TEXT ENTRY FOR MOBILE SYSTEMS:
MODELS, MEASURES, AND ANALYSES FOR TEXT ENTRY RESEARCH

A Thesis
Presented to
The Faculty of Graduate Studies
of
York University

by
ROBERT WILLIAM SOUKOREFF

In partial fulfilment of requirements for the degree of
Master of Science
June, 2002

© W. Soukoreff, 2002
Abstract

TEXT ENTRY FOR MOBILE SYSTEMS:
MODELS, MEASURES, AND ANALYSES FOR TEXT ENTRY RESEARCH

Robert William Soukoreff
York University, June 2002

A theoretical model is developed that predicts upper-bound and lower-bound text-entry rates using a stylus to tap on a soft Qwerty keyboard. The model is based on the Hick-Hyman law for choice reaction time, Fitts' law for rapid aimed movements, and linguistic tables for the relative frequencies of letter-pairs (digrams) in common English. The model's importance lies not only in the predictions provided, but in its characterization of text-entry tasks using keyboards, and in the model's utility in optimising soft keyboards. An important contribution is the construction of a complete table of digram frequencies including the space character.

The accuracy of the model is evaluated in an experimental context.

A review of the interest in soft keyboard optimisation generated by the publication of the model is presented.

A second model is presented that predicts upper-bound text entry rates for two-thumb typing on a miniature handheld Qwerty keyboard. This model is of interest because it presents a deconstruction of the two-thumb text entry task, and because it makes a priori evaluation of miniature keyboards possible. Important contributions include the construction of two Fitts' law models describing the motion of the thumbs during two-thumb typing, and measurement and modelling of key repeat and thumb alternation times.
Acknowledgements

I would like to express my profound thanks to my supervisor, Dr. Scott MacKenzie, for years of instruction, patient mentoring, and friendship. Over the years we have studied together, published together, presented together, and occasionally taken time out to party together. I have had a great time completing my M.Sc. with Dr. MacKenzie’s supervision, and I am happy to be continuing for my Ph.D. under him as well. Oh yeah, and “Proooobe!”

I would also like to thank my fellow student, Shawn Zhang, for support and collaboration during this research. Part of Chapter 3 came from a paper co-authored by Mr. Zhang, Dr. MacKenzie, and myself.

Additional thanks goes to the members of my thesis committee, to the graduate program director, Dr. Minas Spetsakis, and to the graduate program assistant, Lisa Guidi, for helping me to navigate the bureaucratic waters here at the University.

I also thank my family and friends for years of encouragement. I am sure that no-one will be happier or prouder to see me complete this course of study than my parents. Hi Mom, Dad!

Finally, I would like to thank Lena Serghides for saying “Yes!”.
Dissemination of this Thesis

The following chapters or parts thereof have been published in journals or in conference proceedings as peer reviewed papers.

Chapter 1: Introduction


Chapter 2: Modelling Stylus Typing

Chapter 3: Comparing the Soft Keyboard Model with Empirical Data

Chapter 4: Optimisation of Soft Keyboards for Stylus Typing

Chapter 5: A Model of Two-Thumb Text Entry
## Table of Contents

Abstract ........................................................................................................................................ iv  
Acknowledgements .................................................................................................................... v  
Dissemination of this Thesis ....................................................................................................... vi 
Table of Contents .................................................................................................................... vii  
List of Tables................................................................................................................................ xi  
List of Figures ........................................................................................................................ xii  
List of Equations ..................................................................................................................... xv  

