-sequence for each word in dictionary
 sort dictionary by key-sequence
 FOR all words in dictionary
   IF( dictionary[i] sequence != dictionary[i-1] sequence )
      sequence_counts++
   ENDIF
 ENDFOR
END
```

Algorithm 3 2 Sequence count algorithm

```
{ Algorithm to estimate the accuracy of a keyboard layout based on word frequency}
BEGIN
 calculate key-sequence for each word in dictionary
 sort dictionary by key-sequence and then frequency
 FOR each word i in dictionary
   IF( dictionary[i] sequence = dictionary[i-1] sequence )
      probability_of_clash += dictionary[i] frequency
   ENDIF
 ENDFOR
END
```

Algorithm 3 3 Statistical probability algorithm

```
{ Algorithm to estimate keyboard accuracy based on word co-occurrence }
BEGIN
 calculate key-sequence for each word in dictionary
 sort dictionary by key-sequence and then frequency
 FOR each word i in dictionary
   next = 1
   WHILE (dictionary[i] sequence = dictionary[i+next])
     prob_of_clash += confusion_prob[dictionary[i]][dictionary[i+next]]
     next++
   ENDWHILE
 ENDFOR
END
```

Algorithm 3 4 Co-occurrence probability algorithm

Procedure For our experiments we used the British National Corpus (BNC), a "100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of current British English, both spoken and written" (BNC, 1995) A two million word training section of the BNC corpus was selected at random From this, unigram and bi-gram statistics were calculated using the SRILM toolkit (Stanford Research Institute, 1995) A dictionary of the most frequent two thousand words in the corpus was then created We used simulated annealing to create three sets of optimal keyboards with 30 keyboards in each Each set used a different equation to estimate keyboard ambiguity The first set, which we shall refer to as the *sequence set*, was created using Equation (3 2) to minimise ambiguity The second set, which we shall refer to as the *frequency set*, was created by considering word frequency as described in Equation (3 3) Finally the third *co-occurrence*

set, was created by using Equation (3 5) to estimate keyboard performance

Having created three sets of keyboards using each of our estimation metrics, we then simulated user input, to evaluate the actual accuracy of the keyboards when used in practice Using an engine to simulate use of the keyboard by a user, we tested each keyboard on a one million word test corpus, taken from a separate section of the BNC Tests were conducted using bi-gram word prediction, as described in Equation 2 1

Hypothesis The main hypothesis of the experiments were

- The use of word frequency information during keyboard creation would result in keyboards which significantly outperformed those designed to minimise the clash count
- The use of co-occurrence information would result in yet further improvements in prediction performance

3 2 6 Results

The results in Figure 3 1 show the performance of the keyboards, when used to type a 1 million word test corpus Values on the vertical axis refer to the percentage of words offered correctly first time A 1-way ANOVA test confirms that there is a significant difference between the keyboard estimation techniques ($F_{(2\,87)} = 30\,871$, $p < 0\,0005$) Comparing individual techniques, we found that the frequency-based set was significantly better then the sequence-based set ($t = 5\,352$, $df = 87$, $p < 0\,0005$, one-tailed) This confirmed the hypothesis We also found that the co-occurrence set performed significantly better then the frequency-based set ($t = 2\,307$, $df = 87$, $p = 0\,0165$, one-tailed) Again this confirmed the hypothesis

The outliers visible in the sequence-based set highlight the problem whereby, due to word frequency characteristics, a keyboard with a low sequence count will

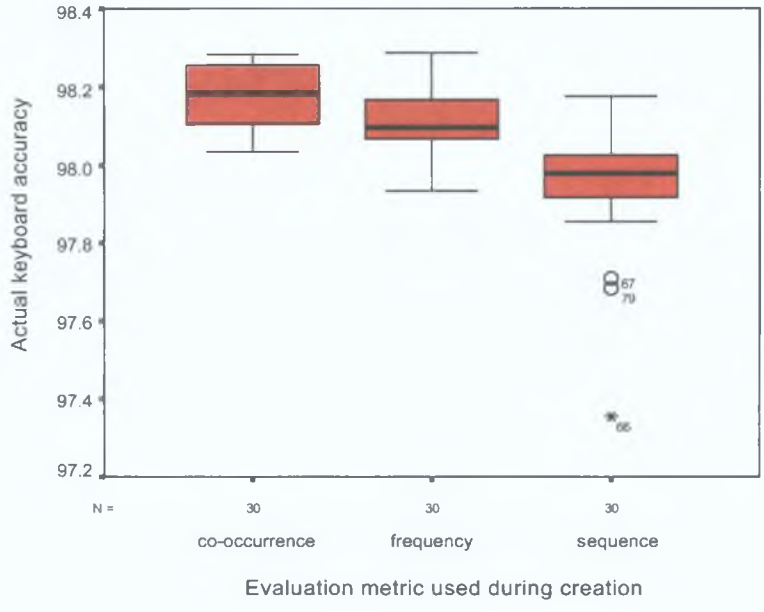


Figure 3.1: Box-plot of keyboard performances.

perform poorly in practice if the words which do clash occur with high frequency.

3.2.7 Conclusions

We have compared three evaluation methods for the creation of optimal keyboards designed for word-based disambiguation. We have argued that any method of evaluating keyboards based on a dictionary of words should take language characteristics into account. Through empirical testing we have shown that creating keyboards based on word frequency statistics offers significant improvements over sequence maximising techniques. We have also shown that further significant improvements can be made through the use of co-occurrence data collected from bi-gram probabilities.

3 3 Optimising keyboard layouts based on bi-gram / bi-action tables

One of the design criteria for the Dvorak keyboard was to increase frequency of alternate-hand key-strikes Kinkead (1975) showed that alternate-hand bi-actions were fastest, followed by same-hand bi-actions, and finally same-finger bi-actions This is because when striking a key with one hand, the other hand can move into position and strike almost simultaneously In contrast, the slowest motion, same-finger bi-actions, has no simultaneous motion This can be seen in Figure 2 4, where keystroke times for a QWERTY keyboard are shown (From Kinkead, 1975)

Hughes *et al* (2002) suggest the use of bi-action tables to create optimised layouts for a PDA and stylus, and predict expert burst speed for any given keyboard layout Using a similar technique to Hughes *et al* (2002), we will create bi-action tables for all 100 possible finger combinations, and, using this data, predict expert speed for our system for various keyboard layouts We will also attempt to further optimise our ambiguous keyboard by mapping the fastest bi-actions to the most common bi-grams, by changing the keyboard layout, but keeping the letter to key mappings

3 3 1 Creating a bi-action table for our ambiguous virtual keyboard

Unlike the bi-action tables of Hughes *et al* (2002) and Kinkead (1975), our proposed bi-action table will not contain an entry for every letter As our keyboard is ambiguous, our bi-action table will only contain 10×10 entries one for each finger combination

Bi-action times of 5 participants wearing 5DT datagloves were measured The gloves were calibrated for each user before experiment This involved measuring the range of movement of each individual finger, and scaling the flexion

Figure 3.2: Bi-action experiment

readings returned by the dataglove accordingly. Participants were shown two virtual hands. The numbers *1* and *2* appeared above the fingers to be timed, indicating the desired bi-action order. When users flexed the first finger, the colour of the number *1* turned from red to green. Timing of the bi-action began when the first finger was flexed, and ended when the second finger was flexed. At this point the number *2* briefly changed from red to green. A new bi-action pairing was then shown. If the user flexed the wrong finger for the first of the bi-action pair, it was simply ignored by the system and the number *1* remained red. If the user flexed the wrong finger for the second of the bi-action pair, the system reset the timer, and reset the colour of the number *1* to red.

Participants were instructed to first make note of both fingers in the required pair before attempting the bi-action, rather then flexing the first and then visually scanning for the second. It was explained that only the timing between the flexion of the first and second fingers was timed. Occasionally the second finger flexion of the bi-action pair was too weak, which resulted in no recognition from the system. Participants initial instinct was to flex the finger again with more exaggerated flexion, this resulted in a false bi-action time, as it represented the time to complete the bi-action, plus the time to realise the system hadn't recognised the second flexion, and flex the second finger again. Participants were told to reset the timing by making an incorrect gesture should such an error oc-

	L_{little}	L_{ring}	L_{middle}	L_{index}	L_{thumb}	R_{thumb}	R_{index}	R_{middle}	R_{ring}	R_{little}
L_{little}	0 234	0 241	0 247	0 190	0 220	0 170	0 147	0 184	0 230	0 157
L_{ring}	0 210	0 237	0 190	0 241	0 224	0 190	0 163	0 157	0 184	0 177
L_{middle}	0 267	0 190	0 241	0 224	0 221	0 210	0 144	0 127	0 200	0 160
L_{index}	0 220	0 204	0 177	0 310	0 221	0 211	0 113	0 160	0 177	0 160
L_{thumb}	0 247	0 210	0 261	0 177	0 284	0 160	0 177	0 190	0 194	0 204
R_{thumb}	0 197	0 157	0 147	0 154	0 127	0 203	0 187	0 204	0 214	0 194
R_{index}	0 190	0 113	0 157	0 153	0 131	0 217	0 197	0 170	0 227	0 193
R_{middle}	0 170	0 174	0 154	0 191	0 113	0 217	0 160	0 204	0 257	0 170
R_{ring}	0 190	0 107	0 124	0 140	0 097	0 220	0 210	0 200	0 287	0 190
R_{little}	0 134	0 141	0 184	0 177	0 180	0 271	0 184	0 224	0 184	0 217

Table 3 1 Average bi-action values

cur, and repeat the correct bi-action from the beginning Nevertheless, despite this instruction, users still occasionally repeated the unrecognised gesture instinctively before they realised their mistake Users were presented with all possible 100 bi-actions in random order They repeated the test 3 times Finally, the minimum bi-action times for each user was recorded The minimum time was chosen taken in favour of the average time in order to minimise the impact of the previously noted errors These minimum times were then averaged across all 5 participants to create our final bi-action table (Figure 3 1) The bi-action times recorded have similar characteristics as those recorded by Kinkead (1975) However, due to the larger movements needed for key-press gestures, the times are slower In particular alternate-hand bi-actions are faster then same-hand, which are in turn faster then same-finger bi-actions (Figure 3 2)

3 3 2 Rearranging optimised ambiguous keyboards based on bi-action tables

Having created bi-action tables for all possible finger combinations, we can predict the expert typing speed for any keyboard layout through the use of letter bi-gram frequencies of the English language We can also rearrange the keys on the optimised ambiguous keyboard we created using Equation 3 3, to maximise the

	L_{little}	L_{ring}	L_{middle}	L_{index}	L_{thumb}
alternate hand	0 176	0 138	0 153	0 163	0 130
same hand	0 236	0 216	0 223	0 228	0 234
same finger	0 234	0 237	0 241	0 310	0 284

	R_{thumb}	R_{index}	R_{middle}	R_{ring}	R_{little}	Average	Kinkead
alternate hand	0 188	0 149	0 164	0 197	0 172	**0 163**	**0 132**
same hand	0 226	0 188	0 200	0 234	0 193	**0 218**	**0 168**
same finger	0 203	0 197	0 204	0 287	0 217	**0 241**	**0 230**

Table 3 2 Inter-hand and finger bi-action times

potential expert speed

Given a character map K, a table of bi-gram possibilities P, and an empirical bi-action table A, peak expert text-entry rate $R(K, P, E)$, in characters per second (CPS), is given by (from Hughes *et al* (2002))

$$R(K, P, E) = \frac{1}{\sum_{\alpha,\beta} P(\alpha, \beta) A(K(\alpha), K(\beta))} \tag{3 6}$$

Given the predicted CPS, the predicted words per minute (WPM) is calculated by multiplying by 60 seconds per minute and dividing by 5 characters per word (MacKenzie *et al*, 1999), thus

$$WPM = \frac{CPS \times 60}{5}$$

We used the bi-gram frequency table of (Soukoreff and MacKenzie, 1995), Table 3 3, for consistency as this table is commonly used for text-entry rate predictions, and is one of the tables used by Hughes *et al* (2002) Using Equation 3 6, we calculated the predicted expert speed for QWERTY, alphabetic, DVORAK, and minimum ambiguity keyboard layouts (Table 3 4) We then used the n-opt algorithm to search for an optimised keyboard layout, $Opti_{Best}$(Figure 3 3), which attempted to maximise predicted expert speed, while maintaining the minimum ambiguity characteristics imparted from Equation 3 3 Also shown in Table 3 4

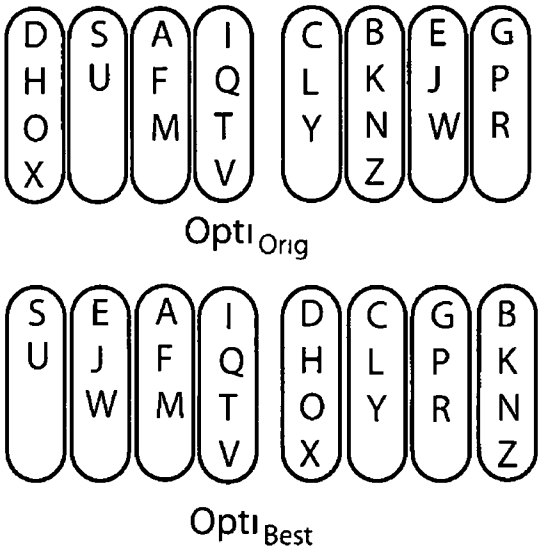


Figure 3 3 Optimised keyboards $Opti_{Orig}$(top), before bi-action optimisation, and $Opti_{Best}$(bottom) after bi-action optimisation

are the times of the original optimised keyboard layout, $Opti_{Orig}$, as created by Equation 3 3, with no knowledge of bi-action speeds, and a worst case keyboard layout, $Opti_{Worst}$, where the optimal keyboard is arranged to produce the lowest possible WPM The table shows that over 5 WPM can potentially be lost in expert speed when creating a minimum ambiguity keyboard if no attention is paid to the finger mappings The predicted expert typing speed of 65 WPM is roughly half that of expert typists on regular keyboards (135 WPM) (Card *et al* , 1983) This is most likely due to the larger hand movements necessary for accurate gesture recognition with the 5DT datagloves

3 3 3 Discussion

As the keyboard layout is ambiguous, with several letters assigned to each key, many bi-gram combinations will have the same bi-action times, leading to less variation between keyboards Nevertheless, our bi-action table gives a good indication of potential expert speed given sufficient practice These times represent potential peak expert speed, which are likely to be slightly unrealistic Unlike

1st/2nd	a	b	c	d	e	f	g	h	i	j	k	l	m	n
a	2	144	308	382	1	67	138	9	322	7	146	664	177	1576
b	136	14	0	0	415	0	0	0	78	18	0	98	1	0
c	368	0	13	0	285	0	0	412	67	0	178	108	0	1
d	108	1	0	37	375	3	19	0	148	1	0	22	1	2
e	670	8	181	767	470	103	46	15	127	1	35	332	187	799
f	145	0	0	0	154	86	0	0	205	0	0	69	3	0
g	94	1	0	0	289	0	19	288	96	0	0	55	1	31
h	1164	0	0	0	3155	0	0	1	824	0	0	5	1	0
i	23	7	304	280	189	56	233	0	1	0	86	324	255	1110
j	2	0	0	0	31	0	0	0	9	0	0	0	0	0
k	2	0	0	0	337	0	0	0	127	0	0	10	1	82
l	332	4	6	289	591	59	7	0	390	0	38	546	30	1
m	394	50	0	0	530	5	0	0	165	0	0	4	28	4
n	100	2	98	1213	512	5	771	5	135	6	83	80	0	54
o	65	67	61	119	34	80	9	1	88	3	123	218	417	598
p	142	0	1	0	280	1	0	24	97	0	0	169	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	289	10	22	133	1139	13	59	21	309	0	53	71	65	108
s	196	9	47	0	626	0	1	328	214	0	57	48	31	16
t	259	2	31	1	583	1	2	3774	252	0	0	75	1	2
u	45	53	114	48	71	10	148	0	65	0	0	247	87	278
v	27	0	0	0	683	0	0	0	109	0	0	0	0	0
w	595	3	0	6	285	0	0	472	374	0	1	12	0	103
x	17	0	9	0	9	0	0	0	10	0	0	0	0	0
y	11	10	0	0	152	0	1	1	32	0	0	7	1	0
z	3	0	0	0	26	0	0	0	2	0	0	4	0	0
SPACE	1882	1033	864	515	423	1059	453	1388	237	93	152	717	876	478

1st/2nd	o	p	q	r	s	t	u	v	w	x	y	z	SPACE
a	1	100	0	802	683	785	87	233	57	14	319	12	50
b	240	0	0	88	15	7	256	1	1	0	13	0	36
c	298	0	1	71	7	154	34	0	0	0	9	0	47
d	137	0	0	83	95	3	52	5	2	0	51	0	2627
e	44	90	9	1314	630	316	8	172	106	87	189	2	4904
f	429	0	0	188	4	102	62	0	0	0	4	0	110
g	135	0	0	98	42	6	57	0	1	0	2	0	686
h	487	2	0	91	8	165	75	0	8	0	32	0	715
i	88	42	2	272	484	558	5	165	0	15	0	18	4
j	41	0	0	0	0	0	56	0	0	0	0	0	0
k	3	1	0	0	50	0	3	0	0	0	8	0	309
l	344	34	0	11	121	74	81	17	19	0	276	0	630
m	289	77	0	0	53	2	85	0	0	0	19	0	454
n	349	0	3	2	148	378	49	3	2	2	115	0	1152
o	336	138	0	812	195	415	1115	136	398	2	47	5	294
p	149	64	0	110	48	40	68	0	3	0	14	0	127
q	0	0	0	0	0	0	66	0	0	0	0	0	0
r	504	9	0	69	318	190	89	22	5	0	145	0	1483
s	213	107	8	0	168	754	175	0	32	0	34	0	2228
t	331	0	0	187	209	154	132	0	84	0	121	1	2343
u	3	49	1	402	299	492	0	0	0	1	7	3	255
v	33	0	0	0	0	0	1	0	0	0	11	0	0
w	264	0	0	35	21	4	2	0	0	0	0	0	326
x	1	22	0	0	0	23	8	0	0	0	0	0	21
y	339	16	0	0	81	2	1	0	2	0	0	0	1171
z	2	0	0	0	3	0	0	0	0	0	3	9	2
SPACE	721	588	42	494	1596	3912	134	116	1787	0	436	2	0

Table 3 3 Bi-gram frequency table (from (Soukoreff and MacKenzie, 1995))

Keyboard	CPS	WPM
$Opti_{Best}$	5 490	65 876
QWERTY	5 401	64 810
DVORAK	5 319	63 828
Alphabetic	5 266	63 188
$Opti_{Orig}$	5 401	61 868
$Opti_{Worst}$	5 044	60 533

Table 3 4 Predicted expert typing speed

typing on a regular keyboard, virtual typing requires more pronounced deliberate movements in order for the typing gestures to be recognised (this will be discussed in greater detail in Chapter 5) This results in higher exertion and thus fatigue with prolonged use Of all users tested, the fastest, with a potential peak expert speed of 111 WPM, expressed a belief that they would be unable to sustain such typing speeds for very long

3.4 Conclusions and recommendations

With its strong bias to index fingers, the QWERTY keyboard layout is not well suited to ambiguous text-entry Furthermore, experiments by Bowman *et al* (2001a) indicate that touch-typing skills do not transfer well to immersive virtual keyboards Thus, optimised keyboards designed to reduce ambiguity offer an attractive alternative to the QWERTY keyboard layout

In this chapter we have suggested two alternative techniques for producing optimised keyboards The first technique used word frequency to predict keyboard performance The second technique used word co-occurrence data based on bigram frequencies to predict likely performance Through empirical tests, we have shown that keyboards designed with both techniques perform significantly better then keyboards designed to simply maximise the individual sequence count The second technique, based on word co-occurrence, proved the most accurate for keyboard design

Through the use of bi-action tables, we have arranged the placement of these keys to optimise expert WPM These tables were also used to predict expert user typing speed, which indicated burst speeds of over 60 WPM could be achieved with sufficient practice

In practice, optimised keyboard layouts offer one solution for improved text-entry They are only one piece in the jigsaw The use of language modelling to predict words, and the interaction techniques used to select them will also play

a significant part in the relative benefits of any keyboard layout Thus, in the following chapters we will examine the benefits of optimised keyboard layouts when used in conjunction with improved language modelling, with various word selection techniques We will also examine the effectiveness of seemingly foreign, optimised keyboard layouts when compared to the more familiar QWERTY and alphabetic keyboard layouts

Chapter 4

Prediction accuracy

4 1 Introduction

In Section 2 2 6, we discussed the prediction techniques possible when using ambiguous keyboards These include statistical, syntactical, and context sensitive techniques Each of these leverage previous knowledge to predict future words However, in Section 2 2 7, we also discussed factors which affect the potential accuracy of any prediction system These include the ambiguous keyboard layout employed, and the interaction techniques possible with any list of predicted or complete words offered to the user

In this chapter we will consider the effects of language modelling on word prediction and completion accuracy We will explore the effects of increased language model training size and order Coupled with this, we examine the effects of keyboard layout and interaction style We will contrast QWERTY, alphabetic and optimised keyboard layouts, and examine the impact of iterative and direct selection techniques

Section 4 2 2 will examine word prediction accuracy, contrasting the clash-count of keyboards, the prediction accuracy as language model training size and order increases, and effects of prediction list length and selection style Section 4 2 3 will examine word completion accuracy We will examine the accuracy

as language model training size and order increases, and contrast the potential savings possible with alternative list lengths and selection styles

Finally, we will make recommendations based on our findings, which will aid designers in the selection of potential keyboard layouts, language models, interaction techniques and word lists lengths for both word prediction and completion

4 2 Experiments

4 2 1 Design and implementation

We undertook to explore the effects of keyboard layout, dictionary size, and N-gram order and training size on the effect of word prediction accuracy We also examined the effects of these on word completion accuracy A word prediction system was designed that uses a language model to rank predicted words The word prediction system consists of two main sections, firstly, a dictionary lookup, which identifies matching words, and secondly a language model lookup, which orders potential words according to their probability in the language model The language model was trained on the BNC (1995) in training sizes ranging from 500,000 words, to 25 million To predict likely real-world accuracy, a text-entry engine was then built which examines the accuracy of the prediction system when used to predict an unseen test corpus of 250,000 words Each keyboard layout was tested with increasing language model training size and order (uni, bi and tri-gram)

Dictionary lookup The dictionary lookup is implemented with a tree structure Each node on the tree contains one child for each key on the ambiguous keyboard and a list of words Words were assigned to a particular node on the tree structure by converting them to their corresponding key-sequence, traversing the tree according to that sequence, and inserting the word on the final node Thus for example, the word *and*, which corresponds to the sequence *152* on an alpha-

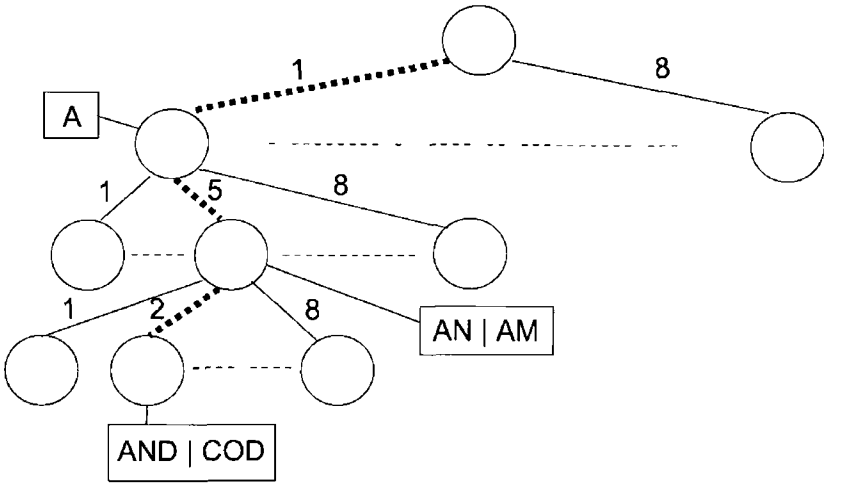


Figure 4 1 Dictionary tree

betic keyboard, would be stored by choosing the first child of the root, followed by the fifth and second children respectively (see Figure 4 1) Thus, the word *cod*, which has the same key-sequence, would be assigned to the same node When a dictionary lookup is performed all matching words in the dictionary are returned by simply traversing the tree according to the key-sequence entered Once a list of possible words is found, it is sorted using our language model

Language Model lookup We use language modelling to order the candidate words offered Uni-gram, bi-gram and tri-gram language models are tested We use Good-Turing discounting, combined with non-linear backoff for unseen higher order N-grams (Katz, 1987) The SRILM toolkit (Stanford Research Institute, 1995) is used to create various ARPA-standard format files (Example 4 2) containing language model statistics based on our training texts These files contain the probability of all N-grams seen in our training texts, as well as the backoff values of lower order N-grams Using these values the probability of a word is calculated using Equation 2 1 (Page 26) The procedure for calculating word probability is listed in Algorithm 4 1