Chapter 1  Introduction ............................................................................................................. 1  
  1.1 Overview .......................................................................................................................... 2  
  1.2 Background ...................................................................................................................... 4  
  1.2.1 Mobile devices ........................................................................................................... 4  
  1.2.2 Text entry .................................................................................................................. 7  
  1.3 Evaluation ....................................................................................................................... 9  
  1.3.1 Methodology ............................................................................................................. 9  
  1.3.2 Text copy tasks versus text creation tasks ................................................................. 10  
  1.3.3 Novice versus expert performance ............................................................................. 12  
  1.3.4 Quantitative versus qualitative analyses .................................................................. 14  
  1.3.5 Speed ......................................................................................................................... 15  
  1.3.6 Accuracy ................................................................................................................... 15  
  1.3.7 Other factors ............................................................................................................ 19  
  1.4 Optimisation techniques ................................................................................................. 20  
  1.4.1 Movement-minimisation ......................................................................................... 20  
  1.4.2 Language prediction ................................................................................................. 21  
  1.4.2.1 Corpus not representative of the user language .................................................. 22  
  1.4.2.2 Corpus ignores the editing process .................................................................... 22  
  1.4.2.3 Corpus does not capture input modalities ......................................................... 23  
  1.4.3 Hybrid input techniques ......................................................................................... 24  
  1.4.4 Key minimization techniques (modes) .................................................................. 24  
  1.5 Survey of text entry techniques ...................................................................................... 27  
  1.5.1 Key-based text entry ................................................................................................. 27  
    1.5.1.1 Telephone keypad ........................................................................................... 27  
    1.5.1.2 Small Qwerty keyboards ............................................................................... 31  
    1.5.1.3 Three-key and five-key text entry .................................................................. 36  
    1.5.1.4 Other small keyboards .................................................................................. 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.9.1.3</td>
<td>Procedure</td>
<td>136</td>
</tr>
<tr>
<td>5.9.1.4</td>
<td>Design</td>
<td>136</td>
</tr>
<tr>
<td>5.9.1.5</td>
<td>The Fitts' law models</td>
<td>136</td>
</tr>
<tr>
<td>5.9.1.6</td>
<td>Measuring $t_{\text{REPEAT}}$</td>
<td>137</td>
</tr>
<tr>
<td>5.9.1.7</td>
<td>Measuring $t_{\text{MIN}}$</td>
<td>138</td>
</tr>
<tr>
<td>5.9.2</td>
<td>Results</td>
<td>138</td>
</tr>
<tr>
<td>5.9.2.1</td>
<td>The Fitts' law models</td>
<td>138</td>
</tr>
<tr>
<td>5.9.2.2</td>
<td>Measuring $t_{\text{REPEAT}}$</td>
<td>141</td>
</tr>
<tr>
<td>5.9.2.3</td>
<td>Measuring $t_{\text{MIN}}$</td>
<td>141</td>
</tr>
<tr>
<td>5.9.3</td>
<td>Discussion</td>
<td>142</td>
</tr>
<tr>
<td>5.9.3.1</td>
<td>The Fitts' law models</td>
<td>142</td>
</tr>
<tr>
<td>5.9.3.2</td>
<td>Fitts' law versus $t_{\text{REPEAT}}$</td>
<td>142</td>
</tr>
<tr>
<td>5.9.3.3</td>
<td>The value of $t_{\text{MIN}}$</td>
<td>143</td>
</tr>
<tr>
<td>5.10</td>
<td>The predicted expert typing speed</td>
<td>144</td>
</tr>
<tr>
<td>5.11</td>
<td>Conclusions</td>
<td>144</td>
</tr>
<tr>
<td>5.11.1</td>
<td>Validity of the prediction</td>
<td>145</td>
</tr>
<tr>
<td>5.11.2</td>
<td>Other contributions</td>
<td>145</td>
</tr>
<tr>
<td>5.11.3</td>
<td>Further work</td>
<td>147</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Conclusions and Future Work</td>
<td>148</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusions</td>
<td>148</td>
</tr>
<tr>
<td>6.1.1</td>
<td>The Fitts' digit model</td>
<td>148</td>
</tr>
<tr>
<td>6.1.2</td>
<td>The two-thumb typing model</td>
<td>150</td>
</tr>
<tr>
<td>6.2</td>
<td>Future work</td>
<td>151</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Ongoing work</td>
<td>152</td>
</tr>
<tr>
<td>Bibliography</td>
<td>154</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

Table 1 Relative frequency (%) of the 15 most frequent keystrokes from four users .................................................................23
Table 2 $27 \times 27$ digrams for the text-entry task ..........................................................58
Table 3 Ten most frequent digrams .............................................................................59
Table 4 Upper-bound and lower-bound predictions .................................................68
Table 5 Entry speed versus keyboard layout .........................................................90
Table 6 Expert predictions for various soft keyboard layouts ...............................97
Table 7 Frequency of the space character ............................................................115
Table 8 Digrams at start of word ...........................................................................116
Table 9 Digrams at end of word ............................................................................116
Table 10 Digitised Sharp $EL-6810$ miniature Qwerty keyboard ..........................117
Table 11 Values for $t_{\text{REPEAT}}$ from other publications ....................................122
Table 12 Sensitivity to the Fitts’ law slope coefficient ...........................................128
Table 13 Sensitivity to $t_{\text{MIN}}$ ........................................................................130
Table 14 Model sensitivity to corpus .....................................................................132
Table 15 Model sensitivity to space key policy .....................................................133
Table 16 Key patterns used for generating left and right hand Fitts’ law models .................................................................137
Table 17 Fitts’ law models ....................................................................................140
Table 18 Results of measurement of $t_{\text{REPEAT}}$ ..................................................141
Table 19 Comparison of predicted expert typing rates for three methods of mobile text entry .................................................................147
List of Figures

Figure 21  Performance comparison for stylus-tapping on a soft keyboard and several other text-entry methods……………………………………..69
Figure 22  The Dvorak keyboard…………………………………………………………..80
Figure 23  An alphabetic soft keyboard…………………………………………………..81
Figure 24  Textware Solutions Fitaly keyboard…………………………………………..82
Figure 25  Aiki Limited JustType keypad…………………………………………………86
Figure 26  Comparison of novice, quick test and expert predictions for six soft keyboard layouts…………………………………………………90
Figure 27  Some alphabetic keyboard arrangements……………………………………..98
Figure 28  Cubon keyboard (from Zhai et al. 2000)………………………………………99
Figure 29  The OPTI I and OPTI II soft keyboards (from MacKenzie et al. 1999b; Zhang 1998)…………………………………………………………101
Figure 30  Hook s and Metropolis I & II keyboards………………………………………103
Figure 31  The Getschow keyboard (from Getschow et al. 1986)………………………105
Figure 32  Lewis keyboard (from Lewis et al. 1999b; Lewis et al. 1999c)………………106
Figure 33  Útilware DotNote soft keyboard………………………………………………107
Figure 34  Two-thumb text entry……………………………………………………………110
Figure 35  Devices with miniature Qwerty keyboards…………………………………..111
Figure 36  Partition of keyboard for left and right thumbs ………………………………118
Figure 37  Example phrase with thumb assignment……………………………………118
Figure 38  Thumb transitions by space key policy………………………………………..120
Figure 39  Illustration of key repeat time…………………………………………………123
Figure 40  Computing entry time for a word…………………………………………….124
Figure 41  Sensitivity to the Fitts’ law slope coefficient…………………………………129
Figure 42  Sensitivity to t_{MIN}……………………………………………………………131
Figure 43  The modified Sharp EL-6053, showing the circuit board and protective cover underneath the keyboard ........................................ 135

Figure 44  A screen-print of the experimental software ........................................ 135

Figure 45  Fitts’ law data and regression lines ...................................................... 139

Figure 46  The minimum distance between two keys on our miniature Qwerty keyboard .......................................................... 140
List of Equations

\[ MT = a + b \times ID \]  \hspace{1cm} (1) ................................ ................................ ................................ .................. 60

\[ ID = \log_2 \left( \frac{A}{W} + 1 \right) \]  \hspace{1cm} (2) ............................................................................ 60

\[ MT_{ij} = a + b \times \log_2 \left( \frac{A_{ij}}{W} + 1 \right) \]  \hspace{1cm} (3) ............................................................................ 60

\[ MT_{ij} = \begin{cases} 
    a + b \times \log_2 \left( \frac{A_{ij}}{W} + 1 \right) & \text{if } i \neq j \\
    MT_{Repeat} & \text{if } i = j
\end{cases} \]  \hspace{1cm} (4) ............................................................................ 61

\[ \sum \sum P_{ij} = 1. \]  \hspace{1cm} (5) ............................................................................ 61

\[ MT = \sum \sum (P_{ij} \times MT_{ij}) \]  \hspace{1cm} (6) ............................................................................ 61

\[ CPS_{\text{max}} = \frac{1}{\bar{MT}} \]  \hspace{1cm} (7) ............................................................................ 62

\[ WPM_{\text{max}} = \frac{CPS_{\text{max}} \times 60}{5} \]  \hspace{1cm} (8) ............................................................................ 62

\[ RT = a' + b' \times \log_2(n) \]  \hspace{1cm} (9) ............................................................................ 62

\[ CPS_{\text{min}} = \frac{1}{\bar{MT} + RT} \]  \hspace{1cm} (9) ............................................................................ 62

\[ WPM_{\text{min}} = \frac{CPS_{\text{min}} \times 60}{5} \]  \hspace{1cm} (9) ............................................................................ 62

\[ RT = 0.2 \times \log_2(27) = 0.951 \text{ seconds.} \]  \hspace{1cm} (10) ............................................................................ 66

\[ MT_{\text{Repeat}} = \frac{1}{6.52} = 0.153 \text{ seconds.} \]  \hspace{1cm} (11) ............................................................................ 67
\[
MT_{ZX} = b \times \log_2 \left( \frac{A_{ZX}}{W} + 1 \right)
\]
\[
= \frac{1}{14} \times \log_2 \left( \frac{2.12}{2.12} + 1 \right) \quad (12) \]
\[
= 0.071
\]
\[
= 71 \text{ ms}
\]

\[
Entry Speed = \left( \frac{Entry Time}{44} \right)^{-1} \times \left( \frac{60}{5} \right) \quad (13)
\]

\[
t_{FITTS} = a + b \times ID \quad , \quad (14)
\]

\[
ID = \log_2 \left( \frac{A}{W} + 1 \right) \quad , \quad (15)
\]

\[
t_{MIN} = \frac{1}{2} \times t_{REPEAT} \quad (16)
\]

\[
t_1 = 0.2951 \times t_{FITTS} + 0.7049 \times t_{MIN} \quad (17)
\]

\[
t_2 = t_1 + t_{FITTS} \quad (18)
\]

\[
t_3 = \max(t_2 + t_{MIN}, t_0 + t_{FITTS}) \quad (19)
\]

\[
t_4 = \max(t_3 + t_{MIN}, t_2 + t_{FITTS}) \quad (20)
\]

\[
t_5 = \max(t_4 + t_{MIN}, t_3 + t_{FITTS}) \quad (21)
\]

\[
t_6 = \max(t_5 + t_{MIN}, t_4 + t_{FITTS}) \quad (22)
\]

\[
t_1 = 0.2951 \times t_{FITTS} + 0.7049 \times t_{MIN} \quad (23)
\]

\[
t_1 = 0.7049 \times t_{FITTS} + 0.2951 \times t_{MIN} \quad (24)
\]

\[
t_n = t_{n-1} + t_{FITTS} \quad (25)
\]

\[
t_n = \max(t_{n-1} + t_{MIN}, t_{RECENT} + t_{FITTS}) \quad (26)
\]

\[
MT = 176 + 64 \times \log_2(A/W + 1) \quad (27)
\]

\[
t_{WPM} = 60.74 \text{ wpm} \quad (28)
\]

\[
\]
\[ ID = \log_2 \left( \frac{A}{W} + 1 \right) = \log_2 \left( \frac{10.15}{5} + 1 \right) = 1.60 \text{ bits} . \quad (29) \]

\[ t_{\text{REPEAT}} = \frac{181.57 + 208.28}{2} = 194.93 \text{ milliseconds}. \quad (30) \]
Chapter 1
Introduction

This chapter provides an overview of mobile text input, from both practical and academic points of view. We begin with a brief history of the emergence and impact of mobile computers, personal information managers, and mobile communications devices. Key factors in conducting sound evaluations of new technologies for mobile text entry are presented, including methodology and experiment design. Important factors to consider are identified and elaborated upon, such as focus of attention, text creation versus text copy tasks, novice versus expert performance, quantitative versus qualitative measures, and the speed-accuracy trade off. A survey of mobile text entry techniques, both in research papers and in commercial products, is presented.

The following chapters extend the theory of text input in several directions. Chapter 2 presents a model of expert and novice stylus (or single-finger) typing, on soft keyboards. Chapter 3 presents the results of an evaluation of the typing model presented in Chapter 2; some shortcomings of the stylus model are discussed. Chapter 4 takes the stylus model in a different direction – a review is presented of recent research that has attempted to find optimised soft keyboard arrangements using the stylus typing model. A model describing two-thumb typing on a miniature Qwerty keyboard is described in Chapter 5. A sensitivity analysis of the model is presented, as well as the results of an experiment that measured several key features of two-thumb typing. Finally, conclusions and directions for future work are discussed in Chapter 6.