```
{ Calculating the probability word n (wdn) given wdn-1 and wdn-2        }
{ <UNK> refers to the probability of an unknown word                    }
{ Algorithm adapted from CMU Toolkit ARPA file discription              }
{ bo_wt_*() = backoff weight                                            }

p(wdn|wdn-2,wdn-1)= if(trigram exists)   p_3(wdn-2 wdn-1,wdn)
   else if(bigram wdn-2,wdn-1 exists)    bo_wt_2(wn-2 wn-1)*p(wdn|wdn-1)
   else                                  p(wdn|wdn-1)

p(wdn-1|wdn-2)     = if(bigram exists)   p_2(wdn-2,wdn-1)
                     else                bo_wt_1(wdn-2)*p_1(wdn-1)

p(wdn-2)           = if(unigram exists)  p_1(wdn-2)
                     else                p_1(<UNK>)
```

Algorithm 4 1 Calculating word probability

Each N-gram is stored with its probability and backoff weight in a hash table Given a list of ambiguous words, and previous N-1 words, the probability of each word is calculated using Algorithm 4 1 The list of ambiguous words is then ranked and offered in order according to their probability

For word completion, the process is similar while typing is in progress, the predicted word is returned by traversing the dictionary tree based on the current key-sequence However, for word completion, the list of possible complete words corresponds to all words below the current node This list is returned and sorted in the same manner as predicted words Complete words are offered after a depth of 2 nodes has been reached when typing

Text-entry engine The performance of word prediction and the associated language model is tested on unseen text using a text-entry engine which simulates user text-entry Given a keyboard layout, and test text, the engine coverts each word into its corresponding key-sequence This is passed to the prediction system, which returns an ordered list of potential words The list is then checked to see if it contains the intended word Using this system, the likely accuracy of any language model, or keyboard layout, in practice can be quickly evaluated

When testing word completion accuracy the system offers one key at a time to the prediction system, and accepts correct complete words as soon as they are

```
\data\
ngram 1=29802
ngram 2=246824
ngram 3=36592

\1-grams
-5 7253 <UNK>   0 0000
-5 4243 abc     -0 1159

-5 7253 abdomen -0 1214
-5 0263 abducted        -0 1174


-4 9472 zurich  -0 2679

\2-grams
-3 3549 a a 0 0000
-3 6051 a about 0 0051
-3 7533 a academic 0 0000
-4 6486 a acquisition 0 0000


-1 3573 zurich upbringing 0 0000

\3-grams
-2 0099 a <comma> but
-1 2775 a <period> no
-0 7034 a about the
-0 7034 a basic law


-0 4024 zoology of the

\end\
```

Example 4 2 Example of the APRA file format All probabilites and back-off weights are given in log10 form

offered As a consequence, word completion statistics represent optimal potential performance This may not necessarily reflect actual use, where users may not initially notice a correct complete word has been offered, or may choose to ignore it

4 2 2 Word prediction accuracy

Clash count

One of the central factors affecting word prediction accuracy is the keyboard layout For any set dictionary, the assignment of letters to keys on an ambiguous keyboard will cause variations in the number of words which map to the same sequence of key-presses As the size of the dictionary increases, so does the clash count Figure 4 2 shows the effect of layout on the percentage of words clashing for our 3 keyboard layouts with increasing dictionary size As the size of the dictionary increases, the strong bias for index fingers on the QWERTY keyboard causes it to have the highest clash count An important value in the design of any ambiguous system is the *maximum sequence count* This value refers to the maximum number of words which map to any one sequence of key-presses and represents the maximum number of words the user may need to iterate through in order to select their desired word Figure 4 3 shows the increase in maximum sequence count as dictionary size increases Here we can see that QWERTY has a significantly higher maximum sequence count as the dictionary increases, more than double that of the alphabetic layout This value is significant, as it highlights potential frustration with the system if a QWERTY layout is employed scrolling through 55 words is likely to cause irritation to even the most patient user Although clash count and maximum sequence count give a worst-case scenario of prediction accuracy, they provide useful insights for the design of interaction techniques because they highlight the extreme examples any system must be capable of handling However, the practical prediction accuracy of the system is better shown through the use of experiments simulating its use in practice

Percentage of words guessed correctly

The T9 prediction system, employed on most modern mobile phones, has a dictionary size of 9025 words (Silfverberg *et al* , 2000) Words are sorted with a

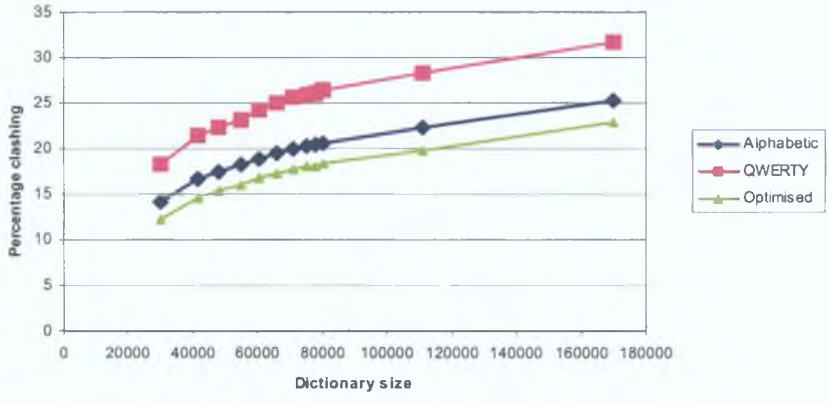


Figure 4.2: Increase in percentage of words clashing as dictionary size increases.

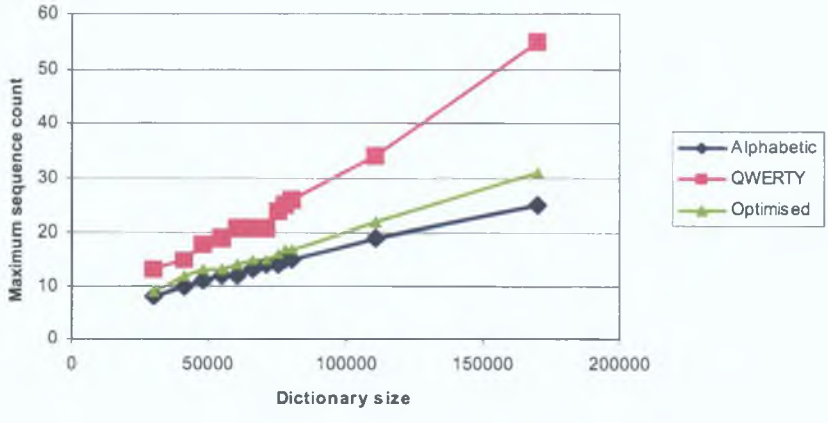


Figure 4.3: Increase in maximum sequence count as dictionary size increases.

Training size	500,000	5,000,000	10,000,000	25,000,000
Dictionary size	29,801	80,262	111,129	170,053

Table 4 1 Dictionary word count for increasing training text size

Keyboard layout	Prediction accuracy (Error rate)			
Alphabetic	96 48 (3 52)	96 36 (3 64)	96 41 (3 59)	96 33 (3 67)
QWERTY	94 69 (5 31)	94 71 (5 29)	94 69 (5 31)	94 65 (5 35)
Optimised	98 37 (1 63)	98 26 (1 74)	98 25 (1 76)	98 24 (1 76)
Dictionary size	29,801	80,262	111,129	170,053

Table 4 2 Uni-gram prediction accuracy and error rate of known words as dictionary size increases

uni-gram language model Tests by Silfverberg *et al* (2000) revealed that the T9 system predicted the intended word with its first guess with 95 percent accuracy However, this level of accuracy is achieved only on words within the dictionary, and no mention is made of the overall accuracy of the system if unknown words are taken into account The small size of the dictionary is mainly due to memory restrictions on mobile phones Our tests explored the effect of using increased language model training size and order on the prediction accuracy Our language models are trained on training corpora, ranging from 500,000 to 25 million words The size of the resulting dictionaries can be seen in Table 4 1

Ironically, as prediction accuracy improves, nearing 100 percent, users assume that the system will predict correctly and neglect to check that the system has in fact predicted the correct word Consequently, the perceived accuracy of the T9 commonly results in rather cryptic SMS messages A quickly typed "I'll be home in 30 minutes", might result in a somewhat confusing "I'd be good in 30 minutes" Thus, as systems become more accurate, the *error rate*, rather then the accuracy gives us an indication of performance This error rate corresponds to the percentage of known words within the dictionary a user will have to correct as they type

Training size (1000's)	500	2500	5000	10000	25000
Percentage of words known	95 72	98 13	98 73	99 06	99 38

Table 4 3 Percentage of words known in test text as training text size increases

Increased language model training size

Table 4 2 shows the prediction accuracy, for known words, of each keyboard as the dictionary increases with uni-gram language modelling From this table it is evident that, using uni-gram prediction, the accuracy of initial prediction only decrease marginally (roughly 0 1 percent) as the dictionary increases in size

The effect of keyboard layout on error rate is clear The alphabetic keyboard has an error rate twice that of the optimised keyboard layout, while the QWERTY error rate is 3 times that of the optimised keyboard

With only a slight decrease in within-dictionary prediction accuracy as our language model training size increases, the benefits of increasing our training size and corresponding dictionary become clear if we examine the overall prediction accuracy of an unseen text, including those words that are not in the dictionary As the dictionary size increases from 29,801 to 170,053 words, the percentage of words known increases from almost 96% to over 99% (Table 4 3) This is the equivalent of typing an unknown word roughly once every 25 words, versus once every 160 words

Increased language model order

The benefit of using higher-order language models is clear if we graph the initial prediction accuracy of keyboards as the language model order and dictionary size increase Figure 4 4 shows the accuracy of within-dictionary prediction for alphabetic, QWERTY and optimised keyboard layouts As we can see, the accuracy of initial predictions increases steadily as the language model training size increases The effect is greatest for the QWERTY layout, which had the

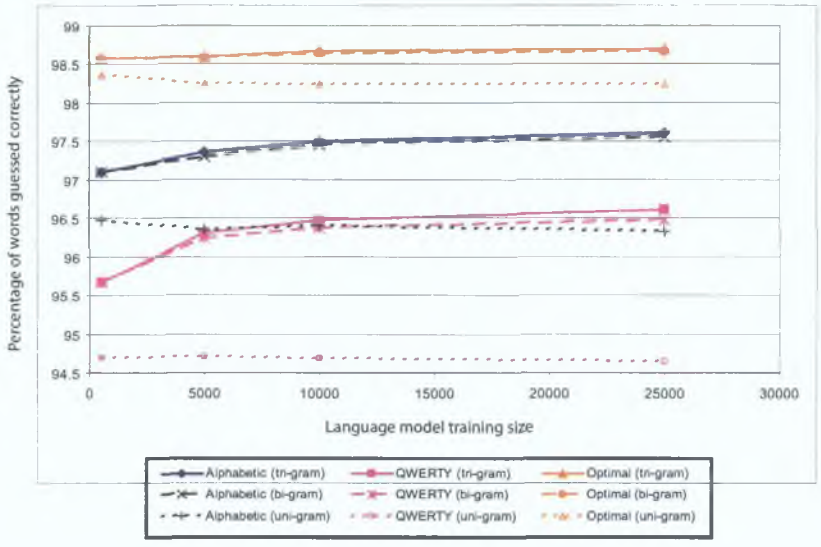


Figure 4.4: Prediction accuracy of known words as language model size increases

highest error rate, and less impressive for the optimised keyboard which already performed with high accuracy. The improvements are most significant as we move from context-insensitive uni-gram language modelling, to context-sensitive bi-gram language modelling, where there is a 30 percent reduction in prediction errors. Less dramatic gains are made as the language model order increases from bi-gram to tri-gram, where only a further 2 percent reduction is made in errors. These improvements are consistent with findings for word completion by Lesher *et al.* (1999), whose spot tests on higher quad-gram language models revealed even smaller gains relative to tri-grams.

Prediction-list length

As mentioned in Section 2.2.7, immersive VR environments do not suffer from the same screen size restrictions as mobile phones or PDAs. Therefore our prediction system can potentially present more than one prediction simultaneously. Figure 4.4 shows that despite improvements offered by the language model, the optimised keyboard still outperformed QWERTY, with an error rate of almost a 3rd that of QWERTY. However, if we increase the number of words a system may suggest,

Keyboard	Prediction list length				
	1	2	3	4	5
QWERTY	3.38	0.84	0.34	0.19	0.13
Alphabetic	2.39	0.48	0.18	0.09	0.06
Optimised	1.30	0.31	0.12	0.06	0.04

Table 4.4: Prediction error rate of known words with increasing prediction list length.

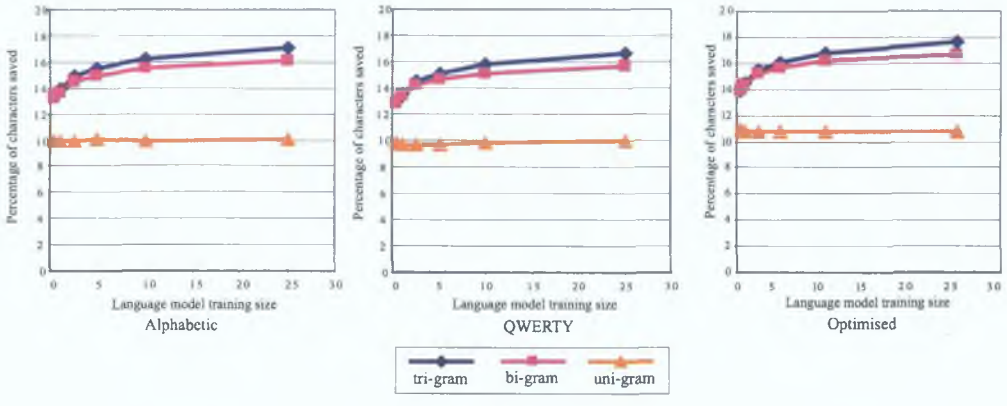


Figure 4.5: The effect of increasing language model order and training size on word-completion.

then the benefits of optimised keyboards become less pronounced.

Table 4.4 shows the prediction error-rate as the prediction-list size increases. At a list length of 5, the prediction accuracy approaches 100 percent and the differences in keyboard layout are likely to have little effect. This is significant when designing our interface as it shows that while optimised keyboards will significantly out-perform a QWERTY keyboard layout based on first prediction alone, an interaction technique which allows for the easy selection of up to 5 potential words will produce only negligible improvements for optimised keyboards.

4.2.3 Word completion accuracy

Tests for word completion were carried out concurrently with those for prediction accuracy. As with the word prediction tests, word completion tests were carried out with increasing language model training size and order.

Increased language order and model training size

The effects of language model order and training size on word completion are similar to those of word prediction Using uni-gram prediction, accuracy remains relatively unaffected by increased dictionary size Savings of roughly 10 percent are achieved using uni-gram prediction with a word completion list length of one, regardless of language model training size, with the optimised keyboard performing just slightly better then QWERTY (Figure 4 5) However, as with word prediction, increasing the language model order has a significant impact on word completion accuracy

Figure 4 5 shows the benefit of increasing N-gram order for QWERTY, alphabetic and the optimised keyboard respectively The benefits of increasing the language model order are clearly visible, improvements in accuracy of over sixty percent are achieved with a QWERTY keyboard when we move from uni-gram to tri-gram prediction As with word prediction accuracy, the improvements are most significant as we move from uni-gram to bi-gram language modelling with less dramatic gains made as the language model order increases from bi-gram to tri-gram However, the benefits of using tri-gram order models over bi-gram increase as language model training size increase, which again is consistent with Lesher *et al* (1999)

Word completion list length

As mentioned in Chapter 2, two techniques for selection of complete words exist the first option is direct selection, whereby any word in a list of potential words can be selected directly, the second option is where the user must iterate though a list of selected words The benefits of direct selection is that a user may choose any word in the list at equal cost However, such a technique may not be feasible in immersive environments where possible selection techniques are determined by the input devices available Consequently, direct and iterative techniques were

evaluated, allowing the advantages of using either technique to be weighed against their feasibility Word completion lists of length 1, 2, 3, and 5 were examined For iterative lists, the cost of one iteration through the list was considered to be the same as one keystroke, this may be generous depending on the interaction technique employed

Direct selection Table 4 5 shows the effect of increasing the language model training size on the percentage of characters saved with word lists of length 1, 2, 3 and 5 respectively Using a tri-gram language model trained on 25 million words, over 29 percent of our test text can be saved using word completion with a list length of 5 This is less than the 55 percent reported by Garay-Vitoria and Gonzalez-Abascal (1997) However, this is to be expected, as the ambiguity of the keyboard makes prediction more difficult Nevertheless, a 29 percent saving of text typed represents a significant potential saving Of particular interest is that over half of the savings offered by word completion can be achieved with a word list length of just one

Iterative selection Table 4 6 shows the percentage of characters saved with increasing word list length with iterative selection Again, values are for a tri-gram language model with a training size of 25 million The results indicate that, if a cost of just one keystroke is assigned to the iteration through the word list, list lengths greater then 2 will actually result in a decrease in performance This would indicate that, if a decision is based solely on word completion accuracy, word lists of 2 should be chosen if iterative selection is to be used

4 3 Conclusions and recommendations

In this chapter we evaluated the effect of keyboard layout, dictionary size and language model order on clash count, and word prediction and completion accuracy The purpose of these experiments was to guide designers in the selection

Keyboard	Completion list length			
	1	2	3	5
QWERTY	16 74	22 21	24 88	27 92
Alphabetic	17 09	22 68	25 45	28 44
Optimised	17 71	23 18	26 18	29 14

Table 4 5 Percentage of characters saved with increased list length with direct selection

Keyboard	Completion list length			
	1	2	3	5
QWERTY	16 74	17 57	15 78	10 37
Alphabetic	17 09	18 81	17 2	12 34
Optimised	17 71	18	16 31	11 13

Table 4 6 Percentage of characters saved with increased list length with iterative selection

of potential keyboard layouts, language models, interaction techniques, and word list lengths for both word prediction and completion

In contrasting keyboard layouts, we found that, when compared to alphabetic, and optimised keyboards, the QWERTY keyboard performance is mixed, and the choice of keyboard will likely depend on the interaction techniques employed With larger dictionary sizes the maximum sequence count for the QWERTY keyboard approaches double that of optimal and alphabetic keyboards with a maximum sequence count of 55 for a dictionary of 170,000 words The accuracy of initial predictions is consistently lower than alphabetic and optimised layouts However, with word prediction lists of 5 words, the difference between keyboards is less pronounced Also, for word completion, the difference between keyboards is less evident Thus, if an interaction technique where users can select up to 5 potential words from a predicted list is used, QWERTY will perform with similar accuracy to alphabetic and optimal keyboard layouts However, if the interaction technique used dictates that users must iterate through each incorrect prediction before selecting the correct one, then alphabetic or optimised keyboard layouts offer attractive alternatives

As with keyboard layouts, the word completion list length employed should be influenced by the interaction technique used If a direct selection technique is used, then list lengths of 5 will provide potential savings of up to 29 percent However, if an iterative selection technique is employed, shorter list lengths are advised, as maximum potential savings are achieved with a list length of 2 Finally, regardless of interaction technique employed, savings of 17 percent, half of all savings, can be achieved with list lengths of just 1

In examining dictionary size we found that, although clash count naturally increases with dictionary size for all keyboards, the within-dictionary prediction accuracy remains unaffected if predicted words are sorted according to frequency The most obvious benefits of increased dictionary size is the percentage of words known Over 99% of words in our 250,000 word test corpus were known when our dictionary exceeded 170,000 words

The benefits of higher order language models become most apparent as the their training size increases The greatest gains are made moving from context insensitive uni-grams to context sensitive bi-grams On average, prediction errors of known words are reduced by over 30 percent when tri-gram language models are used rather then uni-gram For word completion, tri-gram language modelling increases prediction accuracy by 60 percent

Chapter 5

Gesture recognition for virtual typing

5.1 Introduction

Interaction with the virtual keyboard proposed in this thesis is dependent on the accurate recognition of key-press postures Ideally, the recognition system should be user-independent, yet accurate In the following chapter we will discuss various recognition errors, and their relevant impact in the context of text-entry We will examine the effects of dynamic posture recognition as a solution to the segmentation problem We will detail the results of empirical tests to examine the template matching techniques discussed in Chapter 2 Finally, we will suggest the use of a hybrid method for user independent key-press gesture recognition suitable for text-entry using datagloves

5 1 1 Recognition accuracy

One of the key factors to facilitate the use of the virtual keyboards is the accurate and consistent recognition of intended key-presses Users familiar with standard keyboards, which, due to their explicit nature, have 100 percent recognition accuracy, will quickly become frustrated with a virtual keyboard system if it does

not behave in a similar manner to the physical keyboard they are accustomed to The faith of users in the system is undermined if their own occasional errors - where they bend the wrong finger - are compounded by errors made by the gesture recognition system Moreover, frustration levels will likely be affected by the type of errors the system makes As discussed in Chapter 2, errors in recognition can be classified as either false positives, false negatives, or misclassification errors These errors can have various consequences on the usability of any system, and the choice of a gesture recognition technique should take into account the likelihood of each type of error, and the corresponding effects

In the case of ambiguous virtual keyboards, errors can be ranked, in increasing order of severity, as follows

- False negatives
- False positives
- Misclassification

False negatives represent the least problematic of errors This is because, by providing auditory feedback, users can be notified when a key-press is recognised, and any key-press which is not recognised is quickly identified the lack of any feedback

False positives are equally identifiable, as the user will hear a sound upon recognition, and may then delete the unintended gesture However, false positives can be more destructive then false negatives if the system provides a more complicated gesture set then simple key-presses As well as having 10 key-press gestures, represented by the flexion of each of the 10 fingers, the system could also recognise a fist, or a pointing gesture, which could be assigned to functions such as *exit*, *select*, *change mode*, etc The false positive recognition of any of these special gestures would have a larger impact on the system, and might require more user intervention then the simple deletion of an extra key-press

Misclassification errors represent the largest problem for ambiguous virtual keyboards, as they are difficult to identify by the user, and generally will not be noticed until a word, which the user feels should be in the dictionary, is not found This is due to the ambiguous nature of the keyboard, and the continuous changing of the active word, as the system tries to match the current sequence of key-presses to a viable word Users familiar with the system will ignore the active word until they have finished typing it, and it is only at this stage that they will recognise that an error has occurred This results in the user deleting most or all of the word, and attempting to type the word again This is both frustrating and time consuming, and leads to high frustration with the system Like false positives, misclassification errors can also result in the recognition of unintended special gestures, which can be more complicated to remedy

An ideal system is one which accurately reflects the user's intention sensitive enough to eliminate false positives, but not record false negatives, but also forgiving enough to allow for variation between users and thus not misclassify gestures In ranking a gesture recognition algorithm, errors must be evaluated accordingly, thus misclassification is the most important factor, followed by false positives, and finally false negatives

Additional requirements for an ideal gesture recognition system, though not as important, would be ease of programming and adaptability of the system, and negligible training time

5 1 2 Dynamic posture recognition

As previously mentioned, posture recognition only takes finger posture at a particular moment into account, whereas dynamic posture recognition, as we have defined it, measures finger posture over time By measuring finger flexion over time, we can calculate the speed at which fingers are travelling at any given moment This extra information is useful in determining the current state of a posture Specifically, this information helps avoid early false positive recognition

which can arise due to the segmentation problem, discussed in Section 2 4 7, by identifying transition states

The creation of a typical posture can be broken into 3 sections

1 Forming gesture

2 Gesture complete

3 Returning to neutral

Posture recognition should only take place in the second section as, depending on the gesture set, certain gestures may require the hand to *pass through* others during formation and after completion

To counter this, a system could retrospectively cancel a previous posture if a second posture was recognised within a short time However, this is not ideal as it would involve creating a hierarchy of postures that determined which postures should be deleted if another posture was subsequently recognised and which should not This would then have to be re-evaluated every time a new posture was added to the posture set Also, the seemingly erratic responses of the system as it cancels postures might prove disconcerting to the user

A simpler approach, is to simply not recognise postures until they have been completed By analysing finger speed, we can identify sections 1 and 3 of the creation of a posture, as the speed at with the fingers are moving is considerably faster Thus, using dynamic posture recognition, postures should only be recognised if all of the fingers are travelling below a threshold speed

This is clearly visible if we graph the flexion of the fingers, and the corresponding finger speed, during the creation of a typical posture (Figures 5 1 and 5 2) Visual scanning of a large set of similarly graphed postures by eye revealed that a speed above 4% per millisecond (ms) could be considered a transition period To confirm this, we plotted the accuracy of recognition, while varying the threshold speed, above which, gestures could not be recognised (Figure 5 3) This confirmed

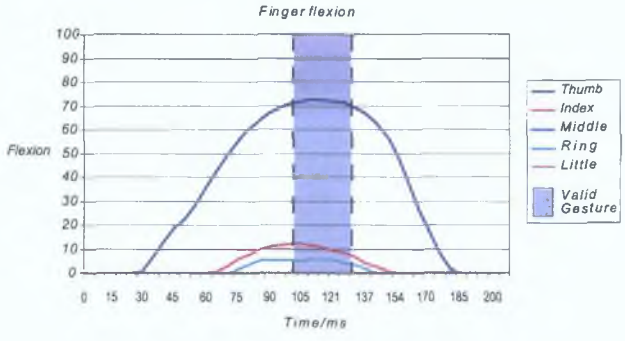


Figure 5.1: *Sample gesture flexion* An example of gesture, with slight sympathetic bending visible.

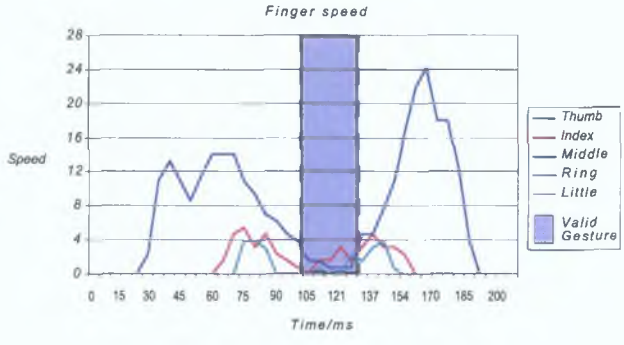


Figure 5.2: *Sample gesture speed* By analysing finger speed, sections 1, 2 and 3 of a posture are clearly identifiable because the speed falls below a threshold.

our initial findings, as the lowest error rates were recorded between 3.5%/ms and 4%/ms.

5.2 Formal evaluation of previous template systems

5.2.1 Outline

In the search for an effective, user independent posture recognition technique suitable for text-entry with dataglove, we undertook to evaluate techniques for simple posture recognition suggested by Sturman (1992), Evans *et al.* (1999) and Kramer *et al.* (1991), with particular focus on the types of recognition errors made, and the effects of sympathetic bending. Also of interest was the effect of dynamic posture recognition on reducing errors caused due to the segmentation

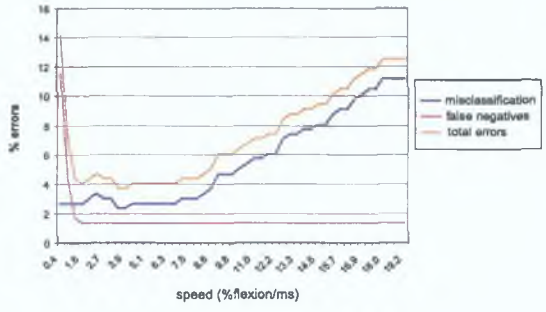


Figure 5.3: *Recognition error rates as threshold speed increases* Error rates drop to 3.7% at a finger speed threshold of 4%/ms, and increase as the threshold value is increased.

problem.

5.2.2 Test procedure

A permanent database of gestures was created to test each algorithm. Five users, wearing 5DT gloves, were recorded performing each gesture. The gloves were calibrated for each user before each recording. The total range of motion of each finger was measured, and the flexion recorded in the database was scaled according to this range. To create the gesture database, each user was shown a pair of virtual hands. Individual fingers on each virtual hand were highlighted to indicate the desired posture. Users were requested to create the posture indicated by the highlighted fingers. (Figure 5.4).

Six postures were tested: one with each of the fingers flexed individually, which are necessary for virtual typing, and a fist gesture, which is a reserved gesture in our system used to signal *delete*. Data from the gloves was then recorded and tagged as users made each gesture. Users indicated the end of the gesture by pressing a physical key with the non-active hand. Users were shown the six postures for the active hand in random order, at which point the active and non-active hand were switched. This process was repeated several times for each user. The resulting data was then inspected by eye, and incorrect gestures - where users created the wrong gesture - were removed. In total over 500 gesture

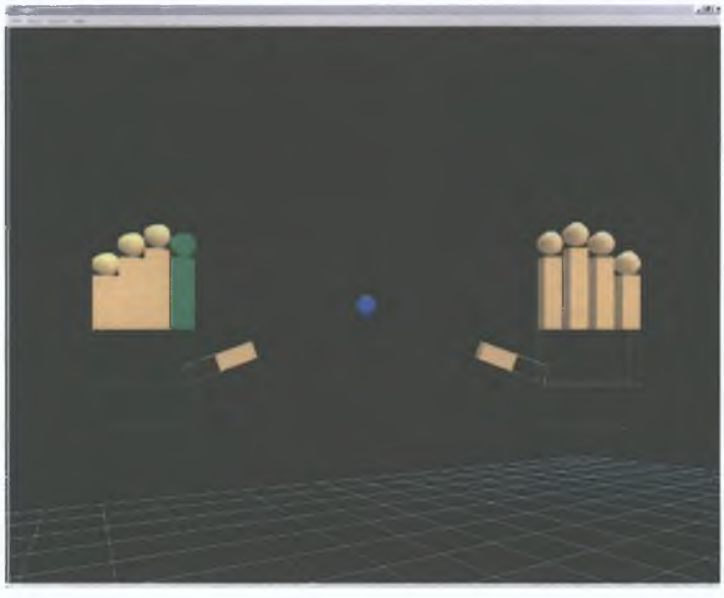


Figure 5.4: *Hightlighted gesture* Users created the indicated gesture, then pressed a physical key and were shown a new gesture.

examples remained after errors had been removed. The data was then fed to each of the gesture recognition techniques, and the accuracy of each method noted. The errors were split into two categories: false negatives, and misclassification. Due to the nature of the test, it was impossible for false positive errors to occur, as the database only contained intended gestures.

5.2.3 Results

Sturman's technique Sturman (1992) suggests thresholds of 80% and 20% flexion for classification of flexion and extension respectively. Using these values we attempted to recognise gestures performed by our test subjects.

Accuracy of the system was tested using static and dynamic posture recognition. As is clearly visible from Tables 5.1 and 5.2, accuracy is quite low for both techniques. Accuracy is higher for static gesture than for dynamic measuring due to the fact that by measuring gestures as they are in the process of being formed, static recognition identifies some gestures before sympathetic bending has occurred.

```
/* Columns  Time    [L]ittle   [R]ing   [M]iddle   [I]ndex   [T]humb
Gesture  Middle finger
Time        L    R    M    I    T
39568 6     0    0    0    0    2
39568 6     0    0    0    0    4
39568 6     0    0    0    0    6
39568 6     0    0    0    0   10
39568 7     0    0    0    0   14
39568 7     0    0   15    0   20
39568 7     0    0   47    0   24
39568 7     0    0  100    0   30
39568 7     0    1  162    0   36
39568 8     0   17  204    0   42
39568 8     0   32  233    0   42
39568 8     0   39  246    0   42
39568 8     0   42  249    0   40
39568 8     0   32  239    0   34
39568 9     0   10  200    0   26
39568 9     0    0  128    0   12
39568 9     0    0   51    0    0
Gesture  Little finger
Time        L    R    M    I    T
39580       0    0    0    0   52
39580       0    0    0    0   54
39580      14    0    0    0   50
39580      64    0    0    0   46
39580     130    0    0    0   46
39580 1   183    0    0    0   44
39580 1   215    0    0    0   42
39580 1   233    0    0    0   44
39580 1   237    0    0    0   42
39580 1   220    0    0    0   40
39580 2   178    0    0    0   38
39580 2    93    0    0    0   20
39580 2     0    0    0    0   24
```

Example 5 1 Tagged data recorded from user, flexion scaled from 0-255 based on user calibration

Analysis of the errors reveals that over 99% of errors are false negatives, indicating that the 80% and 20% thresholds suggested by Sturman are too restrictive, and cannot be comfortably made for simple finger presses This hypothesis was tested by relaxing the thresholds suggest by Sturman to 60% and 50% for flexion and extension respectively, gestures were recognised if the intended fingers are flexed more then 60% and the remaining fingers remain under 50%

Results with this new ratio (Table 5 3) showed a large reduction in errors, with an overall accuracy of 92% The use of dynamic gesture recognition only impro-

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	68 54	0 00	0 00	0 00	0 00	0 00	31 46
ring	0 00	79 55	0 00	0 00	0 00	0 00	20 45
middle	0 00	0 00	72 29	0 00	0 00	0 00	27 71
index	0 00	0 00	0 00	77 38	0 00	1 19	21 43
thumb	0 00	0 00	0 00	0 00	86 90	0 00	13 10
delete	0 00	0 00	0 00	0 00	0 00	63 64	36 36

Table 5 1 Confusion matrix of gestures recognised using Sturman's technique with flexion and extension thresholds of 80% and 20% respectively Each row shows the intended posture, with the corresponding posture recognised

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	66 29	0 00	0 00	0 00	0 00	0 00	33 71
ring	0 00	78 41	0 00	0 00	0 00	0 00	21 59
middle	0 00	0 00	61 45	0 00	0 00	0 00	38 55
index	0 00	0 00	0 00	73 81	0 00	1 19	25 00
thumb	0 00	0 00	0 00	0 00	86 90	0 00	13 10
delete	0 00	0 00	0 00	0 00	0 00	60 23	39 77

Table 5 2 Confusion matrix of Sturman's technique using dynamic posture recognition with flexion and extension thresholds of 80% and 20% respectively

ved accuracy slightly, but reduced misclassification errors caused by premature recognition, in favour of false negative errors due to sympathetic bending

Sturman's method will always under-perform with users who have strong sympathetic bending This is due to the fact that sympathetic bending can cause two fingers to cross the threshold value, leading to a situation of ambiguity which Sturman's technique cannot resolve, with a resulting high error count

Maximum flexion method The maximum flexion technique, as described by Evans *et al* (1999), can recognise 5 postures, 1 for each finger and the thumb In order to recognise extra gestures, such as a fist, exceptions were made to the standard technique A fist is recognised whenever all 4 fingers are above the minimum threshold value and is given higher priority then all other gestures Although adequate for a fist posture, this priority technique would quickly be-

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	85 39	2 25	0 00	0 00	0 00	0 00	12 36
ring	0 00	96 59	0 00	0 00	0 00	0 00	3 41
middle	0 00	0 00	97 59	0 00	0 00	0 00	2 41
index	0 00	1 19	0 00	95 24	0 00	0 00	3 57
thumb	0 00	0 00	0 00	0 00	97 62	0 00	2 38
delete	1 14	15 91	1 14	2 27	0 00	78 41	1 14

Table 5 3 Confusion matrix of gestures recognised using Sturman's technique with flexion and extension thresholds of 60% and 50% respectively

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	78 65	1 12	0 00	0 00	0 00	0 00	20 22
ring	0 00	96 59	0 00	0 00	0 00	0 00	3 41
middle	0 00	0 00	95 18	0 00	0 00	0 00	4 82
index	0 00	0 00	0 00	95 24	0 00	1 19	3 57
thumb	0 00	0 00	0 00	0 00	97 62	0 00	2 38
delete	0 00	3 41	0 00	0 00	0 00	93 18	3 41

Table 5 4 Confusion matrix of Sturman's technique using dynamic posture recognition with flexion and extension thresholds of 60% and 50% respectively

come unworkable if more gestures were added In fact, many of the 32 feasible simple postures are only possible if a maximum threshold is introduced, similar to Sturman's technique

This adapted maximum flexion technique was tested using both static and dynamic posture recognition As can be seen from Table 5 5, overall accuracy using static recognition is quite low This is considerably lower then the 99% accuracy reported by Evans *et al* This is mainly due to addition of the fist gesture to the system, which was often misclassified while it was being formed In fact, 94% of the gestures misclassified were a result of fist gestures

Furthermore, Evans *et al* indicated that aural feedback was given in their experiments during the creation of the gestures, thus false negatives were unlikely to occur Factoring these favourable conditions - where only 5 gestures were tested, no fist gesture was tested, and no false negatives were possible - accuracy

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	93 26	4 49	0 00	0 00	0 00	0 00	2 25
ring	0 00	96 59	0 00	0 00	0 00	0 00	3 41
middle	0 00	0 00	98 80	0 00	0 00	0 00	1 20
index	0 00	1 19	0 00	95 24	0 00	0 00	3 57
thumb	0 00	0 00	0 00	0 00	97 62	0 00	2 38
delete	6 82	40 91	25 00	18 18	0 00	7 95	1 14

Table 5 5 Confusion matrix of static postures recognised using the maximum flexion technique

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	94 38	3 37	0 00	0 00	0 00	0 00	2 25
ring	0 00	96 59	0 00	0 00	0 00	0 00	3 41
middle	0 00	0 00	98 80	0 00	0 00	0 00	1 20
index	0 00	0 00	0 00	95 24	0 00	1 19	3 57
thumb	0 00	0 00	0 00	0 00	97 62	0 00	2 38
delete	0 00	7 95	0 00	0 00	0 00	90 91	1 14

Table 5 6 Confusion matrix of dynamic postures recognised using the maximum flexion technique

of 98 9% would have been achieved

Considerable gains were achieved in recognition accuracy using dynamic posture recognition As can be seen in Table 5 6, accuracy increased to 96% when finger speed was considered The increase in accuracy is largely due to the improved accuracy recognising the fist posture, which is less likely to be misclassified during creation, as fingers are moving at high speed

Euclidean distance Two methods were originally used to define posture templates, to which Euclidean distance would be measured Firstly, postures were defined as the average finger flexion values of recorded users This data was collected in a similar manner to the gesture database, but only the final posture was recorded Data from the gloves was only recorded and tagged when users indicated they had formed a gesture by pressing a physical key with their non active hand This was contrasted with a simpler technique, where templates were

defined as ideal postures Thus, for example, a little finger key-press would be defined as 100% flexion of the little finger, and 0% flexion for the remaining fingers Initial tests showed no difference in performance between our posture templates, thus, the second method of the two, which required no training data, was used for the remainder of tests

Two Euclidean distance classifiers were explored as potential techniques for posture recognition, as suggested by Kramer *et al* (1991) one with a distance threshold, below which postures could not be recognised, and another which always classified the closest posture For the first technique, choosing the ideal distance threshold involves a trade off between two errors A large distance threshold will result in a high false positive rate as the system is essentially too sensitive, while a low threshold value will lead to a high false negative rate An alternative to this, is to always choose the closest gesture irrespective of distance To facilitate this, we created a *flat* gesture, which when recognised, corresponds to *no gesture* Both of these techniques were initially tested using static and dynamic posture recognition However, these quickly revealed that dynamic recognition was superior, thus in-depth testing focussed on analysing the difference between both techniques

Using the distance threshold technique, analysis revealed a threshold distance of 61 produces optimum accuracy Using dynamic posture recognition, this threshold value produced gesture accuracy of 95% (Table 5 7)

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	83 15	0 00	0 00	0 00	0 00	0 00	16 85
ring	0 00	100 00	0 00	0 00	0 00	0 00	0 00
middle	0 00	0 00	97 59	0 00	0 00	0 00	2 41
index	0 00	0 00	0 00	97 62	0 00	1 19	1 19
thumb	0 00	0 00	0 00	0 00	97 62	0 00	2 38
delete	0 00	4 55	0 00	0 00	0 00	93 18	2 27

Table 5 7 Confusion matrix of dynamic postures recognised using the Euclidean distance with a threshold of 61

Although accurate, the majority of errors are due to false negatives Closer inspection of the data reveals that these errors are usually cases of strong sympathetic bending as opposed to weakly formed gestures Example 5 2 is a typical example of this, where the sympathetic bending of the ring finger causes a false negative error, when clearly a gesture is intended

```
/* Flexed little finger posture */
[  99 2,   99 1,   24 3,   7 8,    9 4]
```

Example 5 2 Little finger flexed with strong sympathetic bending of the ring finger

This contrasts with the closest gesture technique Although the accuracy of the closest gesture technique is only marginally better at 96% (Table 5 8), closer analysis reveals that the majority of misclassification errors were a result of weakly formed gestures, rather then gestures with large sympathetic bending, which were recognised with high accuracy Example 5 2 was accurately recognised, however Example 5 3 was not

```
/* Flexed index finger posture */
[     0       0,      0,   45 1     3 9]
```

Example 5 3 Weak index finger flexion

Gesture	% little	% ring	% middle	% index	% thumb	% delete	% not recognised
little	92 13	3 37	0 00	0 00	0 00	3 37	1 12
ring	0 00	96 59	0 00	0 00	0 00	0 00	3 41
middle	0 00	0 00	98 80	0 00	0 00	0 00	1 20
index	0 00	0 00	0 00	96 43	0 00	1 19	2 38
thumb	0 00	0 00	0 00	0 00	97 62	0 00	2 38
delete	0 00	5 68	0 00	0 00	0 00	93 18	1 14

Table 5 8 Confusion matrix of dynamic postures recognised using the closest posture technique

Hybrid method Finally, we used a hybrid method which combined aspects from both the maximum flexion and Euclidean distance techniques Although

the technique employed by Evans *et al* is accurate and performs well with strong sympathetic bending, it has a limited set of gestures which can only be increased by using priority based exceptions One alternative is to use the Euclidean distance to choose the intended posture in place of the maximum flexion suggested by Evans *et al* Thus, if sympathetic bending causes two or more postures to be recognised, the Euclidean distance to each of these potential postures is measured, and the closest one chosen as the intended posture This technique allows for a larger proportion of the 32 simple postures possible

This technique was tested in conjunction with the maximum flexion technique The results were identical, although no improvement is achieved using Euclidean distance, no deterioration in performance is noted either Thus this technique has the accuracy of maximum flexion technique, without its limitations

5 2 4 Analysis

Five methods for posture recognition were tested Of these, four performed with a higher degree of accuracy when dynamic posture recognition – which takes finger speed into account – was applied

Technique	Type	Accuracy %	False neg %	Misclassified %
Sturman 80/20	S	75	25	0
Sturman 80/20	D	71	29	0
Sturman 60/50	S	92	4	4
Sturman 60/50	D	93	6	1
Maximum threshold	S	82	2	16
Maximum threshold	D	96	2	2
Euclidean Threshold	D	95	4	1
Euclidean closest	D	96	2	2
Euclidean hybrid	D	96	2	2

Table 5 9 Summary of the accuracy of tested techniques using static and dynamic posture recognition, and the corresponding errors S and D indicate the type of posture recognition used static and dynamic respectively

Sturman's technique, although allowing for a large gesture set, proved too

restrictive for users with sympathetic bending The maximum flexion technique, suggested by Evans *et al* , although suitable for users with sympathetic bending, allowed for only a small subset of the gestures possible with Sturman's technique Finally, we analysed three Euclidean distance techniques firstly a distance threshold technique, which recognised a posture once it came within a predetermined distance of a posture template, secondly, a closest posture technique, which recognised the closest posture at any stage, lastly a hybrid, flexion threshold technique, which used Euclidean distance to determine the closest posture in cases where more then one finger passed a flexion threshold

Each of the Euclidean distance techniques performed with a high accuracy, with recognition rates of 95 percent and above With little to choose between them in terms of accuracy, error types play a large part in deciding which is most suitable for the posture recognition needed to interact with the proposed virtual keyboards As mentioned previously, error types can be ranked in order of severity False negatives, where an intended gesture made by the user is not identified by the system, prove to be the least problematic when interacting with a virtual keyboard However, false negatives caused due to sympathetic bending prove frustrating to users, who, thinking the system hasn't recognised their gesture because it isn't strong enough, often unintentionally exaggerate the sympathetic bending by attempting to create a stronger gesture

Closer inspection reveals that, of the three Euclidean techniques, each have aspects which dictate the possible errors Each divide the feature space differently and thus have different characteristics

A system using a distance threshold is likely to suffer from false negative errors due to sympathetic bending A compromise must be reached when choosing a threshold distance If the distance is too great, the system will be too sensitive, which leads to false positive classifications However, by reducing the threshold distance, we are likely to leave areas of the feature space unaccounted for, which leads to increased false negatives Thus strong postures, which have

strong sympathetic bending, are less likely to be classified, when clearly a posture is intended

A system measuring the closest posture will never suffer from this problem This is because the *entire* feature space is divided between postures, and a strong posture will always be recognised as *something* However, although reducing false negative errors, this will naturally lead to the increased likelihood of misclassification errors This is due to the cumulative nature of Euclidean distance, whereby the light flexion of other fingers combine to increase the overall distance between a created posture and the intended posture template As we move further away from a template posture, the probability of misclassification naturally increases

Finally the hybrid Euclidean distance technique offers perhaps the best option Like the closest Euclidean technique, it will have no false negative errors due to strong sympathetic bending Its advantage, however, lies in the reduced number of postures which the system must choose between If two fingers are flexed passed the threshold value, then the system must only choose between these postures, rather then the entire gesture set By reducing the gesture set examined, we are reducing the potential for misclassification errors

Example 5 4 was recognised correctly by the hybrid technique, but misclassified as a fist by the closest gesture technique, and simply not recognised as a gesture (false-negative) by the threshold technique This was due to the fact that the hybrid technique only examined the distance to the little, ring, and middle posture templates, as the weak index finger flexion excluded the fist as a possible gesture (in this case the little finger was chosen because, although equidistant to the ring finger, it was simply examined first and no other posture was closer)