1 This chapter is an excerpt from a paper that has been accepted for publication as: MacKenzie, I. S. and Soukoreff, R. W. (2002a). Text entry for mobile computing: Models and methods, theory and practice. Human-Computer Interaction, in press.
1.1 Overview

Although text entry is by no means new in mobile computing, there has been a burst of research on the topic in recent years. There are several reasons for this heightened interest: First, mobile computing is on the rise and has spawned new application domains such as wearable computing, two-way paging, and mobile web and email access. Second, word processors, spreadsheets, personal schedulers, and other traditional desktop applications are increasingly available on mobile platforms. Third, there is a strong demand for the input of text or alphanumeric information that is easily and efficiently entered, recognized, stored, forwarded, or searched via traditional software techniques. Fourth, the phenomenal success of text messaging with mobile phone users has inspired considerable speculation on future spin-off technologies, all expected to benefit from text entry.

The statistics for text messaging on mobile phones are remarkable. In January 2001, GSM Europe reported that fifteen billion SMS text messages are transmitted per month worldwide. This is particularly interesting in view of the limited capability for text input with mobile phone technology.

While the ubiquitous Qwerty keyboard reigns supreme as the primary text entry device on desktop systems, mobile and handheld systems lack an equivalent dominant technology or technique for the same task. And so, the challenge of text entry for mobile computing presents itself. A valid question is “Why not just apply the Qwerty keyboard to the mobile paradigm?” The obvious advantage of Qwerty is familiarity. However, a Qwerty keyboard is bulky and unless the keyboard is full-size, touch typing is hampered or impossible. There several situations where the Qwerty keyboard is of limited applicability. (a) Some mobile devices are intended for

---

2 GSM stands for “Global System for Mobile communications”. The GSM Association, based in Dublin, Ireland, represents the interests of hundreds of satellite operators, manufacturers, suppliers, and regulatory and administrative bodies from around the world. See http://www.gsmworld.com/ for further details. SMS stands for “Short Message Service”. This is the predominant standard for mobile phone text messaging.
one handed use, and this reduces the advantage of the Qwerty arrangement. (b) Many mobile devices are committed to the pen-input paradigm, so a Qwerty keyboard is simply not an option. (c) Where physical buttons or keys are employed, the mobile form factor often limits the key complement to a dozen or less keys.

This chapter is organized as follows. We begin with a brief historical background for mobile and handheld devices. This is important because it juxtaposes the efforts of researchers with those of corporations that created early mobile and handheld devices. Following this, we offer some opinions and observations on the evaluation of text input technology. Many, but not all, of the text input technologies described later in this thesis have been empirically evaluated with user testing. To compare input technologies, the results of these evaluations are crucial. Factors to consider when performing evaluations are presented and elaborated. Following this, we detail one of the most active areas of current research – optimisation of text entry using language modelling and motor control modelling. Finally, we present a survey of the current state of the art in text entry for mobile computing. This chapter concludes with some observations about the technologies reviewed and the open research questions that remain.

There are two notable omissions in this thesis. One is speech recognition as a vehicle for text entry. Always “about to emerge”, speech is an input technology that is quick to grab headlines, but seems perennially unable to enter the mainstream of computing. In our view speech is a deserving (albeit niche) technology, but it is unlikely to supplant traditional interaction techniques for desktop or mobile computing. See Shneiderman (2000) for further discussion. Voice recognition technology is not supported by any products today for general-purpose text entry. Some recent models of mobile phones support limited voice recognition which allows the user to select from the pre-programmed phone numbers by voice, but this is hardly general-purpose text entry.
The other omission is international languages. By “text entry” we do not mean to limit ourselves to English text entry. Languages throughout the world are currently supported in various forms in mobile computing, and this will continue. While the focus in this thesis is on English, the discussions apply to other languages, particularly those based on the Roman alphabet. See Sacher (1998) for a discussion on text entry in Asian languages.

1.2 Background

1.2.1 Mobile devices

Among the earliest of handheld devices was the HP95LX, which was released in 1991 by Hewlett Packard (Palo Alto, CA; http://www.hp.com/). The technological equivalent of an IBM-XT shrunk into a clam-shell format, the HP95LX was small enough to fit in the palm of one’s hand. Although the term Personal Digital Assistant (PDA) had not yet entered the vernacular to describe a handheld computer, this was the first PDA. The HP95LX provided a small Qwerty keyboard for text entry, although touch-typing was impossible due to its size. Later devices (the HP100LX, and HP200LX) followed. These devices demonstrated that the Qwerty keyboard could be adapted (through miniaturisation) to mobile computing devices.

The early 1990s was an exciting time for mobile computing, due to the arrival of pen computing. The ideas touted much earlier by Kay and Goldberg (1977) in their Dynabook project had finally surfaced in commercial products. However, the initial devices were bulky, expensive, and power hungry, and they could not deliver in the one area that garnered the most attention – handwriting recognition. Without a keyboard, the pen was the primary input device. If only “selecting” or “annotating” were required, then the success of pen entry seemed assured. However, some applications demanded entry of text as machine-readable characters, and the handwriting recognition technology of the time was not up to the challenge.
Products from this era, such as GridPad, Momenta, Poquet, and PenPad, did not attain the volume of sales necessary for commercial viability. Most endured only a year or two.