```
/* Flexed little finger posture */
[    100    100,   87 5,  24 3,   7 5]
```

Example 5 4 Little finger with very strong sympathetic bending

5.3 Conclusions and recommendations

In choosing a posture recognition technique, the type of postures to be created play a large role in the complexity of the recognition algorithm employed Postures for virtual typing, which require only fully flexed or extended fingers, are classified as simple postures Due to its reduced complexity, simple posture recognition should ideally be user independent, yet accurate This is hindered by the large variation in dexterity between users, who will suffer sympathetic bending to lesser or greater extent Segmentation of postures may prove problematic if the posture set contains postures which can be mistakenly recognised while another is being formed However, measuring finger speed a well as flexion effectively counters this problem

For virtual typing, the posture set is potentially quite small, with a minimum of only 5 postures needed The maximum flexion technique suggested by Evans *et al* is sufficient for a posture set of this size However, the addition of extra postures will naturally enhance the interaction possible *Exit*, *Delete*, or *Carriage return* gestures are just some of the possible uses for extra postures The maximum flexion technique is less suited to an increased posture set Measuring the Euclidean distance to possible postures is an effective method for classifying these larger simple posture sets Three variations have been described One with a distance threshold, one which simply chooses the closest at all times, and a hybrid technique with a flexion threshold In choosing a technique, the types of errors likely to occur should be considered Misclassification errors are the least desirable errors in a predictive typing environment as they are the hardest to identify during typing Of all the techniques tested the hybrid Euclidean technique is recommended This is due to its characteristics, which mean it performs well with users with strong sympathetic bending, while being less likely to commit misclassification errors

Finally, it should be noted, that although we have discussed the benefits of

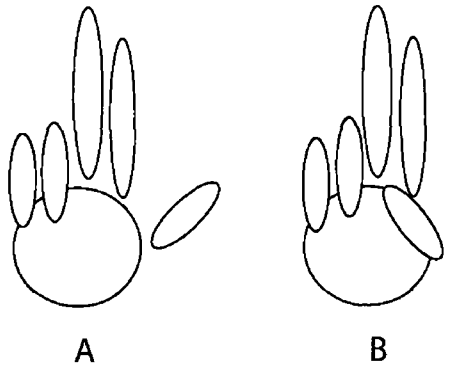


Figure 5 5 Two fingered point Without bending the thumb (A), the gesture is indistinguishable from a flexed little finger with strong sympathetic bending of the ring finger With the thumb flexed (B), the gesture is easier to identify

adding to the posture set in order to increase interaction possibilities and recommended techniques to recognise these increased posture sets, care should be taken that extra postures are chosen with consideration for sympathetic bending The most effective method for accurate posture recognition is simply ensure that gestures not unnecessarily similar A simple two fingered point posture, for example (Figure 5 5a), would be indistinguishable from a flexed little finger with strong sympathetic bending of the ring finger However, it would be easier to distinguish if it was made in combination with a flexed thumb (Figure 5 5b), which remains relatively unaffected to sympathetic bending when the little and ring fingers are bent Thus, although the 32 potential postures theoretically possible might not be available, an adequate posture set is certainly possible

Chapter 6

Interaction techniques for predictive text-entry

6.1 Introduction

This chapter will examine interaction techniques suitable for use with ambiguous virtual keyboards Using the task decomposition, we will create a taxonomy of predictive text-entry We will then examine some of the selection techniques possible with the 5DT dataglove and combine these with our taxonomy to suggest potential interaction techniques for our ambiguous text-entry system Finally, we will conduct experiments to evaluate the selection techniques suggested

6.2 Interaction with ambiguous keyboards A brief discussion

Before discussing the design of interaction techniques, it is perhaps useful to review the theory of dictionary-based ambiguous text-entry, and discuss the importance of effective interaction techniques

Mapping more then one letter to a key results in a reduced keyboard The advantage of such keyboards is the increased ease of key selection Users must

select between eight unique keys rather than twenty-six[1] This eases the problem of key selection, but causes an inherent ambiguity while typing, as the system must attempt to determine the intended key Taking advantage of the entropy of the English language, dictionary lookup provides an effective solution to this problem However, 100% perfect prediction is impossible, thus, the ease afforded by the reduced keyboard is offset by the need to resolve incorrect predictions by the system Central to the effective use of ambiguous keyboards for text-entry is the disambiguation process, where the user confirms that the predicted word is correct, or chooses an alternative The benefits of ambiguous keyboards are lost if the interaction techniques used to disambiguate words are not efficient

A further benefit of using a dictionary to aid prediction, is the potential to increase user throughput by predicting complete words before they are finished Here, as with word prediction, effective interaction techniques are necessary The time taken to highlight and select complete words must be less then the time take to simply finish the word by typing normally

6 3 A taxonomy of predictive text-entry

In Section 2 5 4, we discussed the benefits of task decomposition, by decomposing a task into sub-tasks, we can identify the core components required to perform a task This allows us to create interaction techniques for our overall task as a whole, by mapping interaction methods to each sub-task

Interaction with our ambiguous keyboard can be decomposed into two fundamental sub-tasks Firstly, the selection of virtual ambiguous keys This is the primary selection task in our system, and will be the most commonly performed interaction with the system Secondly, the selection of ambiguous or complete words

The taxonomy of selection has already been discussed (Figure 2 14) The

[1] In practice, ambiguous keyboards with as few as three keys have been used However, our own system will usually have eight

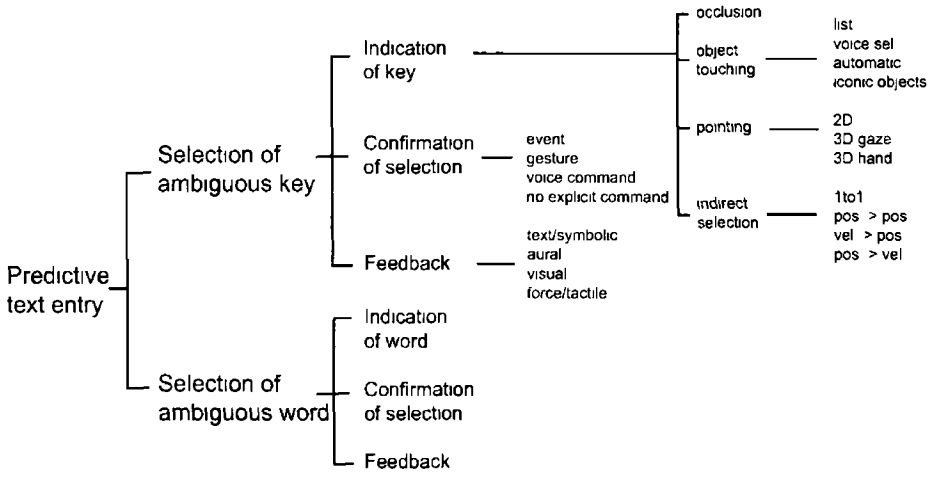


Figure 6 1 Taxonomy of predictive text-entry

task of predictive text-entry can be seen in as an extension of this, where both subtasks map directly to the fundamental task of selection (Figure 6 1)

Having decomposed our task, the benefits of creating a taxonomy become clear By deconstructing the task into its components, we can create various interaction techniques by combining different sets of core-components The problem of creating an interaction technique suitable for predictive text-entry then becomes a problem of combining possible interaction techniques for the core-components Naturally, not every combination will be practical or even possible depending on the input technology available Consequently, creating an interaction technique involves choosing between a sub-set of plausible techniques given our input capabilities

6 3 1 Visual aspects of word selection a discussion

Several options are available for the visual presentation of predicted and complete words The method in which words are visually presented will dictate the interaction techniques which are used to select them These were explored using our prototype application, evolving over time

Word prediction Our initial prototype system adhered closely to the prediction style used in the T9 system for mobile phones As a user typed, the most likely word was suggested If the word suggested was not the one intended by the user, alternatives were selected by performing a *next* gesture Upon recognition of the *next* gesture, the system displays the next most likely word This process is repeated until the intended word was shown, at which point the user accepts the word with a *select* gesture

This technique was then augmented with colour to provide more information to the user Words coloured green indicated that the prediction was one of several possibilities, signifying that users could iterate through alternative words if the predicted word was incorrect Words were coloured amber if only one word in the dictionary matched the current key sequence, indicating that no iteration was possible Finally, words were coloured red to indicate that the last key-press had resulted in a key sequence that didn't match any word in the dictionary This colour scheme augmented the original system, providing more information to the user However, by showing only the most likely prediction, the system forced the user to iterate through predictions, unsure of what the next word offered might be, or even if the desired word was in the dictionary

An alternative solution is to offer an ordered list of words, ranked according to their likelihood This technique shows users all (or at least more) of the possible matching words simultaneously The benefit of this is that, should a suitable interaction technique exist, the user can directly select the desired word, without needing to iterate though alternatives first In Chapter 4 we reviewed the accuracy of prediction with increasing list length, and language size and order Our experiments showed that, for a trigram language model with a training size of 25 million words, prediction accuracy of 99 9 percent could be achieved with list lengths of five Thus, in discussing interaction techniques to be used, it will be assumed that lists of length five will be used by default, with the provision for extra words made according to the selection technique used

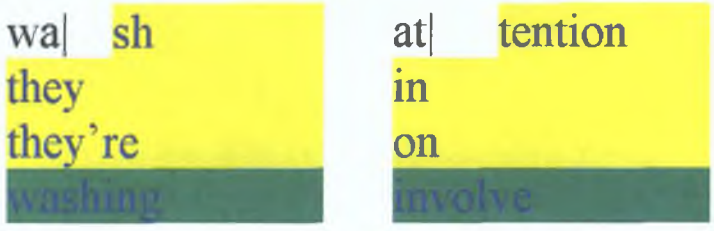
(a) (b) (c)

Figure 6.2: (a) Microsoft Word Example (b) Simple greyed out technique (c) List of words which map to the current sequence interpretation

Word completion As with word prediction, various presentational possibilities exist for complete words. The simplest of these is the *tab completion* style. This is offered by most Unix consoles[2] and with a slightly augmented version, where the whole word is offered, in Microsoft Word (Figure 6.2a). Here, the most likely ending to the sequence of keys entered thus far is shown, greyed out. The user may simply choose to ignore the ending, or complete the word using a *tab* gesture (Figure 6.2b). This can further be augmented by showing a list of potential complete word endings which map to the current word sequence interpretation (Figure 6.2c). However, this requires a second gesture to iterate through each ending, or a separate selection gesture for each ending.

A problem with these tab completion styles is that, with ambiguous keyboards, the most likely complete word may not match the current interpretation of what has been typed so far. For example, while typing *winning* on an alphabetic keyboard, the word completion system might guess the user's intention after three letters. However, the first three ambiguous keys spell the words *who* and *win*. Statistically, *who* is more likely then *win*. Here, the greyed out ending, *ning* after *who*, would be confusing. Therefore, tab complete systems for ambiguous keyboards must either offer the entire word separately, or only offer endings which map the current interpretation of the typed sequence. An alternative is to offer complete words irrespective of the current estimation of the word sequence

[2]The Unix console do not show potential endings, but leave it to the user to decide if the current directory contains any other words with the same beginning.



(a) (b)

Figure 6.3: (a) Complete words displayed irrespective of sequence interpretation (b) Predicted and complete words shown together

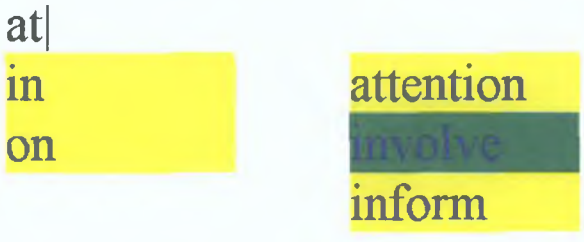


Figure 6.4: Predicted and complete words displayed on separate lists

(Figure 6.3a).

Finally, any proposed word completion technique cannot be considered in isolation. Rather, is must be considered in context, and thus, also facilitate the selection of predicted words. If list selection is to be used for word prediction as well as word completion, then two options exits: the lists can be combined (Figure 6.3b), or they can be separated (Figure 6.4). The advantage of a joint list is that only one interaction technique is needed to select either predicted or complete words. The disadvantage is the possible confusion such a list might cause.

We have opted for a separate list of words in our system for several reasons. Firstly, using a separate list allows for a set number of words to be offered in our word completion list irrespective of the number of ambiguous words which map to the current word sequence. Secondly, the graphical nature of VR affords

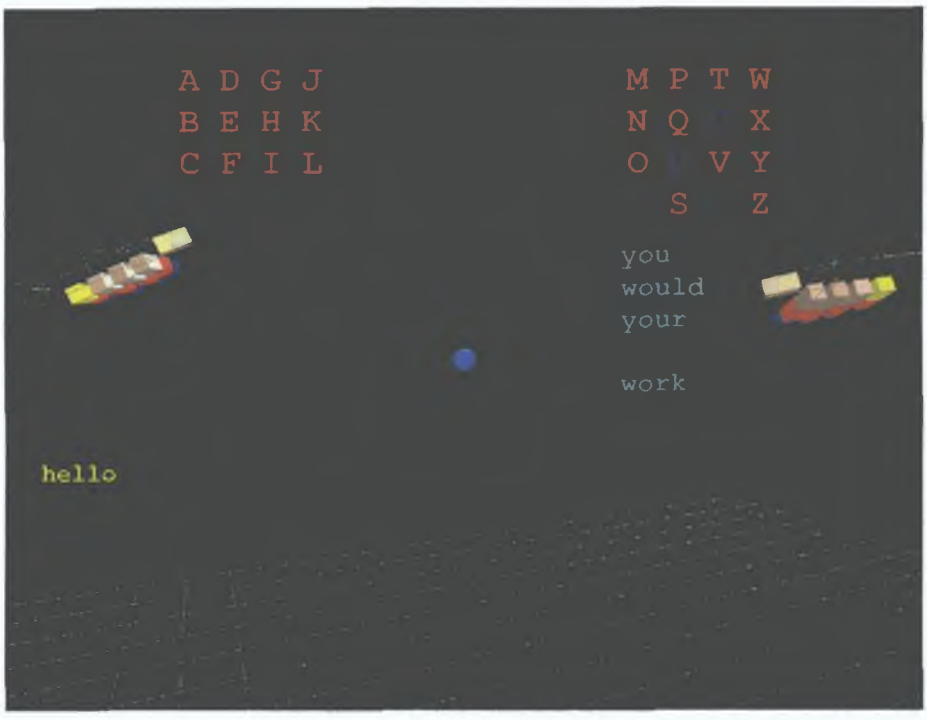


Figure 6.5: Word prediction and completion in use

content-rich environments allowing users to scan or ignore the complete words as they type. Thirdly, the use of datagloves to select words from the predicted or complete words means that any one-handed interaction technique used to select objects from the predicted word list, can be mirrored on the other hand for the complete word list. By having mirrored interaction techniques, the user must only learn one technique, reducing the cognitive load.

Figure 6.5 shows our final visual representation of the virtual keyboard, as well as predicted and complete word lists. Predicted words are offered on the left, with complete words offered on the right. Having decided on the visual presentation of the system, we must then develop interaction techniques with which to use it. In the next section we will consider interaction techniques possible with the 5DT dataglove.

6 4 5DT dataglove· Possible selection techniques

In Section 2 4 2 we reviewed the 5DT dataglove 5 or *5th Glove* To briefly recap, the 5DT glove uses proprietary fibre optic based flexion technology to measure overall finger and thumb flexion The 5 sensor model, which we use during the course of our experiments, does not measure finger abduction It is fitted with a 2-axis tilt sensor, which measures 120 degrees of *pitch* and *roll* The glove does not measure *yaw* (Figure 6 6), nor does it track the hand's position in 3D space Although measuring yaw and 3D position is possible with addition of extra sensing equipment, the interaction techniques considered here are limited to those suitable for use with just the sensors supplied with the glove Without adding separate 3D positional tracking technology, many of interaction techniques for selection discussed in Section 2 5 5 are unfeasible However, although the classic *natural* interaction with objects typical of VR applications may not be possible, there exist sufficient interaction possibilities to create an effective technique for text-entry with ambiguous keyboards The five finger sensors allow for posture recognition without the need for user training Combined with this, the tilt sensor offers 2-DOF, making it suitable for menu interaction, and allows for an adapted 2D pointing technique Finally, many of the techniques discussed in Section 2 5 5 use 6-DOF for selection Although well suited to the selection of virtual objects, 6-DOF interaction is not needed for the tasks we wish to perform Accurate control of 6-DOF adds an unnecessary burden upon users when interaction is possible with fewer degrees of freedom

2D pointing Although possible, 2D pointing is somewhat unnatural using the 5DT glove, pointing is usually performed through the 2-DOF combination of pitch and yaw (Figure 6 6) These correspond to movement in the XY plane The 5DT dataglove does not measure yaw, but an adapted pointing technique can be created by mapping the roll to the Y axis in place of yaw However,

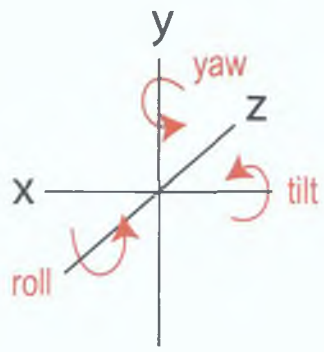


Figure 6.6: 6-DOF movement of the hand. Movement along, and rotation about the 3 axes

although replacing yaw with roll maintains the 2-DOF needed for 2D pointing, the movement necessary to select objects is unnatural and not intuitive. Pointing is more akin to directing a mouse pointer than controlling a ray emanating from the hand.

Gesture recognition Theoretically, the five fingers of the 5DT data glove allow for the recognition of 32 simple postures (Section 48, Page 2.4.5). With two datagloves, this figure increases to 1024 if performed simultaneously. Although many of these may be impractical, there exist many comfortable gestures. However, without a clear identifiable mapping between a gesture and an object, users may be forced to remember seemingly arbitrary gestures.

One solution to this is the use of tulip menu style mapping of fingers to items. Here each individual finger maps to an item and the user simply flexes the finger which maps to the item they require. The benefit of this technique is the clear visual indication of the gesture mapping, which means that users do not need to remember gestures.

1-DOF menus 1-DOF menus, as suggested by Shaw and Green (1994), are also possible using the 5DT data glove. These menus could be implemented using two basic techniques. Firstly, the degree of roll or pitch could map directly to the menu position. Pitching the glove fully upwards would highlight the topmost

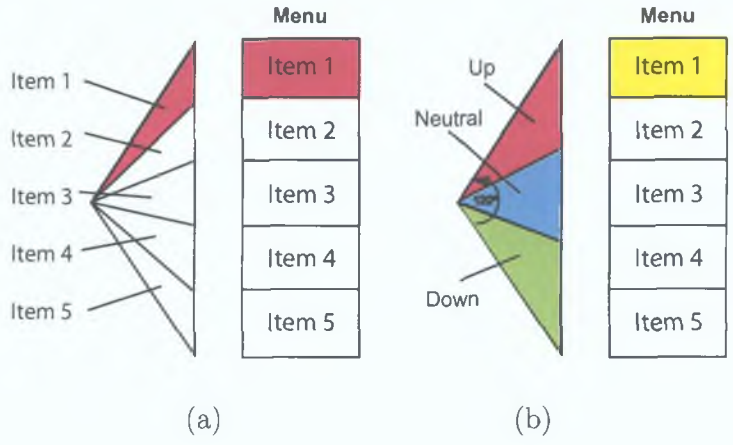


Figure 6.7: Possible 1-DOF selection techniques using the 5DT glove. (a) The highlighted menu item is mapped directly to the pitch or roll angle. (b) Moving the highlighted section is achieved by pitching the glove above or below the neutral zone.

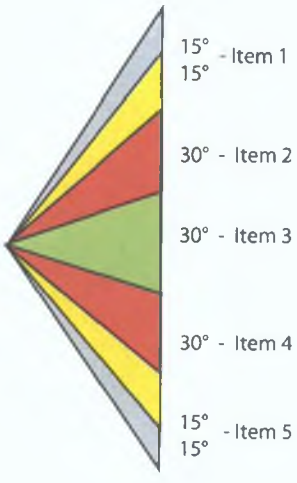
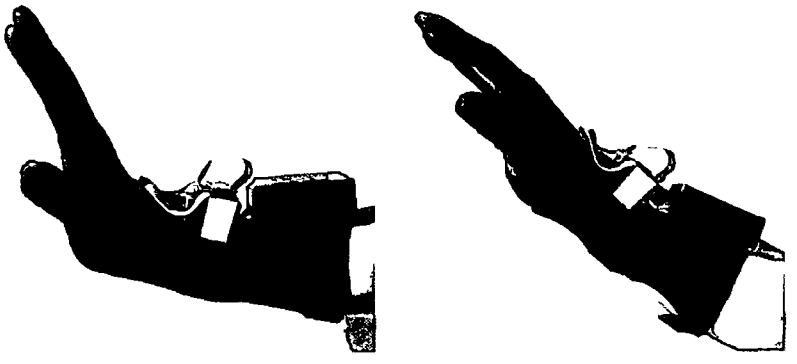


Figure 6.8: Direct mapping of angle to menu item. When the pitch or roll angle passes the threshold of 120 or 0 degrees (indicated by the grey areas), the top and bottom menu items can still be selected. Thus, the effective range of the glove is increased, allowing a greater proportion of the angle to be allocated to internal menu items.

menu item, pitching the glove downwards would select the last menu item, and intermediate menu items would be selected by pitching the hand accordingly This technique depends on being able to divide the available tilt range of the dataglove evenly between menu items, such that sufficient range is assigned to each menu item Here, there is an inherent trade-off between list size and usability, the larger the list, the smaller the angle allocated to each menu item, and the finer the movement necessary for accurate selection To achieve a list length of 5, we divide 120 degrees by 5, which corresponds to an allocation of 24 degrees to each item in the list In practice, for 5 item menus, a larger proportion of the 120 degrees can be allocated to internal menu items As users pass the limit of the gloves effective measuring angle (i e more then 120 degrees of movement) the tilt sensor continues to indicate the angle is at its limit (0 or 120 degrees) Thus, the effective angle of the top and bottom items is larger then 120 degrees, and is in fact closer to 150 (Figure 6 8) In our system, 15 degrees are allocated to the top and bottom of lists, with 30 degrees allocated to each of the internal list items However, even with 30 degrees allocated to each menu item, the selection of items still requires a fine degree of precision Furthermore, the movement of the hand when a *select* gesture is being formed to choose the highlighted menu item, can cause the angle of the hand to change, leading to accidental selection of neighbouring list items A more formal evaluation of the technique is discussed in Section 6 6

The direct selection technique can be used by mapping either the roll or pitch of the glove Each has benefits and drawbacks The pitch motion is more intuitive, users pitch their hand up to move up the list, and down to move down the list In comparison, rolling the hand inwards and outwards does not map as intuitively However, the placement of the tilt sensor on the glove results in pitch readings that do not accurately reflect the orientation of the hand (Figure 6 9a) As a consequence, larger pitch motions are needed to affect the orientation of the sensor (Figure 6 9b) In contrast, the position of the tilt sensor has no effect on



(a) (b)

Figure 6 9 Tilt sensor movement as a user bends their hand Due to the position of the sensor (indicated by the white rectangular marker at the wrist), pitching the hand at the wrist has little effect Users must pitch their arm at the elbow to effect the sensor

the roll sensor Thus, although less intuitive, the roll sensor requires significantly less arm motion

An alternative to this direct mapping technique, is to use a neutral zone, above or below which the current menu selection moves up or down We shall refer to this as a *click* selection technique, as the concept is similar to that of pressing an up or down button The advantage of this is that a greater range (40 degrees) can be assigned to each position, and list sizes have no limit The downside of this technique, is that selection of objects further down the list takes longer, as users must return to the neutral position after each *click* However, as lists can be sorted according to their probability, the need to select distant list objects may occur with sufficiently low frequency, as to warrant its use As with the direct selection technique, the roll can also be used in place of the pitch for the click technique

Each of these techniques is used to highlight items in a menu Once an object has been highlighted, it must be selected through the use of a selection command The options for selection are either a time-out after an item has been selected for a period of time, or selection using a gesture Therefore, selection

using 1-DOF menus will potentially need one motion to highlight an object and another to select it Although 1-DOF menus are likely be slower then selection with gestures, the advantage of the technique is the reduced gesture set needed to select all items

6 5 Interaction techniques for predictive text-entry

Having defined our taxonomy in Section 6 3 and explored the selection techniques possible in Section 6 4 we can now create potential interaction techniques for our ambiguous text-entry system

6 5 1 Selection of ambiguous keys

Selection involves choosing between 8 possible keys From our taxonomy, we know that the selection of keys is comprised of the 3 sub-tasks common to selection Indication of key, confirmation of selection, and feedback

Indication of key Each of the three selection techniques discussed in Section 6 4 can be applied to the selection of ambiguous keys However, the most obvious candidate is the mapping of finger flexion gestures to keys This maps closely to our interaction with physical keyboards, and allows for the direct selection of each of the eight keys with one gesture

Confirmation of selection Unlike the selection of *exit* from a system menu for example, confirmation of our selected key would slow and frustrate users Therefore, a more plausible solution is to assume automatic confirmation of selection, with users correcting any mistakes