One of the most significant events in pen-based computing was Apple Computer Inc.’s (Santa Clara, CA; http://www.apple.com/) announcement in 1993 that it would enter the pen computing market. Thus emerged the Apple MessagePad (a.k.a. Newton). Apple was a major player in desktop computing at the time, and its commitment to pen computing was taken seriously. To a certain extent, Apple added legitimacy to this entire segment of the computing market. However, the Newton was expensive and rather specialized. It was embraced by many technophiles but it did not significantly penetrate the larger desktop or consumer market. The Newton’s handwriting recognition, particularly on early models, was so poor that it was ridiculed in the media, for example by Garry Trudeau in his celebrated Doonesbury cartoons (Trudeau 1996). Nevertheless the Newton received considerable attention and it ultimately set the stage for future mobile devices.

The next significant event in mobile or pen computing was the release in 1996 of the Palm Pilot (now called the Palm) by Palm Inc. (Santa Clara, CA; http://www.palm.com/). The Palm was an instant hit. Six years hence, it is the technology of choice for millions of users of mobile devices. There is much speculation on why the Palm was so successful. Some factors seem relevant: The price of the Palm was about US$500, a few hundred less than a Newton. The Palm supported HotSync (including cables and software for transferring data between the Palm and a desktop computer) as a standard feature. The Palm was smaller and lighter than the Newton, and could fit into one’s pocket. Because of lower power consumption, the batteries lasted for weeks instead of hours. Finally, and perhaps most importantly, the Palm avoided the thorny issue of cursive or block-letter handwriting recognition by introducing a greatly simplified handwriting technique known as Graffiti. (Graffiti is discussed in Section 1.5.2.2.) By simplifying
recognition, Graffiti required less CPU power and memory, achieved better character recognition, and ultimately enjoyed widespread acceptance among users.

The year 1996 also saw the release of the Windows CE operating system by Microsoft (Redmond, WA; http://www.microsoft.com/). Devices such as Casio’s Cassiopeia or Philips’ Velo, which used Windows CE, were more powerful than previous mobile computing devices, but were also larger. The first version of Windows CE supported only a soft keyboard device for text entry, but later versions included the JOT handwriting recogniser, by Communications Intelligence Corp. (Redwood Shores, CA; http://www.cic.com/), and eventually the Microsoft Transcriber handwriting recogniser.

A recent entry in the pen computing market is the Crosspad by A. J. Cross Company (Lincoln, RI; http://www.cross.com/). The Crosspad avoids handwriting recognition by recording the user’s writing as ink trails. The user’s notes can be downloaded to a desktop computer as ink trails for storage and subsequent recognition on the desktop computer. Software accompanying the Crosspad supports handwriting recognition of keywords, and indexing and retrieval by keyword.

All of the devices mentioned above are handheld computers that support text entry. Another quite different group of devices that support text entry are messaging devices such as mobile phones and pagers. In Europe, where text messaging has been available since 1991, more text messages are transmitted daily than voice messages.3 In North America most mobile phones and pagers do not yet support text messaging but this is changing. The latest generation of two-way pagers such as the Blackberry by Research In Motion (Waterloo, Ontario, Canada;

---

3 Short Message Service (SMS) is the predominant text messaging technology in Europe. SMS supports transmission and reception of messages up to 160 characters long via mobile phones (phone-to-phone). Instant Messaging is a similar technology popular in North America, but it is used mostly in PC-to-phone messaging.
http://rim.net/, but also see http://www.blackberry.net/) and PageWriter by Motorola (Schaumburg, IL; http://www.motorola.com/) support text entry via a miniature Qwerty keyboard. The Treo by Handspring (Mountain View, CA; http://www.handspring.com/) is a cellular telephone combined with a Palm, that comes equipped with either a miniature Qwerty keyboard or Graffiti.

This perspective would not be complete without a hint at what the future might hold. At the top of our list of anticipated devices is a unit that combines the programmability of the PDA, wireless telephony, text messaging, and unfettered internet and email access. Pieces of this scenario already exist, but implementations require a specialized configuration, optional components, or support only a subset of standard features. These transitional technologies, that do not quite make the grade, are those that require an add-on RF transceiver, or provide internet access only to sites supporting a specialized protocol, for example, Wireless Access Protocol (or WAP).

For text input, the pen-based paradigm has dominated the PDA market, but there is a parallel trend toward text messaging in mobile phones and pagers using keyboard-based technology. If these technologies converge, then which text input technology will prevail? This is a difficult question to answer, and although there is no definitive answer, the following section identifies the key issues to consider.

1.2.2 Text entry

There are two competing paradigms that form the basis of text input methods: pen-based input and keyboard-based input. These paradigms emerged from two ancient technologies: typing and handwriting.\(^4\) User experience with typing and handwriting greatly influences expectations for text entry in mobile computing; however the two tasks are fundamentally different.

\(^4\)These are “ancient technologies” in that they predate computers.
A key feature of keyboard-based text entry is that it directly produces machine-readable text (i.e., ASCII characters). This is necessary for efficacious indexing, searching, and handling by contemporary character-based computing technology. Handwriting without character recognition produces “digital ink”. This is fine for some applications such as annotation, visual art, and graphic design. However, digital ink requires more memory and in general is not well managed by character-based computing technology. Specifically, digital ink cannot be easily indexed and searched (although Poon and colleagues, 1995, report some success with a graphical search mechanism for digital ink that is not based on recognition). For handwritten entry to achieve wide appeal it must be coupled with recognition technology.

An important consideration that is implicit in the discussion of text input technology is user satisfaction. The point was made earlier that the Palm succeeded where the Newton failed, in part because of users’ acceptance of Graffiti as a text input technology. Understanding and meeting user expectations is paramount in creating an acceptable text input technology. Users’ expectations for text entry are set by current practice. Touch typing speeds in the range of 20 to 40 words per minute are modest and achievable for hunt-and-peck typists. Rates in the 40 to 60 words per minute range are achievable for touch typists, and with practice, skilled touch typists can achieve rates greater than 60 words per minute. Handwriting speeds are commonly in the 15 to 25 words per minute range. These statistics are confirmed by several sources (Card, Moran and Newell 1983; Devoe 1967; Lewis 1999; MacKenzie, Nonnecke, Riddersma, McQueen and Meltz 1994a; Van Cott and Kinkade 1972). Users, perhaps unrealistically, expect to achieve text input rates within these ranges on mobile devices. Furthermore, they expect these rates immediately, or within a short time of using a new input technology.5 The preceding paragraphs have outlined qualities that a successful text input

---

5 It is worth noting that both handwriting and touch typing rates are very slow when compared with the speed of speech communications. Walker (1988) reports average speaking rates for young adults ranging between 172.6 and 197.4 words per minute, depending upon whether the task was reading aloud or conversation. The ease and speed of verbal communications remains a distant goal.
method should have. Production of machine-readable characters at a speed acceptable to users is a reasonable objective. To determine if a particular text input method meets this objective, or to compare new and existing text input methods, a user evaluation is performed. The following section discusses this important step in developing text entry techniques for mobile computing.

1.3 Evaluation

Research in mobile text entry is flourishing in part because user needs are not currently met by many mobile devices. Sometimes, completely new text input technologies are invented, while at other times refinements or enhancements of previous techniques are developed. Either way, evaluation is a critical and demanding part of the research program. The questions researchers pose are ambitious: “Can entry rates be improved if we arrange the buttons on a keyboard in a certain way?” or “What is the effect if we use context to guess the next letter or word?” or “Can we apply an altogether different technology, like pie menus, touch pads, or pattern recognition, to the problem of text input?” In this section, we discuss some important issues in undertaking valid and useful evaluations of text entry techniques.

1.3.1 Methodology

An evaluation is valuable and useful if the methodology is reproducible and results are generalizable. “Reproducible” implies that other researchers can duplicate the method to confirm or refute results. This is achieved for the most part simply by following an appropriate reporting style (e.g., APA 1995). “Generalizable” implies that results have implications beyond the narrow context of the controlled experiment. This is achieved through a well-designed experiment that gathers measures that are accurate and relevant, and in tasks that are representative of real-life behaviour. There is, unfortunately, a trade off here. In “real-life”, people rarely focus solely on a single task. Methodologies designed to use real-life tasks,
therefore, may find that measurements so gathered include spurious behaviours such as pondering, or secondary tasks. The trade off, therefore, is between the accuracy of our answers and the importance or relevance of the questions they seek to address. That is, we can choose between providing accurate answers to narrow questions, or providing vague answers to broad questions. The reader is implored not to interpret this too strictly, but, hopefully, the point is made. In designing an experiment, we strive for the best of both worlds: answering interesting or broad questions (viz. using real-life tasks), and doing so accurately (viz. accurately measuring the behaviour of interest, such as entry speed or typing accuracy).

In the following sections, we identify some factors relevant to methodologies for evaluating text entry on mobile systems.

1.3.2 Text copy tasks versus text creation tasks

An important distinction in text entry evaluations is between text copy and text creation tasks. In a text copy task, the subject is given text to enter using the input technique under investigation. In a creation task the source text is either memorised or generated by the subject. Related to these two types of tasks is the concept of Focus of Attention (FOA). FOA speaks to the attention demands of the task. Consider the case of an expert touch-typist using a Qwerty keyboard to copy text from a nearby sheet of paper. This is a single FOA task. The typist attends only to the source text, because the typist can feel the keyboard and does not need to look at it when typing. However, if the text is entered using a stylus on a soft keyboard, users must also visually attend to the keyboard. (A soft keyboard cannot be operated “eyes free”.) Stylus typing, therefore, is a two FOA task. If users make errors and correct as they go, they must look at the screen to monitor results. This increases touch typing to a two FOA task, and stylus typing to a three FOA task. Clearly, the feedback channel is overburdened in a three FOA task.
Despite the above, evaluations involving text copy tasks are generally preferred to those with a text creation task. One difficulty with text creation tasks is identifying errors – it is difficult to know exactly what a subject intended to enter if the subject is generating the text. Even if the message content is known \textit{a priori}, errors in spelling or memory recall may occur, and these meta-level mistakes are often indistinguishable from errors due to the interface itself.

A second difficulty in text creation tasks is the loss of control over the distribution of letters and words entered. The task should require the subject to enter a representative number of occurrences of characters or words in the language (i.e., results are generalizable). However, it is not possible to control for this if the subject is generating the text.

A third difficulty in text creation tasks is the measurement of text entry speed. Measurements will include the time invested in pondering – thinking about what to write. It is difficult to separate this from the effort in actually entering the text.

The main advantage of a text creation task is that it mimics typical usage. The disadvantages just cited, however, are significant and drive most researchers to use text copy tasks despite the increased FOA noted earlier. One way to mitigate the effects of increased FOA is to dictate the source text to the subjects. Ward and colleagues (2000) used this technique, however they noted that subjects found the approach stressful and hard to follow.