Feedback Using a regular keyboard users receive tactile feedback as they strike each key When using gestures there may be no such feedback Therefore, aural and visual feedback provide useful affirmation of key selection Aural feedback

confirms that a gesture was recognised, while visual feedback highlighting the selected key provides confirmation that the gesture recognised was indeed the intended one

Feedback is also necessary to indicate when a gesture, although recognised, is not valid As each key is struck, the current sequence of ambiguous keys is compared to a dictionary of potential words If no matching words exist, then the user has either selected an incorrect key, or is attempting to type a word which is not in the dictionary Thus, visual or aural feedback should indicate that a gesture was recognised, but that the system dictionary contains no words matching the current key sequence

6 5 2 Selection of predicted and complete words

Here, the selection involves picking a word from a list of ambiguous words, or choosing to finish an incomplete word from a separate list As with the selection of keys, the selection of words is comprised of the 3 sub-tasks common to selection indication of the desired word, confirmation of selection, and feedback As mentioned in Section 6 3 1, any one-handed technique used to select predicted words can be mirrored to select ambiguous words

Indication of desired word As with the selection of ambiguous keys each of the three selection techniques discussed in Section 6 4 can be applied to the selection of predicted and complete words However, unlike ambiguous words, the ideal technique is less obvious

On average, the selection of words will only occur every 6th interaction technique, the average length of a word is five letters, at which point a word will be selected Therefore, if speed is to be sacrificed for either technique, it should be for word selection Combined with this, the words to be selected do not have the same proprieties as the ambiguous keys Specifically, words are ordered according to their probability As a result, each item will not be selected with the same

list position	predicted words	complete words
1	97 61	49 71
2	1 90	18 42
3	0 30	13 34
4	0 09	10 51
5	0 04	8 03

Table 6 1 Selection percentage of list items for word prediction and word completion

frequency Table 6 1 shows the breakdown of words selected from predicted and complete lists respectively, where words are ranked according to their probability Figures shown reflect selection statistics for an alphabetic keyboard using a tri-gram language model trained on 25 million words, however they reflect the trend for QWERTY, and optimised keyboards

As we have discussed, 2D selection, although possible with the 5DT glove, is unnatural, and thus the least attractive option Similar to key selection, words could be mapped directly to finger flexion gestures However, this would require a change of mode, to indicate the users intention to switch between selecting keys and selecting words A gesture, which didn't map to any key, for example a fist, could be used to indicate this change of mode Another option is to pitch or roll the hand to one angle during typing to select keys and to an alternative angle to select ambiguous or complete words The benefit of this technique is that all words can be selected directly, the drawback of this technique is that two motions are needed to select a word

1-DOF menus, offer an attractive alternative 1-DOF menus, unlike mapping fingers to words, would not require a mode change If the angle comfortable during typing maps to the most likely word on the list, then only the selection gesture is needed to select the word If the desired word is not highlighted by default, then users can pitch or roll the hand accordingly to highlight the desired word, before selecting it This technique is particularly suitable for selection of predicted words, where the most likely word is predicted with high frequency

Confirmation of selection If tulip menus are used, then indication and confirmation, like that for keys, become one For 1-DOF menus, confirmation of selection is needed to confirm selection of the currently highlighted word through the use of a defined gesture or time-out The flexion of the thumbs is a natural candidate for the selection gesture, as during typing on a regular keyboard, the space-bar is hit with the thumb to indicate the end of a word

Feedback As with key selection, visual and aural feedback is used to compensate for the lack of tactile feedback Visual feedback is used to indicate the currently selected word, while aural feedback is used to indicate that the selected word has been confirmed as the desired word

6 6 Evaluation of selection techniques.

In Section 6 4 we discussed several options for the selection of items from a list using 5DT datagloves To recap, these were

- Direct mapping of the pitch or roll angle of the glove to the highlighted list item
- Highlighting a list items using a *click* up or down motion with either pitch or roll
- Mapping the flexion of each finger to a menu item

We conducted an experiment to evaluate the characteristics of each technique This evaluation measured the speed, accuracy, and user preference of each selection technique

6 6 1 Experiment details

Five conditions were tested in our within-subject experiment Direct mapping of both tilt angles to the highlight menu item, *click* highlighting with both roll

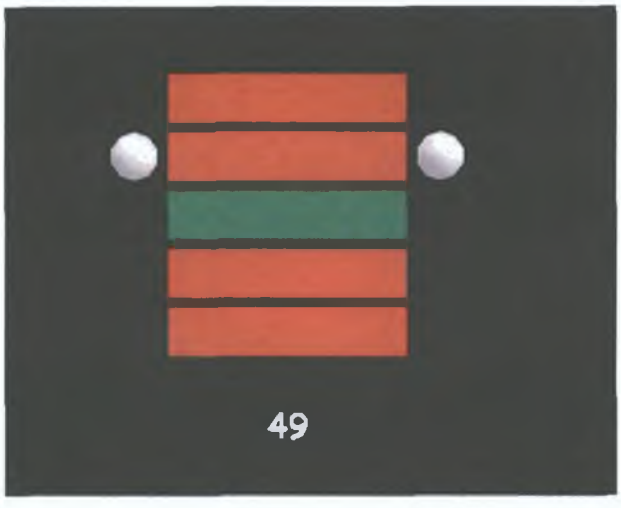


Figure 6.10: Selection experiment

and pitch, and finally mapping each finger to a list item. For this finger mapped technique, users rolled their hand outwards (which would correspond to a mode change during real use) and then selected the menu item by bending the corresponding finger. This technique was used to simulate how the system would be used if combined with a similar technique for selecting letters.

Users were presented with a list, represented by 5 rectangles (Figure 6.10). The users were asked to select an item from the list 50 times. The item to be selected was indicated by two white circles positioned on either side of the list. When the user selected the correct list item, the circles turned green. If the user selected an incorrect list item the circles turned red. Regardless of the outcome, a new item was indicated for selection. This was repeated 50 times with the number of remaining selections indicated at the bottom of the list. The time between the indication of the desired list object by the system, and the selection of a list item by the user, and the accuracy of selection were recorded.

For the condition in which each finger mapped to a list item, the list was coloured red while the hand was in the horizontal *typing* position. When the user rolled their hand 90 degrees outwards, the list turned green indicating that a list object could be selected. The user could then bend the finger which mapped to

the list item After a user had selected an item, they were instructed to return their hand to the horizontal position, at which point another desired list item was indicated

For the other four conditions, which didn't require a mode change, four of the list items were coloured red, while the list item currently highlighted was coloured green Participants highlighted the intended list item and flexed their thumb to select it Users were then instructed to return their hand to the horizontal typing position, at which point another desired list item was indicated

The main hypotheses were

- The techniques directly mapping tilt angle to the selected item, and mapping fingers to list items would prove significantly faster then the *click* selection techniques
- The error rates for both *click* techniques would prove significantly lower then the equivalent direct tilt techniques

Participants Eight users participated in the selection experiments, 2 female, and 6 male Participants were postgraduate students and staff volunteers from the School of Computing

Equipment Participants interacted with the system using 5DT datagloves These were calibrated individually for each participant before each recording The total range of motion of each finger was measured After calibration, fingers were considered flexed if the Euclidean distance to a *flex* gesture was closer then the Euclidean distance to a *flat* gesture, as described in Chapter 5

Procedure At the beginning of each experiment, each of the five selection techniques was explained and users were given an opportunity to familiarise themselves with each technique Users were then tested using each technique Before each technique was tested, users were reminded of the technique to be tested

Test conditions were counterbalanced to avoid any learning effects Users were asked to perform as quickly as possible while maintaining as high a degree of accuracy as possible

Finally, at the end of the experiment users were asked to rank the five techniques in order of preference

Results Figure 6 11 shows the average selection times recorded from participants Values indicate the average time taken to select each list item, and the overall average selection time for each technique Within-group one-way ANOVA reveals that the selection technique has a significant effect on selection speed ($F_{(4,28)} = 34\,816$, $p < 0\,0005$) The graph confirms our hypotheses The finger mapping technique, shows the most consistent selection times Here, each list item can be selected directly Consequently, there is no difference between the time to select the first or last item In contrast to this, the *click* selection techniques are fast for selecting the middle list item (which is highlighted by default), but slower to select items further from the centre Similarly, the direct angle mapping techniques are fast for selecting the middle item, slower at selecting the 1st, 2nd, 4th and 5th items However, in contrast to the *click* technique there is little time difference between the time to select these items Pairwise comparisons reveal that average selection times for both *click* select techniques are significantly lower then both the direct tilt and finger mapping techniques ($p <$ 0 01 after Bonferroni adjustment for multiple comparisons)

Of note is that the selection times for the central list item are faster for all tilt techniques, compared to the finger mapping technique This is because two gestures are needed to select an item with the finger mapping technique, an outwards roll, followed by a finger flexion In contrast, the tilt techniques are centred on the central list item by default, so only a *select* gesture is needed Based on this, an adapted, hybrid gesture selection technique is suggested, where the hand may remain horizontal to select the top object with the thumb, or roll

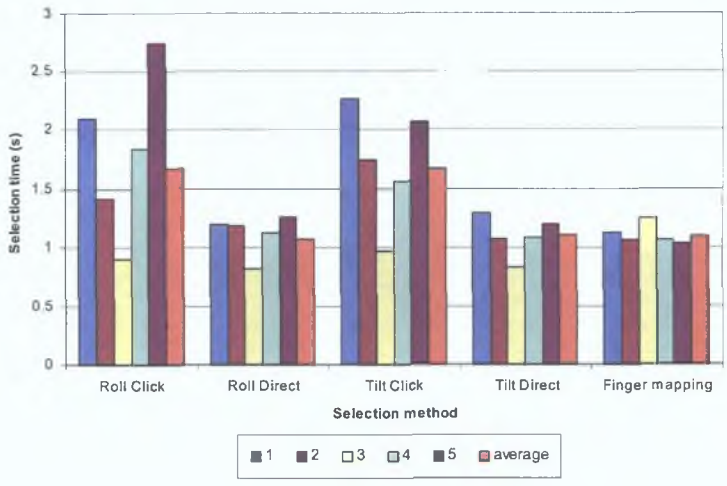


Figure 6.11: Comparison of selection times for the five techniques tested

the hand 90 degrees to the vertical position, and select the other four list items. The advantage of this technique will become more apparent when we discuss word list frequency.

Figure 6.12 shows the error rates of the five selection techniques tested. Again, within-group one-way ANOVA reveals a significant effect of selection technique on accuracy ($F_{(4,28)} = 7.990,\ p < 0.0005$). The two direct angle mapping techniques perform poorly, with average error rates of over 10 percent. T-Tests reveal that the error rate for both click techniques is significantly lower then their direct angle equivalents. ($t = 3.473,\ df = 7,\ p = 0.005$) and ($t = 2.544,\ df = 7,\ p = 0.019$) for roll and pitch techniques respectively.

Our own observations revealed that accuracy was poor for direct angle selection because users struggled with the fine control of the pitch or roll angle needed to highlight a list item. Slight movements in the hand caused the highlighted item to flicker. Even when the hand was steadied on the correct item, forming the select gesture often caused small movements which altered the highlighted item just as it was being selected.

In contrast, larger movements were needed for the *click* techniques, which, although slower, lowered the chance of the highlighted item changing as the user

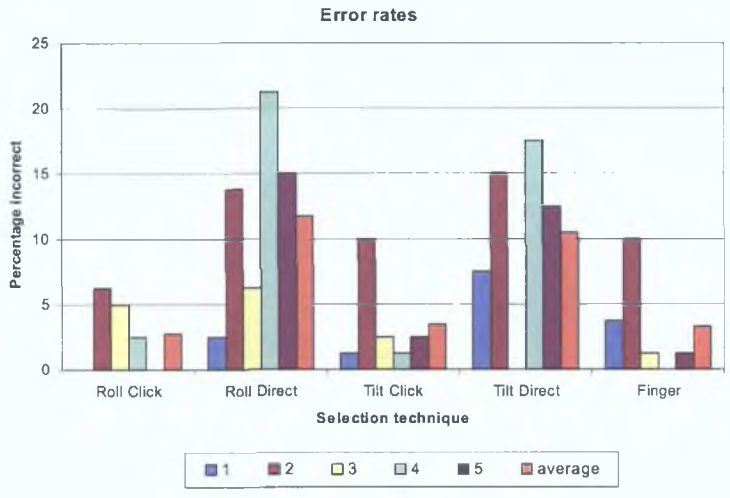


Figure 6.12: Comparison of error rates for the five selection techniques tested

	Actual				
Intended	1	2	3	4	5
1	77	3	0	0	0
2	1	72	7	0	0
3	0	0	79	1	0
4	0	0	0	80	0
5	0	0	0	2	78

Table 6.2: Confusion matrix for finger mapping selection technique

formed the select gesture.

Finally, from observation of experiments, the majority of errors for the finger mapping technique appeared to be caused by the mental mapping of the index finger to the top item rather than the thumb. Similarly, the second list item was selected with the middle finger rather then the thumb. This was supported by a confusion matrix contrasting the intended gesture, and the actual gesture recorded (Table 6.2).

Figure 6.13, shows the average user preference rankings of all five techniques (lower is better). The finger mapping technique was preferred by almost all users. Both *click* techniques were least liked, as, although accurate, they proved slow and frustrating for users.

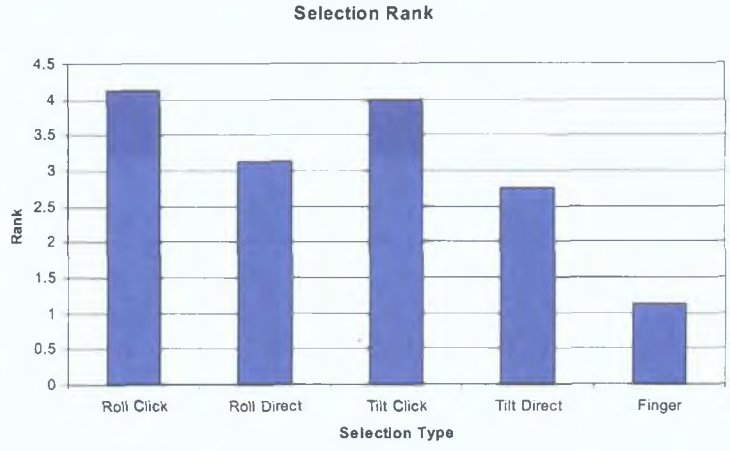


Figure 6.13: Comparison average rankings for the five selection techniques tested (techniques were ranked 1-5, thus lower is better)

	Roll Click	Roll Direct	Pitch Click	Pitch Direct	Finger mapping
User ratings	4.13	3.13	4.00	2.75	1.13
Accuracy	2.75	11.75	3.50	10.50	3.25
Speed	1.67	1.07	1.67	1.10	1.09

Table 6.3: Recap of selection characteristics

Discussion Table 6.3 shows a recap of each of the 5 techniques, comparing average selection speed, accuracy, and preference. Based on the results of these experiments, mapping finger flexion to menu items would seem the best solution. It was the most preferred method, and was fast and accurate. However, the experimental conditions did not reflect the menu interaction likely during actual use. In particular, the frequency which which menu items were selected did not reflect the true likely usage. In the experiments list items were selected at random, with users selecting from each position ten times. In reality, word lists are ranked according to probability. As a result, certain menu items will be selected with a significantly higher frequency then others. Recall from Table 6.1, that for word completion, the most likely menu item will be selected with a frequency of almost 50 percent, while for word prediction, the figure approaches 100 percent.

If we apply this data to the selection times, we find the characteristics of each

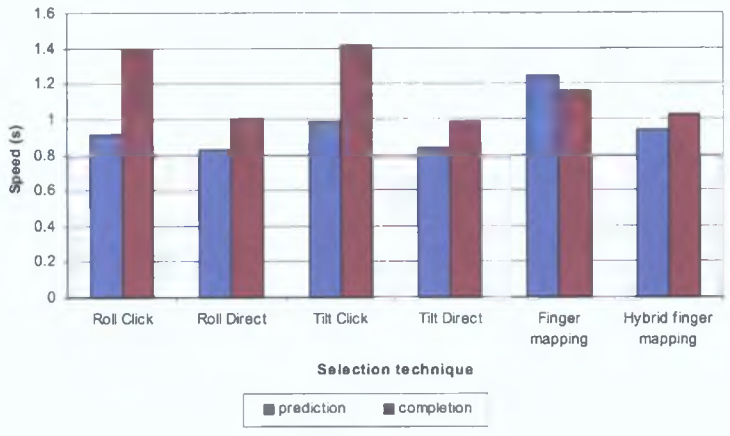


Figure 6.14: Comparison of likely selection times for the five techniques tested if used on ordered word lists

technique changes. Figure 6.14 and 6.15 show the likely accuracy and speed of each technique if used in practice to select predicted and complete words from ordered selection lists.

The speed advantage of regular finger mapping is reduced, as the majority of selection occurs on the fastest position for all the other techniques (horizontal thumb flexion). The click techniques become much more attractive, offering low error rates, and high average speed for word prediction. Although not tested, the predicted hybrid gesture selection times are estimated. Here, thumb times are based on the average selection time for the central list item for both click techniques (where selection is performed with the flexion of the thumb when the hand is horizontal as proposed in the hybrid finger technique) and the times for other fingers are based on those recorded for the regular finger mapping technique.

6.7 Conclusions and recommendations

In this chapter we have discussed interaction techniques for ambiguous text-entry. Specifically, we have shown the value of decomposing a larger task into sub-tasks, which can in turn be used to choose appropriate interaction techniques. We have discussed some of the potential selection techniques which are possible using

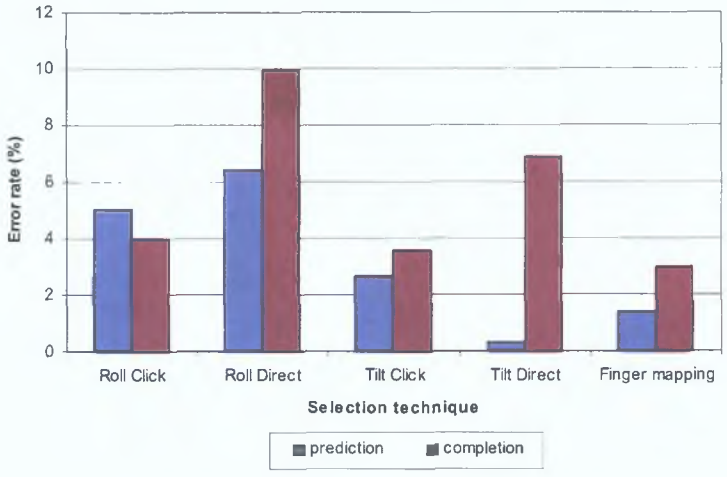


Figure 6.15: Comparison of likely selection accuracy for the five techniques tested if used on ordered word lists

the 5DT dataglove, and, combining these with our taxonomy, suggested several options for ambiguous text-entry. Ultimately, the choice of techniques depends on several factors. Speed, accuracy and likeability all have an impact on the selection of appropriate techniques.

Finally, we conducted experiments to examine the speed, accuracy, and user preference of five proposed selection techniques. Based solely on results of our experiments, directly mapping finger flexion to list items proves the most attractive option for list selection. However, when considered in the greater context of their use as interaction technique for word selection within our text-entry system, word ranking and selection frequency must be taken into account. When these are considered, the *click* techniques and the hybrid finger mapping technique are the most appealing candidates. The final decision may be determined by gesture recognition accuracy. If gesture recognition is hampered by sympathetic bending, then the *click* techniques offer the best solution, otherwise, the hybrid finger-mapping technique would seem preferable.

Chapter 7

System evaluation

7 1 Introduction

The design of our text-entry technique followed the iterative evolution model common in many design methodologies, as outlined in Section 1 4 Formative evaluation was used throughout the design to gain greater insight into the system, to identify useful features, and finally to detect any potential problems By developing early working prototypes, we could evaluate the viability of the underlying text-entry technique This evolving prototype also allowed us to assess individual aspects of the system, such as possible interaction techniques and gesture recognition methods Following informal formative tests, we conducted a summative evaluation of the system Both quantitative and qualitative analyses was used to compare various aspects of the system Here, features such as keyboard configuration, and the use of visual aids, were examined by measuring both user performance and preference

In the following chapter we will discuss findings from our informal formative evaluation conducted during the system design, and our formal summative evaluation

7 2 Formative evaluation

Formative evaluation was conducted during the development of the system This allowed insight into usability of the system, and highlighted potential problems with it While conducting this informal *hallway* testing, quantitative data was recorded However, more focus was given to qualitative data, where insight was gained from observing users, who were encouraged to *think out loud* as they interacted with the system This gave an insight into any problems users might have with the system Users were also encouraged to share any positive aspects of the system

7 2 1 Experiment details

The primary objective of early formative experiments was to confirm the potential of the system as a viable text-entry technique, and to identify any potential problems with its use Other objectives included assessing the use of word completion as an aid to increase throughput, and to examine possible alternative keyboard layouts and interaction techniques The gesture needed to signify a key-press wearing the datagloves was significantly larger, and more pronounced than that needed on a regular keyboard Therefore, it was hypothesised that the interaction with the virtual system would be sufficiently foreign as to reduce the effects of muscle memory, thereby resulting in poor performance of the QWERTY keyboard layout It was further hypothesised that alternative keyboards, such as those designed for easy searching, or optimised for prediction accuracy might offer potential gains over the QWERTY keyboard layout

Participants Five users participated in the first evaluation of the system, 3 male, and 2 female All participants were computer science postgraduate students, and where thus very familiar with a QWERTY keyboard layout Short typing tests on a standard keyboard revealed that participants had a regular

text-entry speed ranging from 25 to 43 WPM

Equipment Participants interacted with the system using 5DT datagloves These were calibrated individually for each participant before each recording The total range of motion of each finger was measured After calibration, flexion greater then 60 percent of the overall range was considered *flexed* while anything less then 50 percent was considered *extended*

The prediction engine used a dictionary of the most frequent 2000 words of the Brown corpus, with words ranked according to their frequency Any words appearing in the test text which were not originally in the 2000 word dictionary were added before the test

Procedure Users were shown a visual representation of the virtual keyboard, (Figure 7 1), and the ambiguous nature of the finger-to-key mapping was explained Due to its availability on most mobile phones, all five participants were familiar with the concept of predictive spelling on ambiguous keyboards However, only 3 participants actually used the feature on their own phones Users were shown how to interact with the system, how to select predicted and complete words, and how to delete incorrect words and letters using *fist* gestures

Two within-subject variables were used for the tests Firstly, keyboard layout was changed QWERTY, alphabetic, and optimised keyboards were tested The second variable was the ability to use word completion, which was either active or inactive Each user participated in six two-minute tests, three tests with each keyboard without word completion, and three tests with it The six conditions were counterbalanced to offset any learning effects User's text-entry times, and the number of letters saved by word completion were logged by the system Throughout the procedure, users were asked to *think out loud* to articulate any problems they identified in the system Users were observed throughout the procedure to evaluate the performance of the system, and to monitor errors

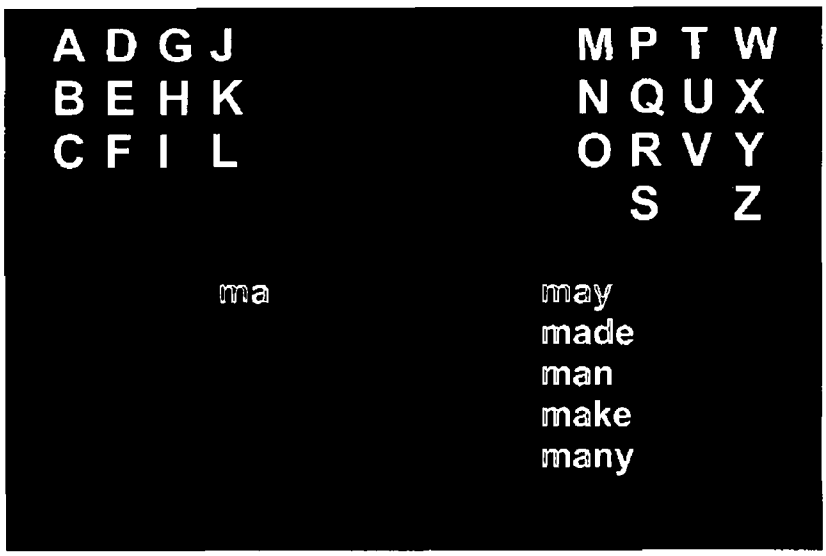


Figure 7 1 Virtual ambiguous keyboard

made by the users or the system

Finally, having completed each of the 6 conditions, users were asked to briefly evaluate and comment on alternative selection techniques During the tests, users selected letters with the direct mapping of the roll angle of the glove to the highlighted item, as described in Chapter 6 After the experiment, users were shown the iterative, *click* selection technique, which was proposed as an alternative to the direct selection technique, and asked to compare them

Results and observations The primary objective of the initial tests was to confirm the potential of the text-entry technique The positive response from users, and the recorded text-entry times confirmed the viability of the predictive text-entry techmque On average, users typed at a speed of 9 WPM during the sixth test, which corresponded to ten minutes practice Users found the concept easy to understand, and were quick to begin typing