However, a carefully designed experiment may capture the strengths of both a text creation task and a text copy task. One technique is to present subjects with short, easy-to-memorize phrases of text. Subjects are directed to read and memorize each phrase before entering it. Entry proceeding in this fashion benefits from the desirable property of reduced FOA (like a text creation task). As well, the desirable properties of a text copy task are captured, i.e., control over letter or word
frequencies, and performance measurements that exclude thinking about what to write. There are numerous examples of this approach in the literature (e.g., Alsio and Goldstein 2000; MacKenzie et al. 1994a; MacKenzie and Zhang 1999b; Rau and Skiena 1994). A similar technique is to present text in a large block (e.g., a complete paragraph) but to visually interleave each line of the presented text (input) with each line of generated text (output), on a computer monitor. As input proceeds, each character entered appears directly below, and in close physical proximity to, the intended character. This is a text copy task, however FOA is reduced to that of a text creation task, because subjects attend only to one location for both the source text and the results of entry. An example of this methodology is reported by Matias and colleagues (Matias, MacKenzie and Buxton 1993; Matias, MacKenzie and Buxton 1996a).

1.3.3 Novice versus expert performance

Most work on the design of text input methods focuses on the potential, or expert, text entry rate of a particular design. However, the novice experience is paramount for the success of new text input methods. This is at least partially due to the target market. Mobile and handheld products, such as mobile phones and PDAs, once specialized tools for professionals, are increasingly targeted for the consumer market. It follows that “immediate usability” is important. In other words, it may be a moot point to establish the expert, or “potential” text entry rate for an input technique if prolonged practice is required to achieve it. Consumers may be discouraged by their initial experience and frustration, and never invest the necessary effort to become experts.

However, measuring immediate usability is easier said than done. In typical studies of new interaction techniques, participants are given a demonstration of the technique followed by a brief practice session. Then, data collection proceeds over several blocks of trials. However, the measurements are a poor indicator of novice behaviour, at least in the sense of immediate, or walk-up, usability. Within a few
minutes, participants' knowledge of the interaction technique develops and the novice status fades. Measuring expert performance is also not easy, since acquisition of expertise requires many blocks of trials over many days, or more.

Some longitudinal text entry studies evaluations are hereby cited (Bellman and MacKenzie 1998; Gopher and Raij 1988; MacKenzie et al. 1999b; Matias et al. 1996a; McMulkin 1992). An example of results from a typical longitudinal study appear in Figure 1. Users’ improvement in entry speed is shown over 20 sessions of input for two types of soft keyboards. The data were fitted to the standard power law of learning (see Card, English and Burr 1978). Prediction equations and squared correlations are shown, as are extrapolations of the predictions to 50 sessions.
1.3.4 Quantitative versus qualitative analyses

When reporting quantitative results, there are many common pitfalls to avoid such as inaccuracy in measurements, lack of control or baseline conditions, inferring too much from data, using too small a sample size, collecting insufficient data, artificially biasing data by aggregation, non-random presentation of conditions, or inappropriate treatment of outliers. The reader is directed to textbooks in experimental psychology for further discussions (e.g., Martin 1996). For a discussion on aggregation bias, see Walker (1993).

Researchers may be excused for bending the rules, perhaps, but all too common are published reports stating only qualitative results steeped in anecdote, or, worse yet, testimonials unsupported by empirical data. An excerpt from one such publication illustrates our point:

While we have yet not done systematic user testing, anecdotal experience to date is consistent: Users well practiced in both ... and ... consistently find the latter to be about three times faster, with accuracy for both systems very high.\footnote{An excerpt from a paper published in the proceedings of a conference in human-computer interaction.}

Testimonials such as this surely do not meet the criteria for good research – that the results are generalizable and reproducible. Unless a controlled experiment is performed using quantitative metrics or established qualitative test instruments, there is no way to gauge the performance of a new text input technique. Conjuring up a new input technique is fine, but research demands more. It demands that new ideas are implemented and evaluated in conformance with the rigors of an empirical evaluation.

Although quantitative tests form the backbone of any scientific study, qualitative aspects of the investigation are also important. In human-computer interfaces,
users must feel comfortable with the interaction technique and must feel their efforts have a reasonable payoff in their ability to accomplish tasks. Participants develop impressions of each device or condition tested, and these should be solicited and accounted for in the final analysis. Typically, these opinions are sought via a questionnaire administered at the end of a condition or experiment. The reader is referred to textbooks in human-computer interaction for direction in questionnaire design (e.g., Dix, Finlay, Abowd and Beale 1998).

1.3.5 Speed

Within the domain of text input there are two primary evaluation metrics: speed and accuracy. The simplest way to measure and report speed is to measure the number of characters entered per second during a trial, perhaps averaged over blocks of trials. This gives a measure in characters per second (cps). To convert this to words per minute (wpm) the standard typists' definition of a word as five characters (regardless of whether the characters are letters, punctuation, or spaces) is employed (Gentner, Grudin, Larochelle, Norman and Rumelhart 1983). Therefore, words per minute is obtained by multiplying characters per second by 60 (seconds per minute) and dividing by 5 (characters per word).

1.3.6 Accuracy

Accuracy is more problematic. For a simple treatment of accuracy, we obtain a metric that captures the number of characters in error during a trial, and report these as a percentage of all characters in the presented text. A more complete analysis involves determining what kind of errors occurred, and why. The difficulty arises from the compounding nature of mistakes (see Suhm, Myers and Waibel 1999, for some elaboration), and the desire to automate as much of the data measurement and analysis as possible. Four basic types of errors include entering an incorrect character (substitution), omitting a character (omission), adding an extra character (insertion), or, swapping neighbouring characters (transposition). While it is
straightforward for a human to compare the intended text with the generated text and tabulate the errors, in practice the amount of analysis is simply too much, given a reasonable number of subjects, conditions, and trials. Additionally, tabulation errors may be introduced if performed manually.