Also of primary concern during the evaluation, was the identification of any potential problems During the initial design of the prototype system, informal testing had been conducted by the author However, the larger user group quickly

identified gesture recognition as a key factor in the accurate use of the system In particular, users found that the system repeatedly mis-interpreted a little finger key-press as a ring key-press This was due to sympathetic bending of the ring finger when the little finger was bent The problem was further exaggerated for the two female subjects, whose smaller hands resulted in poor recognition This was primarily due to the nature of the dataglove's *one size fits many* design These recognition problems represented the greatest source of frustration for participants

Another problem which became evident during testing was simple user error Even when the system's gesture recognition performed accurately, users often simply pressed the wrong finger After searching for and finding the desired letter, users frequently depressed the wrong finger, not through lack of concentration, but by simply confusing the finger-to-key mapping Observation of this recurring phenomenon, combined with user feedback, offered insight to its cause Users felt the most likely cause of the problem was due to the way in which they mentally mapped their fingers to the columns Users regularly viewed the index finger column as the *first* column, and as a result, bent the first digit on their hand, the thumb, to select it Similarly, the second column was often selected with the index rather then middle finger However, unlike recognition errors, which were more likely to go unnoticed, users quickly recognised their mistake and deleted the unintended letter Isolating the reason for this problem is difficult, as software cannot determine the difference between a selection error by the user, and a recognition error by the system The only reliable method of detection is if participants give an aural cue to an observer while being tested, to indicate that *they* were at error rather then the system

Secondary objectives of the experiments included evaluating alternative keyboard layouts, and the potential use of word completion Here, analysis of quantitative data recorded during the experiments revealed that no keyboard performed significantly better then others The QWERTY layout resulted in the fastest

text-entry time (14 WPM), and on average resulted in higher WPM rates then alphabetic and optimised layouts However, as expected, users – even those that were comfortable touch typists – were forced to visually hunt for keys, as the muscle memory, which aided in regular typing, had little effect

Also examined was the potential advantage offered by word completion Here the quantitative results were more conclusive The use of of word completion significantly improved text-entry rates On average, users saved 20 percent of characters, increasing WPM by over 30 percent, through the use of word completion However, there was a further advantage of word completion which was not foreseen Throughout the experiment, the focus of users was often broken This occurred for several reasons Users became frustrated if they could not locate a letter on the keyboard, or if the system failed to instantly recognise an intended gesture, or if it mis-classified a gesture This resulted in users occasionally becoming *lost* in longer words, unsure of how many letters they had typed and which letter to type next Because of the ambiguous nature of the system, while typing longer words, the beginning of a sequence regularly matched alternative full words Consequently, if users momentarily lost focus, they could not quickly see which letter they were on They had to count the letters in the current offered word, and then count the corresponding distance into the required word, to determine which letter they should type next For longer words this was quite difficult and frustrating Often, users simply gave up, deleted the word, and began typing afresh The ability to complete words significantly reduced this problem as word completion offered a method to quickly finish longer words, essentially circumventing the problem For this reason, as much as any perceived increase in efficiency, users expressed a strong preference to the use of word completion

When comparing the iterative *click* selection technique to the direct mapping technique, all five participants tested preferred the direct mapping technique, feeling the iterative technique was too slow in comparison

During the tests, users were able to delete incorrect characters or whole words

using two delete gestures Creating a fist with the left hand deleted the entire word, while creating a fist with the right hand deleted the last typed character It quickly became apparent that this mapping confused users, who could not remember which hand mapped to which function This resulted in unnecessary deletion of almost complete words when a user, attempting to delete an incorrect character, deleted an entire word by creating the delete gesture with the wrong hand

7 2 2 Conclusions

The results from initial formative testing confirmed the viability of predictive text-entry as a text-entry technique that was both efficient, and easily understood and adopted Users quickly grasped the basic concept and could begin typing immediately The tests also identified problems which simple task analysis did not specifically, the accurate recognition of key-press gestures This was due in part to the physical characteristics of the glove, and in part to the dexterity of individual users

The tests demonstrated clearly the benefits of word completion, both in terms of overall throughput, but also as a method for reducing the likelihood of user frustration as they become *lost* while typing long words

The results of varying keyboard layout, although not statistically significant were telling Although all participants were computer science postgraduate students, intimately familiar with the QWERTY keyboard layout, users did not instinctively know where letters resided on the keyboard, and had to resort to the *hunt and peck* style typing of novice typists This is in line with findings by Bowman *et al* (2001b), and reaffirms our view that potential gains can be achieved from alternative keyboard layouts

Finally, the use of a small 2000 word dictionary resulted in relatively few clashes during text-entry This was of most benefit to the QWERTY layout, whose unbalanced keyboard layout might ordinarily have resulted in a higher

clash rate With a larger dictionary the value of optimised keyboards may have been more apparent

7 3 Summative evaluation

As the system evolved, a more thorough, systematic evaluation of key aspects of the system was undertaken This summative evaluation was carried out to contrast the effectiveness of several variations of the system As with the formative evaluations, quantitative and qualitative data was recorded However, in the summative evaluation, a more formal approach was taken to experiments A larger group of participants was used (49), allowing for more statistically significant results Users participated in longer experiments (40 minutes), were required to contrast subjective workload during the tests, and complete a post-hoc questionnaire afterward

For these tests, several aspects of the system were examined firstly, keyboard layout, secondly, keyboard size and finally the effect of the use of visual aids

Among the key questions explored were

1 Is there any effect in altering the keyboard layout from that of a universally recognised QWERTY to an alternative layout designed for faster searching or more accurate prediction?

2 Does the addition of visual aids simplify the mental one-to-one mapping of fingers to columns of letters?

3 Does the reduction of key count from eight to six keys adversely effect the typing speed?

4 What is the average speed of beginners using the system for the first time, and is there any correlation between users typing speed and their speed on a regular keyboard?

7 3 1 Experiment details

Two separate experiments were conducted, with users participating in one or the other Participants were allocated to each one based on the ability of the system to clearly identify intended gestures Users with strong sympathetic bending of the little finger which could not be resolved by the system were allocated to the first experiment

In the first experiment a keyboard layout comparison was conducted Here, four keyboard layouts were contrasted with within-subject tests a traditional QWERTY layout, an alphabetic layout, an optimised layout and a random layout Key-count was also contrasted in the first experiment, users with strong sympathetic bending in the little fingers were tested with six-key keyboards, where the little and ring finger corresponded to the same gesture Users without sympathetic bending were tested on full eight-key keyboards

The second experiment compared key-count with within-subject tests Here participants were limited to those without sympathetic bending Also tested was the potential advantage of visual aids to reduce selection problems The visual aid used was a graphical representation of the user's fingers behind the keyboard (Figure 7 2) This was used in an attempt to reinforce the column-to-finger mapping The aim here was to reduce the user errors caused by flexing the wrong finger to select a key

Participants Forty-one users participated in the summative evaluation of the system, 27 male, and 14 female All participants were students completing taught postgraduate studies in the School of Computing Students participated in the experiments as part completion of practical work for their HCI module and received credit accordingly

Equipment As with formative studies, users interacted with the system using 5DT datagloves These were calibrated for each individual user before each recor-

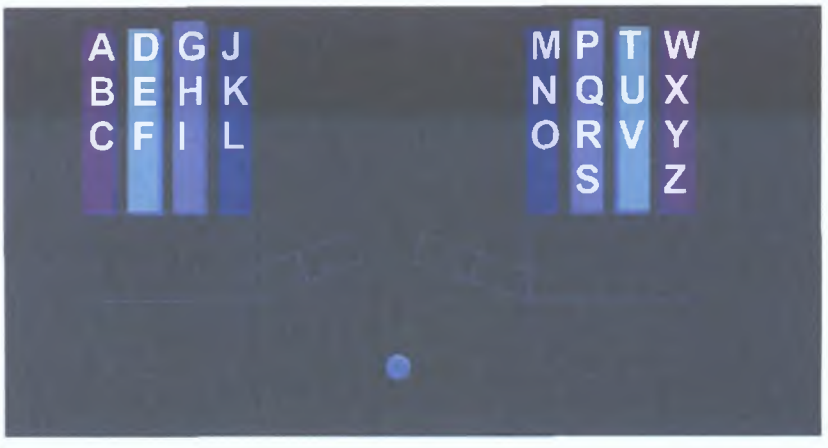


Figure 7.2: Graphical representation of gloves to aid proprioception

ding. The total range of motion of each finger was measured. After calibration, flexion greater then 60 percent of the overall range was considered *flexed* while anything less then 50 percent was considered *extended.* [1]

A tri-gram language model was used based on a 500,000 word corpus of books collected from the Project Gutenburg online collection of electronic texts (Gutenburg, 1971). This resulted in a dictionary of 15665 words, with over 400000 tri-grams.

Initial procedure At the beginning of each test users were asked to complete a brief test to measure their typing speed on a standard keyboard. Following this, participants put on, and calibrated the gloves. At this point, users were evaluated for test suitability. As detailed previously, the *one size fits many* nature of the gloves occasionally resulted in difficulties in recognition, particularly with female users with slight hands. If this evaluation indicated the presence of sympathetic bending of the little fingers, users were allocated to the first test, using the six-fingered keyboard. Otherwise users were evenly allocated to either the first or second tests. Of forty-nine potential participants, eight were deemed unsuitable

[1] Due to timetable constrains, the summative evaluation was completed concurrently with the gesture recognition experiments. Thus, the recommendations offered in Chapter 5 had not been implemented in the system at this point. However, the design of the tests, using six- and eight-finger keyboards, circumvented many of the problems caused by sympathetic bending.

due to incompatibilities with the datagloves Twenty-five users participated in the first experiment nine using six-column keyboard layouts and sixteen using eight-column layouts Sixteen users completed the second experiment

Subjective workload During the experiment users rated their experiences using a NASA-TLX (Task Load Index) rating system developed by Hart and Staveland (1988) The NASA-TLX system contrasted the subjective workload experienced by the participants under the various four conditions Participants were asked to rate the system according to six metrics mental demand, physical demand, temporal demand, performance, effort, and frustration level Each user rated the system four times, once after each condition

Post-hoc questionnaire Finally, participants were required to fill in a post-hoc questionnaire In it, they were asked to rate the four conditions in order of preference, and to make any further comments on the system

Experiment 1 keyboard layout and key count

Participants were presented with the virtual keyboard and the text-entry technique was explained This included explaining the nature of the dictionary-based ambiguous text entry how disambiguation was done not at the letter level, but rather at the word level Participants were told not to attempt to disambiguate individual letters as they typed, but rather to wait until they finished typing a word Selection of ambiguous words was explained as well as how to finish a word using word completion Finally, the delete gesture (a fist) was demonstrated Unlike in previous experiments, a fist gesture with either hand only deleted one letter

Users were then encouraged to become comfortable with the system and to try each of the demonstrated features A period of roughly five minutes was allocated for user orientation and practice Users trained on a random keyboard which was

only used for the duration of training This was to counter any learning effects which might bias other layouts

Once participants were comfortable with the use of the system, they completed four five-minute tests, one test for each keyboard layout Users typed short sentences designed to be read once and easily remembered (MacKenzie and Soukoreff, 2003) To counter the effects of learning, keyboard order was counterbalanced using balanced Latin squares (MacKenzie, 2002)

The main hypotheses of the experiment were

- The lack of muscle memory transfer from traditional keyboards would force users to search the unstructured QWERTY layout for each key Consequently, alphabetic keyboards, with their structured layout would prove easier to type on (reflected by a higher WPM), and would be preferred by users

- Due to is accurate predictions the optimised keyboard layout, equally as foreign as the QWERTY, should result in a higher WPM then QWERTY, and be favoured by users

- Subjective workload would reflect the difficulty in searching for letters, and selecting words Thus, alphabetic keyboards would offer the lowest ratings, followed by optimised, QWERTY, and finally random

Experiment 2 keyboard size and visual aids

Participants allocated to experiment 2 were familiarised with the system in the same manner as those in experiment 1, with each user being given 5 minutes training before beginning the tests Users were tested with six- and eight-key keyboards, with and without visual aids Again, as with experiment 1, the four tests were counterbalanced using balanced Latin squares

The main hypotheses of experiment 2 were

- The visual aids would reduce key selection errors, and thus increase WPM.
- The reduction in errors would be reflected in the subjective workload. In particular, mental workload, effort, and frustration should be decreased in the conditions were the visual aids are used.
- Eight key keyboards would produce a higher average WPM, as the decreased ambiguity would lead to more accurate word prediction.

7.3.2 Experiment results

Experiment 1.

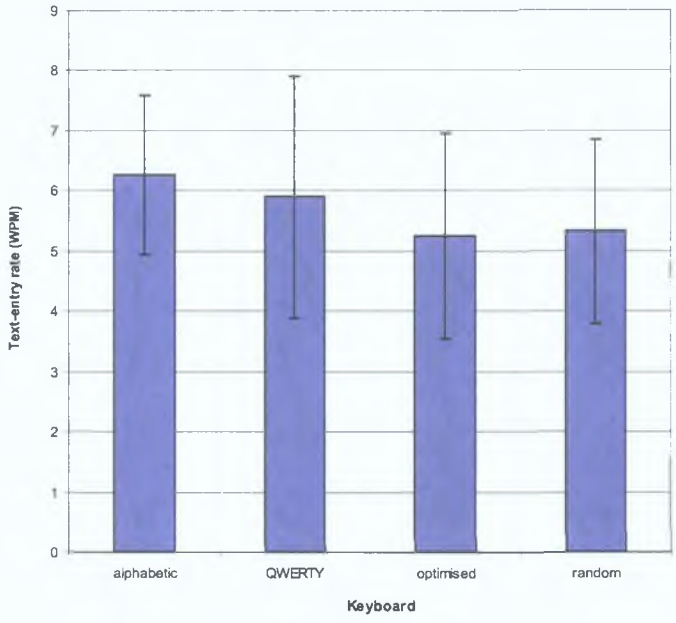


Figure 7.3: Contrast in user WPM with various keyboard layouts

Keyboard comparison Figure 7.3 shows a graph comparing the average WPM of all 4 keyboards. Within-subject one-way ANOVA reveals a significant effect of keyboard layout on performance ($F_{(3,72)} = 3.832$, $p = 0.013$), and confirms our hypothesis that the QWERTY layout would prove sub-optimal. Post-hoc inspection reveals that this is due to the alphabetic keyboard, which is significantly

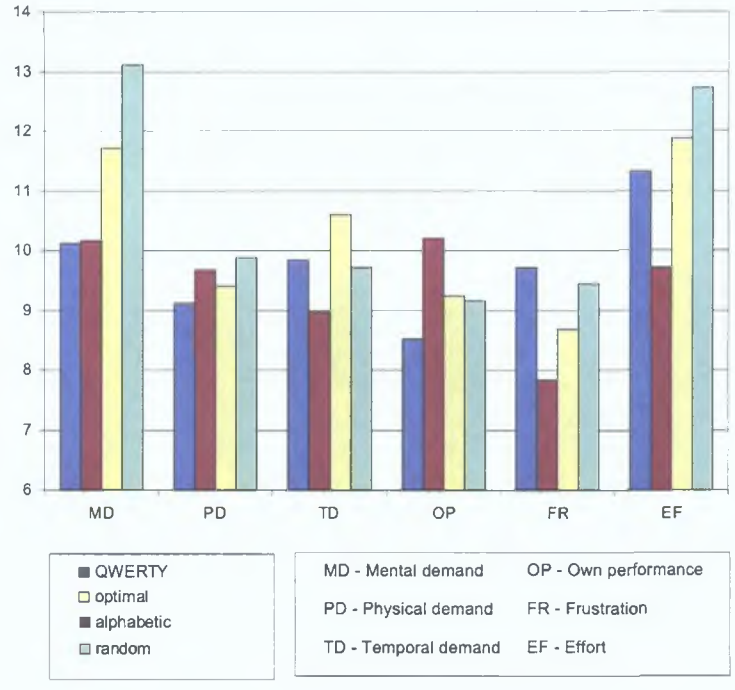


Figure 7.4: Contrast in participant subjective workload with alternative keyboard layouts

faster then both the optimum and random keyboards ($p = 0.029$ and $p = 0.014$ respectively, using the Bonferroni adjustment for multiple comparisons).

Subjective workload Figure 7.4 shows the average subjective workload of users for the 4 keyboard layouts. The alphabetic keyboard shows the highest perceived performance, and has the lowest frustration and effort levels. Although not statistically significant, the visible trend is carried across the range of workload measures.

User preference When asked to rank keyboards in order of preference, over 50 percent of participants ranked the alphabetic keyboard first (Table 7.1). A Friedman test for significance revealed a significant effect of keyboard on user preference ($\chi^2 = 11.976$, $df = 3$, $p = 0.007$) . This was in keeping with the subjective workload indicated with each keyboard layout, and the resulting text-entry rate.

Ranks	
	Mean Rank
Alphabetic	1 88
QWERTY	2 28
Optimised	2 84
Random	3 00

Table 7 1 Friedman test on user keyboard preference

eight fingers		six fingers	
with v/aids	without v/aids	with v/aids	without v/aids
10	6	11	5

Table 7 2 User preference for visual aids for 6 and 8 finger keyboards

Experiment 2

Visual aids The effects of the visual aids on selection accuracy proved inconclusive As selection mistakes could not be measured directly, the effect was measured by observing the text-entry rate of users Users expressed a mixed reaction to the aids, some found them useful, while others found them distracting This was reflected in the WPM observed, which showed no significant difference between typing speed with or without the use of gloves as selection aids

A post-hoc questionnaire of user preferences (Table 7 2) revealed that, despite the fact that it did not improve their typing speed, users preferred the use of visual aids for both six- and eight-fingered keyboards The NASA-TLX results (Figure 7 5) proved inconclusive and at times contradictory Visual aids had no significant effect in reducing user perceived workload

Contrasting six- and eight-finger keyboards Analysis of user text-entry speed with six- and eight-finger keyboards revealed no significant difference in WPM Users typed with an average speed of 5 7 and 5 9 WPM with six- and eight-fingered keyboards respectively Our post-hoc questionnaire revealed an almost even split between user preference Again, NASA-TLX results were incon-

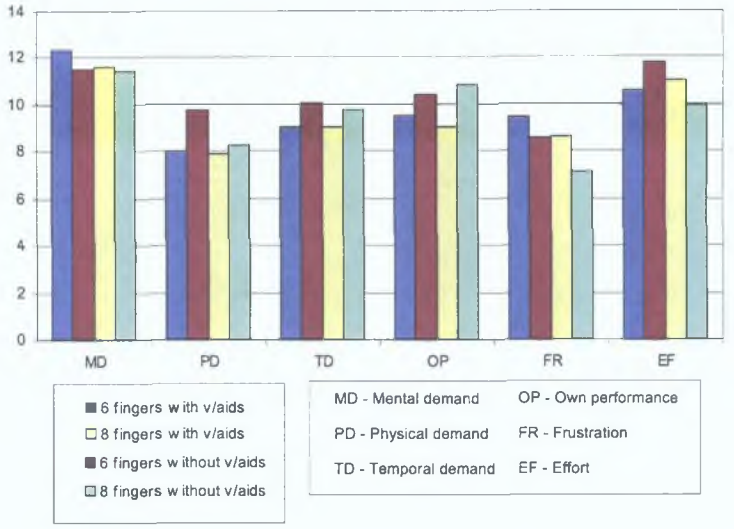


Figure 7.5: Subjective workload varying keyboard size and the use of visual aids

clusive, essentially mirroring user preference and observed WPM. No significant difference in workload was perceived as participants switched between six- and eight-fingered keyboards.

Experiment Results: Discussion

Keyboard layout The most significant result from the experiments was the effect of keyboard layout on text-entry speed. Although performance using an alphabetic keyboard was not significantly better then QWERTY, it was significantly quicker than optimised and random keyboards, while QWERTY was not. The fact that QWERTY was not fastest is a significant result in itself. Although this contradicts research by Norman and Fisher (1982) on standard keyboards, it is in keeping with more recent research by Smith and Zhai (2001) on soft keyboards. All participants in the experiments were computer science postgraduate students, and were thus quite familiar with the QWERTY keyboard layout. However, the movement required to select keys seems to have proved foreign enough to eliminate any muscle memory which may have developed while typing on standard keyboards. This resulted in the need to search for each key, on what

essentially became an almost random keyboard The alphabetic layout, with its structured layout, proved optimal in terms of recorded WPM, but was also significantly preferred over all other keyboards tested The recent explosion of the SMS text messaging phenomenon in Europe may partly explain this Users send text messages on an alphabetic layout, similar to that used in our system, are thus somewhat familiar with the alphabetic layout presented This, combined with the higher accuracy offered by the alphabetic keyboard, make it an attractive alternative to the QWERTY layout

Interestingly, several users commented that they felt the structured nature of the alphabetic keyboard actually reduced ease of learning With an unstructured layout, users made a conscious effort to learn the location of keys as they found and typed them However, with the alphabetic keyboard, this seemed not to occur, users simply mentally recited the alphabet if they couldn't find a letter, pressed it, and continued Thus, although an alphabetic keyboard may be initially advantageous for the beginner, a QWERTY may prove more beneficial for more continued use

Key count The lack of any significant effect of reducing key count, from eight to six columns, can most likely be attributed to the accurate prediction of the language model, and the levelling effects of the selection technique As discussed in Chapter 4, the effect of the language modelling, combined with a selection list of 5 words, is to decrease the influence of keyboard layout on prediction accuracy Thus, the potential increase in ambiguity caused by reducing the key count does not have a significant effect on the accuracy of word prediction

Chapter 5 discussed the use of Euclidean distance to reduce the effect of sympathetic bending However, users with strong sympathetic bending will still cause occasional often frustrating errors One solution to this problem is to train the system to recognise the individual characteristics and idiosyncrasies specific to each user However, this requires training time, which may not be acceptable

in all situations Our results show that a more attractive option is to simply reduce the key count Users with strong sympathetic bending can use reduced keyboards, without any significant effect of typing speed

Word selection technique Although not specifically tested in our experiments, it became clear from observations that the selection technique employed had a significant effect on the use of the system

The selection technique employed, which was preferred by users in previous formative experiments, was a direct mapping between the roll angle of the wrist and the highlighted word The list was ordered according to likelihood, with the most likely word placed at the top, mapping to the prone position, and alternate words selected by rolling the hand inward This technique allows users to both type, and select the most likely word, with their hands in the most ergonomically correct and comfortable typing position However, our tests revealed that despite the fact that pronation (the inward rotation of the wrists) was not necessary, and was in fact less comfortable, users preferred this position, as it reflected the typing position they were familiar with The effect of this however, was that the angle of the wrist corresponded to the third word in the list rather then the first As a consequence, users were forced to rotate their hand back to the prone position in order to select the most likely word

As a result, we feel that the list order should be altered, optimising them for selection speed according to likely usage By placing the most likely word in the middle of the list, and highlighting it by default, selection speed should increase Re-ordering the list in this manner is particularly advantageous for the *click* selection techniques as the second and third most likely words can be selected with just one *click* (Figure 7 6) Similarly, the the fifth word can be selected with two *clicks* rather then four Naturally, re-ordering the list in this manner impacts upon the recommended list length for tilt selection Recall from Table 4 6 (Page 101) that list lengths greater then two were sub-optimal for iterative

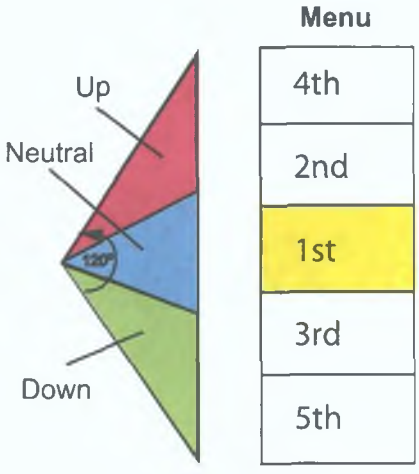


Figure 7.6: Proposed word selection technique

Keyboard	Completion list length		
	1	3	5
QWERTY	16.74	18.42	16.83
Alphabetic	17.09	18.97	17.48
Optimised	17.71	19.81	18.49

Table 7.3: Percentage of characters saved with revised, re-ordered list with iterative selection.

click selection techniques. However, when lists are re-ordered in the proposed method, lists length of three become optimum (Table 7.3).

Visual Aids Although liked by participants, visual aids had no effect on typing speed. The need to measure WPM to assess the reduction of selection errors is due to the problem of uniquely identifying the selection errors as distinct from gesture recognition errors. Measuring WPM as a measure of accuracy is based on the assumption that reduced selection errors, with a corresponding reduction in the time to delete incorrect selections and select the correct key, would have a positive effect on overall WPM. The assumption is perhaps a weak one. However, the effectiveness of the visual aids, or lack thereof, might partly be explained by one participant's description of the selection problem. They explained that having identified the correct letter, and, helped by the visual aids, having deter-

mined the correct finger to press, they nevertheless still occasionally pressed the wrong finger They likened it to the children's game where flexing interlocked fingers became difficult if the hands were first crossed, children can see the finger they wish to bend, but often bend the finger of the wrong hand Similarly, the participant explained, the graphical representation helped identify which finger to bend, but nevertheless they still occasionally bent the wrong one Consequently, the errors may be inherent to the direct selection of columns with fingers

Word Completion As with previous experiments, word completion proved well liked, and well used On average the use of word completion saved users from typing 25 percent of the required test text Fazly (2002) proposed word completion systems could predict with higher accuracy if words were never offered more then once Fazly suggested that if users did not select a complete word from the list of potential words, it could be assumed that none of the words were correct, and they could be discarded, this frees up spaces on the list for subsequent predictions, thus improving accuracy Although not quantitatively tested through software, our own observations revealed that this would not be prudent, as users often typed groups or *runs* of letters before checking the word completion list Users regularly ignored the word completion list until they had typed sufficient characters, such that there was a high likelihood that the word would appear on the completion list

Typing speed All participants completed a short typing test on a standard keyboard at the beginning of the trail Tests revealed that there was no significant correlation between standard typing speed and that observed on our virtual keyboard Although the two fastest typists, with rates of over 50 WPM on a regular keyboard, produced the fastest typing times with the virtual keyboard, reaching 11 WPM, this trend did not hold for the majority of users Motivation is likely to have been a key factor here, as some users were keen to perform well, and

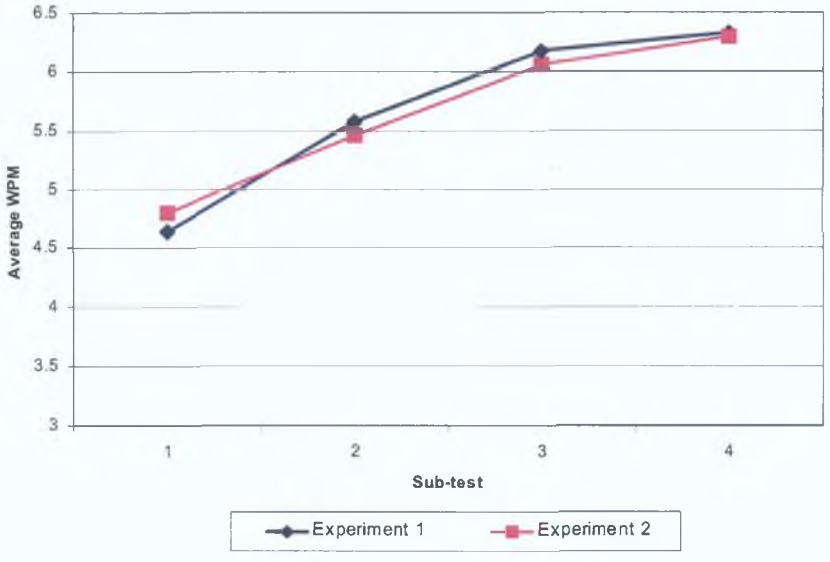


Figure 7.7: Increase in average WPM as experiments progress

reach the end of the test before the alloted time, while others were less interested.

Regardless of enthusiasm, there was a steady increase in typing speed throughout each experiment, as users became more familiar with the technique, and became more accustomed to the typing gestures needed to type accurately. In both experiment 1 and 2, this increase was clearly visible (Figure 7.7), despite the fact that in experiment 1, users were using various keyboards throughout the course of the experiment.

Subjective workload We feel the subtle changes of the NASA-TLX rating experiments may have provided more significant results. Specifically, slight variation in the presentation of the questionnaire, which allowed users to contrast ratings for other keyboards or techniques, would allow for a better comparison by participants. For our experiments, users filled out a sheet detailing rating of the subjective workload experienced (Appendix A). One page was allocated for each of the four conditions in both experiments. As a result, users were given no frame of reference with which to compare second and subsequent conditions. Some participants tried to look at previous ratings, but as they were wearing the

datagloves while filling out the subjective rating, turning pages proved difficult We believe that more consistent and significant results would have been evident if all four conditions of the experiments were rated on just one page, where users could make a better contrast between the variations in the system We feel this is an important factor, as participants are essentially performing the same task in each test, with only slight variations made in each instance As a consequence, there is a clear improvement in typing speed regardless of these changes, as shown in Figure 7 7 This clearly indicates user learning, which is likely to result in a reduced perceived workload Without a reference, indicating the workload experienced during previous sessions, users are likely to assign lower workload ratings as the experiment progresses, irrespective of the small changes made in each condition Counterbalancing will help reduce the learning effect, however, as the majority of the subjective workload is likely to be as a result of the technique itself rather then the changes made throughout the experiments The effect is that the results are somewhat diluted

7.4 Conclusions and recommendations

The design of our text-entry technique followed the iterative evolution model discussed in Section 1 4 Although computerised quantitative analysis was used to evaluate certain aspects of the system – such as the effectiveness of language models, or keyboard layout accuracy – for many aspects of the system, user evaluations were necessary

Using continuous formative evolution, greater insight is gained into the requirements, and potential problems, central to the use of the system Formative user evaluation helped identify the recognition of key-press gestures, particularly those of the little fingers, as a difficult problem crucial to the comfortable use of the system It highlighted the benefits of word completion, and provided encouraging results regarding the usability, and ease of learning of the central technique

Our larger summative evaluation provided a more detailed insight into the finer points of the system Most significant here was the effect of keyboard layout on text-entry speed Here we found a alphabetic keyboard provided the highest text-entry rate Over 50% of users also ranked the alphabetic keyboard first among those tested The tests also showed that reducing the key count from eight to six had no effect on the text-entry speed of participants Thus, six-fingered keyboards are an attractive option for users with strong sympathetic bending The use of visual aids to improve key selection accuracy was liked by many but also disliked by other users, and, as it provided no significant difference in text-entry speed, should at best be optional for any text-entry technique Finally, our tests revealed that the minor variations in the interaction technique used to select words can have significant effects on the efficiency of selection The benefits of ordered lists are compromised if user's choice of hand position results in the third most likely word being highlighted by default Thus, although the fundamental design of the interaction technique may remain fixed, minor adjustments should be possible based on user preferences

Chapter 8

Conclusions and future work

8.1 Conclusions and guidelines

This thesis presented the design, implementation and evaluation of a predictive text-entry technique for immersive environments. The technique combined datagloves, a graphically represented keyboard, and a predictive spelling paradigm to produce an effective text-entry solution which can easily be incorporated into immersive environments. Having discussed the underlying design, it identified four main factors affecting the use of such a technique: keyboard layout, prediction accuracy, gesture recognition, and interaction techniques. Chapter 3 examined optimised keyboard layouts, developing a keyboard design to minimise ambiguity while typing. Chapter 4 examined the effect of language modelling on word prediction and word completion accuracy. In Chapter 5 we examined the gesture recognition techniques suitable for identifying key-press postures. Interaction with the proposed ambiguous virtual keyboard was considered in Chapter 6. Finally, we conducted empirical experiments to examine the effects of keyboard layout and size, and the use of visual aids to improve text-entry rates, the results of which are discussed in Chapter 7. Our research resulted in a large body of results pertaining to the performance of various aspects of the system. These are now reviewed and offered as a set of recommendations for the use of predictive

text-entry in immersive environments

Gesture recognition

Central to the use of a virtual keyboard accessed with datagloves, is the accurate recognition of key-press postures Sympathetic bending and the *one size fits many* nature of the 5DT dataglove – used throughout the course of our experiments – represent the greatest problems for the design for a gesture recognition system that requires no training Misclassification errors represent the most significant problem for ambiguous text-entry, as they are the most difficult to identify by the user, false-negative and false-positive errors can be identified through aural feedback, however, misclassification errors are usually only recognised when the system fails to predict the correct word Because of this, the use of a hybrid Euclidean distance technique is recommended for the accurate recognition of key-press postures Such a technique only recognises postures once a finger is flexed past a predefined threshold, and uses Euclidean distance to determine the intended key-press when sympathetic bending causes more then one potential posture to be identified In addition, measuring finger speed is recommended to eliminate misclassification of transition errors, particularly for larger posture sets

Interaction techniques

Deconstruction of the task of ambiguous text-entry reveals two core selection tasks selection of ambiguous keys, and selection of ambiguous words In contrast to the selection of keys, which is ideally suited to a direct mapping with fingers, the ideal method for word selection is less obvious Of the selection techniques tested, both *click* techniques proved most appealing due to their accuracy In addition, the *click* techniques do not need a change of mode to select words However, during our experiments *click* techniques were the least liked by users Finally, the proposed, but untested, hybrid finger mapping technique offers an attractive alternative, as the direct selection style was preferred by users, and the

technique is not dependent on an accurate word prediction

Prediction accuracy

Higher-order language modelling offers significant improvements over the traditional uni-gram prediction typically found on most mobile phones A 30 percent reduction in errors is possible with a tri-gram model given a sufficiently large training corpus Increasing the training size of the language model, and the resulting increased dictionary has only slight negative effects on prediction accuracy for uni-gram prediction, but significantly improves higher-order models Finally, word completion offers a significant savings opportunity, savings of 30 percent can be achieved with a tri-gram language model with a training size of 25 million words Significantly, the selection technique used has an impact on the performance of word completion If iterative selection techniques, such as the *click* select method, are used, the word completion lists should be shorter, with lists of 2 or 3 optimum depending on the list ordering Finally, the greatest benefit of word completion is not the amount of letters saved, but rather that, in allowing users to quickly complete longer words, they are less likely to get *lost* due to the changeable nature key sequence interpretation Thus, particularly for beginners, word completion is recommended

Keyboard layout

The choice of keyboard layout represents perhaps the most flexible aspect of the predictive text-entry system, and is likely to be the most subjective Due to its heavy index finger bias, the universal QWERTY keyboard is not an ideal candidate for ambiguous text-entry Coupled with this, the lack of muscle memory transfer when using datagloves reduces even competent typists to the *hunt and peck* style of typing One alternative is to create a keyboard layout which minimises ambiguity while typing Our experiments revealed that optimal minimum ambiguity layouts can be created using co-occurrence bi-gram data However,

such keyboards are initially quite foreign, and the benefits afforded by their layout are significantly reduced by higher-order language modelling and the use of increased prediction list lengths with direct selection techniques Their use seems unlikely to be warranted unless a uni-gram language model is used for prediction An alphabetic layout, in contrast, offers considerable appealing features Its uniform letter allocation results in a higher prediction accuracy then QWERTY and it is easier to search than seemingly random alternatives such as QWERTY and optimised keyboards Our summative evaluation revealed that, on average, an alphabetic keyboard resulted in the fastest text-entry times However, both of the fastest individual recorded times were on QWERTY keyboards Finally, in cases of strong sympathetic bending, six finger keyboard layouts are recommend as they displayed no negative impact on typing speed Ultimately the decision should be left to the user Beginners are likely to prefer and preform better with an alphabetic layout, and we believe this should be offered by default However, experienced typists may dislike such a layout Accordingly, the QWERTY should be available as an alternative Only users expecting to reach expert level, who are prepared to learn a new foreign layout, are likely to experience any benefit from an optimised keyboard

8 2 Future work

Like many research projects, the work carried out in this thesis has raised more questions then it has answered This thesis has focussed on the problems which we believe are central to the effective use of predictive text-entry in immersive environments prediction accuracy, gesture recognition, keyboard layout, and interaction techniques In researching all four topics we have sacrificed depth for breadth, and any of the four core areas could be the subject of considerable further research

Gesture recognition In exploring gesture recognition, we believe that the use of language modelling statistics would offer valuable information and would prove a useful augmentation The use of bi-gram statistics to identify typing errors is certainly not a new concept, and was proposed over 20 years ago by Ullmann (1977) Currently, applications such as Microsoft Word detect typing errors after keyboard keys are struck Common transposition errors – e g typing *hte* instead of *the* – are corrected automatically without any user intervention This technique could equally be applied to gesture recognition Sympathetic bending can lead to uncertainly as to which key was intended by the user However, bi-gram information could be used as a parameter in such situations were a similar Euclidean distance to two possible gestures indicated that both were likely The effect of such corrections would be a reduction in misclassification errors, which, as discussed previously, are the most difficult for the user to identify

Interaction techniques In discussing interaction techniques we have focussed solely on the core task of text-entry However, to be truly useful, more complex symbolic input would be necessary, as would the entry of unambiguous words Mode changes, increased layouts, greater DOF would all provide rich grounds for future research Our own research focussed on the techniques possible with the 5DT dataglove Because of its limited tracking abilities, more advanced interaction – such as editing previously typed words or adding markup such as bold or italics – is unlikely to be possible without convoluted, unintuitive interaction techniques A richer set of interaction techniques would be achievable if 3D spatial trackers were employed

Prediction accuracy Examining prediction accuracy, we have focussed solely on language modelling techniques Language modelling was used as it is ideally suited to the relatively uninflected English language, and is already used in a limited form in modern mobile phones However, as discussed in Chapter 2, a variety

of alternative techniques exist Most likely, a combination of techniques would provide the most fruitful results The difficulty in marrying several techniques is in determining the weight each should be assigned when ranking predictions It is entirely conceivable that without a well weighted hierarchy, hybrid techniques might preform worse then language modelling alone Nevertheless, as experiments by Lesher *et al* (2002) indicate, when aided by computers, humans can predict 10 percent better than statistic based systems alone, as they combine several factors into their predictions Thus, hybrid techniques, which combine knowledge from various sources in a fashion similar to humans, warrant further research

Keyboard layout In examining keyboard layouts, we believe that alphabetic keyboards proved superior because they facilitated easy searching During our own user tests, users often expressed the belief that one positive aspect of the random keyboard was the perceived word connectivity Although the keyboard was designed at random, they felt that it had properties which simplified searching Zhai *et al* (2002) designed keyboards designed to increase word connectivity, however, no experiments were carried out to evaluate the performance gains possible Based on user feedback, we believe that the design of such keyboards warrants further research

Final thoughts Finally, although our text-entry technique has attempted to bring the familiar *static* keyboard into VR, alternative gestural techniques – designed for small devices – such as Dasher (Ward *et al* , 2000) and Hex (Williamson and Murray-Smith, 2003) question the need for such an approach These techniques, which use linguistic information to dynamically alter the keyboard during typing, would take on another dimension in 3D and would provide fascinating research material if combined with spatially tracked gloves

Bibliography

Bahuman, A , Ramakrishnan, C , Ashish, D , Crouch, D , and Hasan, S (2000) Optimizing ambiguous keyboard layout the for the English language via genetic algorithm

Beck, C , Seisenbacher, G , Edelmayer, G , and Zagler, W (2004) First user test results with the predictive typing system FASTY, Submitted to ICCHP 2004

Bier, E A (1987) Skitters and jacks interactive 3D positioning tools In *Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 183–196 ACM Press

BNC (1995) British National Corpus, available from http //www natcorp ox ac uk/

Boissière, P and Vigouroux, N (2003) An Overview of Existing Writing Assistance Systems, French-Spanish Workshop on Assistive Technology (October 16th 2003), Paris

Bolt, R A (1980) Put-That-There Voice and gesture at the graphics interface In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270 ACM Press

Bowman, D (1999) *Interaction Techniques For Common Tasks In Immersive Virtual Environments Design, Evaluation, and Application* PhD thesis, Computer Science Dept, Georgia Institute of Technology

Bowman, D , Kruijff, E , LaViola, J , and Poupyrev, I (2004) *3D User Interfaces Theory and Practice* Addison Wesley, Boston, USA

Bowman, D , Ly, V , and Campbell, C (2001a) Pinch Keyboard Natural Text Input for Immersive Virtual Environments Technical report, Virginia Tech Dept of Computer Science

Bowman, D , Rhoton, C , and Pinho, M (2002) Text Input Techniques for Immersive Virtual Environments an Empirical Comparison In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 2154–2158

Bowman, D , Wingrave, C , Campbell, J , and Ly, V (2001b) Using Pinch Gloves for both Natural and Abstract Interaction Techniques in Virtual Environments In *Proceedings of HCI International*, pages 629–633

Bowman, D A , Johnson, D B , and Hodges, L F (1999) Testbed evaluation of virtual environment interaction techniques In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 26–33 ACM Press

Bowman, D A and Wingrave, C A (2001) Design and Evaluation of Menu Systems for Immersive Virtual Environments In *Proceedings of the Virtual Reality 2001 Conference (VR'01)*, page 149 IEEE Computer Society

Brown, M (1992) Why a QWERTY keyboard? Electronic resource Available from Userlab Inc website http //www userlab com/Downloads/QWERTY PDF

Card, S K , Newell, A , and Moran, T P (1983) *The Psychology of Human-Computer Interaction* Lawrence Erlbaum Associates, Inc , Hillsdale, NJ

Cardinal, J and Langerman, S (2004) Designing small keyboards is hard In *In Proc Latin American Theoretical INformatics (LATIN), Springer Lecture Notes in Computer Science, volume 2976*, pages 391–400

D L Grover, M T K and Kuschler, C A (1998) Reduced keyboard disambiguating computer United States Patent 5,818,437 Tegic Communications, Inc , Seattle, WA

Dagan, I , Lee, L , and Pereira, F (1999) Similarity-Based Models of Cooccurrence Probabilities *Machine Learning*, 34(1-3) 43–69

Darragh, J (1989) Adaptive Predictive Text Generation And The Reactive Keyboard Technical report, Department of Computer Science, University of Calgary Electronic Copy [Accessed 6 June 2002]

Defanti, T and Sandin, D (1977) Final Report to the National Endowment of the Arts Technical Report US NEA R60-34-163, University of Illinois at Chicago Circle, Chicago, Illinois

Dunlop, M D and Crossan, A (2000) Predictive text entry methods for mobile phones *Personal Technologies*, 4(2) Electronic Copy [Accessed 01 July 2002]

Essential Reality (2003) p5 glove Essential Reality website, http //www p5 com/

Evans, F , Skiena, S , and Varshney, A (1999) VType Entering Text in a Virtual World submitted to International Journal of Human-Computer Studies

Evreinova, T , Evreinov, G , and Raisamo, R (2004) Four-key Text Entry for Physically Challenged People In *8th ERCIM Workshop User Interfaces for All"*

Fakespace Inc (2004) Fakespace Inc website, http //www fakespacelabs com/

Fazly, A (2002) *The use of syntax in word completion utilities* PhD thesis, University of Toronto, Canada

Fazly, A and Hirst, G (2003) Testing the efficicy of part-of-speech information

in word completion In *Proceedings of the Workshop on Language modeling for text entry methods*, pages 6–9, Budapest, Hungary

Fels, S and Hinton, G (1993) Glove-Talk A Neural Network Interface Between a Data-Glove and a Speech Synthesizer *IEEE Transactions on Neural Networks*, 4(1) 2–8

Fifth Dimension Technologies (2004) 5DT website, http //www 5dt com/

Fitts, P M (1954) The information capacity of the human motor system in controlling the amplitude of movement *Journal of Experimental Psychology*, 47 381–391

Fitts, P M and Peterson, J R (1964) Information capacity of discrete motor responses *Journal of Experimental Psychology*, 67 103–112

Foulds, R A , Soede, M , and van Balkom, H (1987) Statistical disambiguation of multi-character keys applied to reduce motor requiremens for augmentitive and alternative communication *Augmentative and alternative communication, Vol 3, pp 192-195*

Freeman, W T and Weissman, C D (1994) Television Control by Hand Gestures Technical Report TR94-24, Mitsubishi Electric Research Labs IEEE Intl Wkshp on Automatic Face and Gesture Recognition, Zurich, June, 1995

FrogPad (1999) FrogPad(tm) website http //www frogpad com/

Gabbard, J L , Hix, D , and Edward Swan II, J (1999) User-Centered Design and Evaluation of Virtual Environments *IEEE Computer Graphics and Applications, Invited paper*, 19(6) 51–59

Garay-Vitoria, N and Gonzalez-Abascal, J (1997) Intelligent word-prediction to enhance text input rate (a syntactic analysis-based word-prediction aid for people with severe motor and speech disability) In *Proceedings of the 2nd*

international conference on Intelligent user interfaces, pages 241–244 ACM Press

Garbe, J (2000) *Optimizing the layout of an ambiguous keyboard using a genetic algorithm* PhD thesis, University of Koblenz

Gazdar, G (1996) *Paradigm merger in natural language processing*, pages 88–109 Cambridge University Press

GKOS (2000) The Global Keyboard Optimised for Small Wireless Terminals Available from GKOS website http //www gkos com

Glaser, R E (1981) A telephone communication device for the deaf In *Proceedings of the Johns Hopkins First National Search for Applications of Personal Computers to Aid the Handicapped*, pages 11–15, Piscataway, NJ IEEE Service Center

Goldstein, M and Chincolle, D (1999) The Finger-Joint Gesture Wearable Keypad in Second Workshop on Human Computing Interaction with Mobile Devices

Good, I J (1953) The population frequencies of species and the estimation of population parameters *Biometrika*, 40(3) 237–264

Grimes, G J (1983) Digital data entry glove interface device United States Patent 4,414,537 November 8 Bell Telephone Laboratories, Murray Hill, NJ

Guenthner, F , Langer, S , Kruger-Thielmann, K , Richardet, N , Sabatier, P , and Pasero, R (1993) KOMBE Communication Aids for the Handicapped

Gutenburg, P (1971) Online collection of ebooks Available from http //www gutenberg net/

Hand, C (1997) A Survey of 3D Interaction Techniques *Computer Graphics Forum*, 16(5) 269–281

Hansen, J P , Johansen, A S , Hansen, D W , Itoh, K , and Mashino, S (2003) Language Technology in a Predictive, Restricted On-Screen Keyboard with Dynamic Layout for Severely Disabled People In *EACL workshop on Language modeling for text entry methods*, Budapest, Hungary

Harbusch, K and Kuhn, M (2003) Towards an Adaptive Communication Aid with Text Input from Ambiguous Keyboards In *Proceedings of EACL 2003*, Budapest

Harling, P (1993) Gesture Input Using Neural Networks Department of Computer Science, University of York, York, YO1 5DD, UK

Hart, S G and Staveland, L E (1988) Development of NASA-TLX (Task Loading Index) Results of Empirical and Theoretical Research In Hancock, P A and Meshkati, N , editors, *Human Mental Workload*, pages 139–183 North Holland Press, Amsterdam

Hasan, S (2003) *N-Best Hidden Markov Model Supertagging for Typing with Ambiguous Keyboards* PhD thesis, The University of Koblenz and Landau

Hawes, N and Kelleher, J (2004) Context-Sensitive Word Selection For Single-Tap Text Entry Presented at the Stairs Track at the Europen Conference on Artifical Intelligence (ECAI'04) August 22–27, Valencia, Spain

Hughes, D , Warren, J , and Buyukkokten, O (2002) Empirical Bi-Action Tables A Tool for the Evaluation and Optimization of Text-Input Systems Application I Stylus Keyboards *Human-Computer Interaction*, 17 271–309

Immersion Corporation (2004) Immersion Corporation website, http //www immersion com/

Jacob, R (2000) User Interfaces In Hemmendinger, D , Ralston, A , and Reilly, E , editors, *Encyclopedia of Computer Science, Fourth Edition* Grove Dictionaries Inc Electronic Copy [Accessed August 27, 2003]

Jean Laurent, D P (1997) Introduction to Pattern Recognition Haitian Scientific Society Seminars Series Available from http //www math umb edu/hss/seminars/1997/pj/

Johnson, A B and Hagstad, R F (1981) DTMF telecommunications for the deaf and speech impaired In *Proceedings of the Johns Hopkins First National Search for Applications of Personal Computers to Aid the Handicapped*, pages 29–32, Piscataway, NJ IEEE Service Center

Johnson, P (1992) *Human-Computer Interaction Psychology, Task Analysis and Software Engineering* McGraw-Hill, Maidenhead, UK

Jurafsky, D and Martin, J H (2000) *Speech and Language Processing* Prentice-Hall, Inc, Upper Saddle River, New Jersey

Kadous, M (1995) GRASP Recognition of Australian sign language using instrumented gloves Honours thesis, School of Computer Science and Engineering, University Electronic resource Available from http //citeseer ist psu edu/kadous95grasp html

Katz, S M (1987) Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3) 400–401

Keenan, F (2002) PCs and Speech A Rocky Marriage

Kelleher, J (2003) *A Perceptually Based Computational Framework for the Interpretation of Spatial Language* PhD thesis, School of Computing, Dublin City University

Kessler, D G , Hodges, L F , and Walker, N (1995) Evaluation of the CyberGlove as a Whole-Hand Input Device *ACM Transactions on Computer–Human Interaction*, 2(2) 263–283

Kettner, L (1995) A Classification Scheme of 3D Interaction Techniques Technical report B 95-05 Technical report, Institute for Computer Science, Department of Mathematics and Computer Science, Freie Universitat Berlin, Germany

Kinkead, R (1975) Typing speed, keying rates, and optimal keyboard layouts In *Proceedings of the 19th Annual Meeting of the Human Factors Society*, pages 159–161, Santa Monica, CA Human Factors Society

Klarlund, N and Riley, M (2003) Word n-grams for cluster keyboards In *Proceedings of the Workshop on Language modeling for text entry methods*, pages 6–9, Budapest, Hungary

Kramer, J and Leifer, L (1989) The Talking Glove A speaking aid for non-vocal deaf and deaf-blind individuals In *Proceedings of RESNA 12th Annual Conference*, pages 471–472, Toronto, Ontario, Canada

Kramer, J P , Lindener, P , and George, W R (1991) Communications system for deaf, deaf-blind, or non-vocal individuals using instrumented glove United States Patent 5,047,952 September 17

Kuhn, M and Garbe, J (2001) Predictive and highly ambiguous typing for a severely speech and motion impaired user In *Proceedings of Universal Access in Human-Computer Interaction* , pages 933–936, Mahwah, New Jersey Lawrence Erlbaum Associates

LaViola, J J (1999) Whole-Hand and Speech Input in Virtual Environments Technical Report CS-99-15

Lesher, G and Moulton, B (2000) A method for optimizing single-finger keyboards In *Proceedings of the RESNA 2000 Annual Conference*, pages 91–93 RESNA Press

Lesher, G , Moulton, B , and Higginbotham, D (1999) Effects of ngram order and training text size on word prediction In *Proceedings of the RESNA '99 Annual Conference*, pages 52–54, Arlington, VA RESNA Press

Lesher, G , Moulton, B J , and Higginbotham, D J (1998) Optimal character arrangements for ambiguous keyboards

Lesher, G , Moulton, B J , Higginbotham, D J , and Alsofrom, B (2002) Limits Of Human Word Prediction Performance In *In Proceedings of the Seventeeth Annual International Conference on Technology And Persons With Disabilities*

Levine, S H (1985) An adpative approach to optimal keybpard design for nonvocal communication In *Proceedings of the international conference on cybernetics and society*, pages 334–337

Levine, S H and Goodenough-Trepagmer, C (1990) Customised text entry devices for motor-impaired users *Applied Ergonomics, Vol 21, pp 55-62*

Levine, S H , Minneman, S L , Getshow, C O , and Goodenough-Trepagnier, C (1986) Computer disambiguation of multi-character key text entry An adaptive design apporach In *Proceedings of the international conference on systems, man and cybernetics*, pages 298–301

Lewis, J (1999) Development of a digram-based typing key layout for single-finger/stylus input In *Proceedings of the Human Factors and Ergonomics Society 43rd Annual meeting*, pages 415–419, Houston, Texas

Lewis, J , Potosnak, K , and Magyar, R (1997) *Keys and keyboards*, pages 1285–1315 North-Holland (Elsevier), Amsterdam

Liang, R -H and Ouhyoung, M (1996) A Sign Language Recognition System Using Hidden Markov Model and Context Sensitive Search In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology'96*, pages 59–66 Electronic Copy [Accessed June 15, 2004]

Light, L W and Anderson, P G (1993) Typewriter Keyboards via Simulated Annealing

MacKenzie, I , Zhang, S , and Soukoreff, R (1999) Text entry using soft keyboards *Behaviour and Information Technology*, 18 235–244

MacKenzie, I S (1995) *Movement time prediction in human-computer interfaces* , pages 483–493 Kaufmann Los Altos, CA

MacKenzie, I S (2002) Research Note Within-subjects vs Between-subjects Designs Which to Use?

MacKenzie, I S and Soukoreff, R W (2003) Phrase sets for evaluating text entry techniques In *CHI '03 extended abstracts on Human factors in computing systems*, pages 754–755 ACM Press

MacKenzie, I S and Zhang, S Z (1999) The design and evaluation of a high performance soft keyboard In *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '99*, pages 25–31, New York Electronic Copy [Accessed 29 April 2002]

MacKenzie, S , Kober, H , Smith, D , Jones, T , and Skepner, E (2001) LetterWise Prefix-based disambiguation for mobile text input In *Proceedings of the ACM Symposium on User Interface Software and Technology - UIST 2001*, pages 111–120, New York ACM Press

Mankoff, J and Abowd, G D (1998) Cirrin A word-level unistroke keyboard for pen input In *ACM Symposium on User Interface Software and Technology*, pages 213–214

Matias, E , MacKenzie, I , and Buxton, W (1994) Half-QWERTY A one-handed keyboard facilitating skill transfer from QWERTY In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, pages 88–94, New York ACM Press

Mehring, C , Kuester, F , Singh, K D , and Chen, M (2004) Kitty Keyboard independent touch typing in VR Poster presented at IEEE Virtual Reality conference

Mine, M (1995) Virtual environment interaction techniques Technical Report TR95-018, UNC Chapel Hill Computer Science Dept

Mine, M (1996) Working in a Virtual World Interaction Techniques Used in the Chapel Hill Immersive Modeling Program Technical Report TR96-029, UNC Chapel Hill Computer Science Dept

Mine, M R , Brooks, Jr , F P , and Sequin, C H (1997) Moving Objects in Space Exploiting Proprioception In Virtual-Environment Interaction *Computer Graphics*, 31(Annual Conference Series) 19–26

Newby, G B (1993) Gesture Recognition Using Statistical Similarity In *Proceedings of the conference Virtual Reality and Persons with Disabilities* Electronic Copy [Accessed August 1, 2003]

Norman, D and Fisher, D (1982) Why alphabetic keyboards are not easy to use Keyboard layout doesn't much matter *Human Factors*, 24 509–519

Noyes, J (1983) The QWERTY keyboard A review *International Journal of Man-Machine Studies*, 18 265–281

Oommen, B J , Valiveti, R S , and Zgierski, J R (1991) An adaptive learning solution to the keyboard optimization problem *IEEE transactions on systems, man and cybernetics Vol 21, No 6*

Osawa, N and Sugimoto, Y Y (2002) Multimodal Text Input in an Immersive Environment In *The 12th International Conference on Artificial Reality and Telexistence (ICAT 2002)*, pages 85–92

PalmOne Inc (2004) Graffiti PalmOne website, http //www palmone com/us/products/input/graffiti2 html

Perlin, K (1998) Quikwriting Continuous Stylus-Based Text Entry In *ACM Symposium on User Interface Software and Technology*, pages 215–216

Pierce, J S , Forsberg, A S , Conway, M J , Hong, S , Zeleznik, R C , and Mine, M R (1997) Image plane interaction techniques in 3D immersive environments In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 39–ff ACM Press

Poupyrev, I , Billinghurst, M , Weghorst, S , and Ichikawa, T (1996) Go-Go Interaction Technique Non-Linear Mapping for Direct Manipulation in VR In *Proceedings of UIST'96*, pages 79–80

Poupyrev, I , Tomokazu, N , and Weghorst, S (1998a) Virtual Notepad handwriting in immersive VR In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, pages 126–132

Poupyrev, I , Weghorst, S , Bilhnghurst, M , and Ichikawa, T (1998b) Egocentric Object Manipulation in Virtual Environments Empirical Evaluation of Interaction Techniques *Computer Graphics Forum*, 17(3)

Pratt, V R (1998) Thumbcode A Device-Independent Digital Sign Language Electronic resource Available from http //boole stanford edu/pub/thumbcode ps gz [Accessed 3 July 2002]

Preece, J , Benyon, D , and University, O (1993) *A Guide to Usability Human Factors in Computing* Addison-Wesley Longman Publishing Co , Inc

Rosenberg, R (1998) *Computing without Mice and Keyboards Text and Graphic Input Devices for Mobile Computing* Doctoral dissertation, Dept of Computer Science, University College, London

Rua, H and Skiena, S S (1994) Dialing for documents an experiment in information theory In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 147–155 ACM Press

Sandberg, A (1997) Gesture recognition using neural networks Dept Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm (TRITA-NA-E9727)

Schneiderman, B (2000) The Limits of Speech Recognition *Communications Of The ACM*, 43(9) 63–65 Electronic Copy [Accessed 6 June 2002]

Shamaie, A and Sutherland, A (2003) Accurate Recognition of Large Number of Hand Gestures In *Second Iranian Conference on Machine Vision and Image Processing*, K N Toosi University of Technology, Tehran

Shannon, C E (1951) Prediction and entropy of printed English In Sloane, N and Wyner, A D , editors, *Claude Elwood Shannon Collected Papers*, pages 194–208 IEEE Press, Piscataway NJ, USA

Shaw, C and Green, M (1994) Two-handed Polygonal Surface Design In *ACM Symposium on User Interface Software and Technology*, pages 205–212

Silfverberg, M , MacKenzie, I S , and Korhonen, P (2000) Predicting text entry speed on mobile phones In *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2000*, pages 9–16 ACM Press

Smith, B and Zhai, S (2001) Optimised Virtual Keyboards with and without Alphabetical Ordering - A Novice User Study In *Proc of INTERACT 2001 Eighth IFIP Conference on Human-Computer Interaction*, pages 92–99, Tokyo, Japan

Soukoreff, R and MacKenzie, I (1995) Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard *Behaviour and Information Technology*, 14 370–379

Stanford Research Institute (1995) SRILM - The SRI Language Modeling Toolkit

Starner, T and Pentland, A (1995) Visual Recognition of American Sign Language Using Hidden Markov Models In *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland

Stocky, T , Faaborg, A , and Lieberman, H (2004) A commonsense approach to predictive text entry In *Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 1163–1166 ACM Press

Sturman, D J (1992) *Whole-hand input* PhD thesis, Massachusetts Institute of Technology

Sturman, D J and Zeltzer, D (1994) A survey of glove–based input *IEEE Computer Graphics and Applications*, 14(1) 30–39

T9 (2004) Tegic Communications web page Available from http //www t9 com/

Textware Solutions (1998) Fitaly Keyboard From Textware Solutions website, http //www fitaly com Available from http //www fitaly com

Thomas, B , Tyerman, S , and Grimmer, K (1997) Evaluation of Three Input Mechanisms for Wearable Computers In *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, page 2 IEEE Computer Society

TwiddlerII (2004) Handykey Web Site Available from http //www handykey com/site/twiddler2 html

Ullmann, J (1977) A binary n-gram technique for automatic correction of substitution, deletion, insertion, and reversal errors in words *Comput J*, 20(2) 141–147

Van Laarhoven, P and Aarts, E (1987) *Simulated Annealing Theory and Applications* Kluwer Academic Publishers, Dordrrecht, The Netherlands

Ward, D , Blackwell, A , and D , M (2000) Dasher - a Data Entry Interface Using Continuous Gestures and Language Models In *UIST 2000 The 13th Annual ACM Symposium on User Interface Software and Technology*, pages 129–137, San Diego, CA, USA Electronic Copy [Accessed 2 August 2002]

Ward, D J (2001) *Adaptive computer interfaces* PhD thesis, University of Cambridge

Watson, R (1993) A Survey of Gesture Recognition Techniques Technical Report TCD-CS-93-11

Williamson, J and Murray-Smith, R (2003) Dyanmics and Probabilistic Text Entry Technical report, TR-2003-147 Dept Computing Science, University of Glasgow Electronic Copy [Accessed August 1, 2003]

Witten, I and Bell, T (1991) The zero-frequency problem Estimating the probabilities of novel events in adaptive text compression *IEEE Transations on Information Theroy* , 34(4) 1085–1094

Witten, I H (1982) *Principles of Computer Speech* Academic Press, Boston

Wu, H and Sutherland, A (2001) Dynamic gesture recognition using PCA with multiscale theory and HMM In *Proc SPIE Vol 4550, Image Extraction, Segmentation, and Recognition, Tianxu Zhang, Bir Bhanu, Ning Shu, Eds* , pages 132–1–39

Zatsiorky, V , Li, Z -M , and Latash, M (2000) Enslaving effects in multi-finger force production *Experimental Brain Research*, 131(2) 187–195

Zhai, S , Hunter, M , and Smith, B (2000) The Metropolis Keyboard – An Exploration of Quantitative Techniques for Virtual Keyboard Design In *Proceeding of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST 2000)*, pages 119–128, San Diego, California

Zhai, S , Hunter, M , and Smith, B (2002) Performance Optimization of Virtual Keyboards *Human-Computer Interaction*, 17(2,3) 89 Electronic Copy [Accessed 10 August 2003]

Zimmerman, T G (1985) Optical flex sensor United States Patent 4,542,291 September 17 VPL Research Inc , Palo Alto, CA

Zimmermann, T , Lanier, J , Blanchard, C , Bryson, S , and Young, H (1987) A hand gesture interface device In *Proceedings of CHI+GI'87 Human Factors in Computing systems and Graphics Interface*, pages 189–192, Toronto, Ontario, Canada

Zipf, G K (1935) *The Psycho-Biology of Language* Houghton Muffin, Boston

Appendix A

NASA TLX Questionaire

During summative evaluation of the system, each participant completed a questionnaire An informed consent form (Page 191) was completed before the experiment commenced Page 192 or 193 was selected where appropriate depending on the experiment Page 194 (From Hart and Staveland, 1988) was made available for reference throughout the experiment Page 195 was presented to participants after each condition Finally, Page 196 was completed after the final condition

Informed consent form.

I state that I am over 18 years of age, and wish to participate in a program of research being conducted by Barry McCaul, in the School of Computing, Dublin City University, as part of his Ph D research

The purpose of the research is to evaluate text-entry techniques within immersive VR environments The experiments consist of both automatic and visual monitoring of my interaction with the VR system, while I attempt to complete requested tasks in the environment I will also be asked to complete a brief questionnaire, designed to asses my previous VR ability and evaluate the text input techniques used

All information collected in the study is strictly confidential, and my name will not be identified at any time I understand that I am free to ask questions or withdraw from participation at any point

Signature of Participant Date

General Information

Name _____ Student Number _____

Standard keyboard typing speed (WPM) ____ Date of Birth __ / __ / 19___

Have you had any previous experience with VR, including 3D computer games?

_____

_____

Have you had any previous experience with datagloves?

_____

_____

Please rank (1 - 4) the 4 keyboards tested in order of preference

__ Qwerty
__ Opti
__ Alphabetic
__ Random

Would you like to comment further on your choices?

_____

_____

_____

General Information

Name _____ Student Number _____

Standard keyboard typing speed (WPM) ____ Date of Birth __ / __ / 19___

Have you had any previous experience with VR, including 3D computer games?

_____

_____

Have you had any previous experience with datagloves?

_____

_____

Please rank (1 - 4) the 4 methods tested in order of preference

__ 6 fingers without visual cue
__ 6 fingers with visual cue
__ 8 fingers without visual cue
__ 8 fingers with visual cue

Would you like to comment further on your choices?

_____

_____

_____

NASA TLX Rating Scale Definitions

RATING SCALE DEFINITIONS		
Title	**Endpoints**	**Descriptions**
MENTAL DEMAND	*Low/High*	How much mental and perceptual activity was required (e g thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding simple or complex, exacting or forgiving?
PHYSICAL DEMAND	*Low/High*	How much physical activity was required (e g pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
TEMPORAL DEMAND	*Low/High*	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
EFFORT	*Low/High*	How hard did you have to work (mentally and physically) to accomplish your level of performance?
PERFORMANCE	*Good/Poor*	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
FRUSTRATION LEVEL	*Low/High*	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

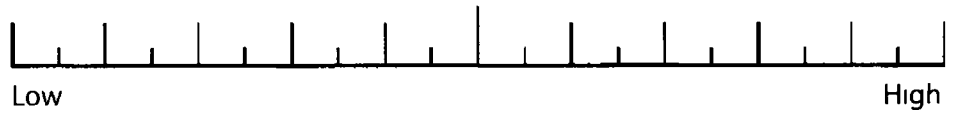
4 NASA TLX Participant Rating Form

Place a mark on each scale that represents the magnitude of each factor in the task you just preformed

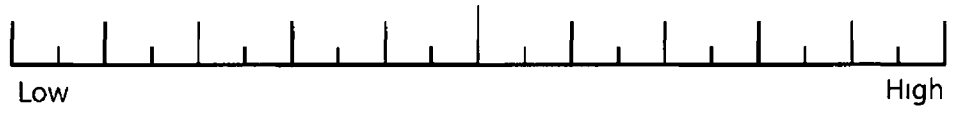
Mental Demand

Physical Demand

Temporal Demand



Effort



Performance

Frustration

Please circle the member of each pair that provided the most significant source of workload variation in the tasks

MD	Mental Demand	EF	Effort
PD	Physical Demand	OP	Performance
TD	Temporal Demand	FR	Frustration

PD / MD	TD / PD	TD / FR
TD / MD	OP / PD	TD / EF
OP / MD	FR / PD	OP / FR
FR / MD	EF / PD	OP / EF
EF / MD	TD / OP	EF / FR