However, automating error tabulation is not trivial. Consider an experiment where the subject is required to enter the 19-character phrase: “the quick brown fox”. If the subject enters “the quxxi brown fox”, the incorrect word contains either three substitution errors, or two insertion (“xx”) and two omission (“ck”) errors. The explanation with the fewest total number of errors (3) is preferred, and, in this simple example, yields an error rate of \((3 / 19) \times 100\% = 15.8\%\). Algorithms for “string distance” calculations, such as the Levenshtein string distance statistic (Damerau 1964; Levenshtein 1966), might assist in automating analyses such as these; however this has not been applied as yet to measuring errors in text entry tasks. We have explored the merits of the Levenshtein string distance statistic as a measure of errors in text entry tasks. Initial results appear in Soukoreff and MacKenzie (2001).

Analysis of errors is slightly different when considering word-level errors. In this case, any error within a word is logged as one error, and errors are reported as the percentage of incorrect words. Now consider the erroneous text “the quick quick fox”. The error here is likely due to the subject not following the source text closely enough. This should probably be counted as one word-level error, rather than a series of character-level errors.

A useful tool for designers is the confusion matrix, graphically depicting the frequency of character-level erroneous transcription errors. Figure 2 is a confusion matrix taken from Chang and MacKenzie’s (1994) comparative study of two handwriting recognisers. The confusion matrix displays intended characters versus recognized characters illustrating how often an intended character (left-hand
The characters that the subjects were instructed to enter appear vertically along the left. The characters that were actually recognised appear horizontally along the bottom. Frequently misrecognised characters, such as l-I, k-K, i-I, and g-s, are readily apparent.

Note that correctly recognised characters are not shown. Each dot represents three misrecognition errors.
column) was misrecognised and interpreted as another character (bottom row). Each dot represents three occurrences.

Difficulties in error tabulation have pushed some researchers to ignore errors altogether (e.g., Venolia and Neiberg 1994), or to force the subject to enter correct text only (e.g., Lewis 1999).

Directing subjects to “correct as you go” is another possible approach. Assuming subjects adhere to the instructions, the resulting text is error free; thus, the error rate is 0%. In general though, subjects will leave errors in the generated text, even if requested not to. This results in two kinds of errors: those that were corrected, and those that were not. In any case however, overhead is incurred by correcting errors along the way. One possible measure of accuracy is keystrokes per character (KSPC) (MacKenzie 2002a; Soukoreff et al. 2001). The ideal is $KSPC = 1.0$, but, in practice, $KSPC > 1$ if subjects correct as they go. If, for example, a 25-character phrase was entered and two substitution errors occurred, each corrected by pressing backspace followed by the correct character, then $KSPC = (25 + 4) / 25 = 1.16$.

Reaction time is a factor that comes into play when correct-as-you-go is employed. If text entry proceeds quickly an error may be followed by several additional entries before the subject can react to the error. The overhead in correcting the error may be substantial (see Matias et al. 1996a for a discussion of this).

Clearly, both speed and accuracy must be measured and analysed. Speed and accuracy are well known to exist in a continuum, wherein speed is traded for accuracy or vice versa (Hancock and Newell 1985; Pachella and Pew 1968; Pew 1969; Swensson 1972; Wickelgren 1977). Subjects can enter text more quickly if they are willing to sacrifice accuracy. For subjects to perform with high accuracy, they must slow down. The trade-off suggests that measuring only speed or only accuracy will skew the results so as to make the text input method appear better (or worse) than
it really is. An example of a reporting technique that combines speed and accuracy is given in Figure 3. Conditions are “better” toward the top and right of the figure, because they are both fast and accurate.

![Graph showing speed and accuracy comparison](image)

Figure 3 - Simultaneous presentation of results for speed and accuracy (from MacKenzie, Nonnecke, McQueen, Riddersma and Meltz 1994b)

1.3.7 Other factors

The text input process can be significantly impacted by factors that have little to do with the input device, such as whether a device is operated while standing, sitting, or walking, or whether a device is operated with one or two hands. Designers of novel text input techniques must be aware that users want to operate mobile devices anytime, anywhere. Lack of a one-hand interaction method impacts the commercial success of a technology.
Evaluations are often conducted to test a refinement to existing practice. Often the new technique is only a minor improvement or optimisation over the status quo. Some key initiatives in improving current practice through language and movement modelling are presented in the next section.

1.4 Optimisation techniques

There are two popular approaches to optimising the text entry task: movement minimisation, and language prediction. Movement minimisation seeks to reduce the movements of the finger or pen in interacting with a mobile device to enter text. Language prediction exploits the statistical nature of a language to predict what the user is about to enter. There are also hybrid approaches. The following sections summarise these modelling and design techniques.

1.4.1 Movement-minimisation

The main reason for using a Qwerty keyboard for text input is to support touch typing. Without touch typing, the next-best reason is familiarity with the letter arrangement. However, the shape (not tall, but very wide) and size of a Qwerty keyboard is imposing and ill-suited to the mobile paradigm. Recent work has focused on the limited case of single-finger or stylus entry, either on a soft keyboard or on a small physical keyboard with a reduced key set. This work combines a statistical language model with a movement time prediction model to assist in modelling and designing input techniques wherein device or hand movement is as efficient as possible (Hunter, Zhai and Smith 2000; Lewis, Allard and Hudson 1999a; Lewis, LaLomia and Kennedy 1999b; Lewis, LaLomia and Kennedy 1999c; MacKenzie et al. 1999b; MacKenzie, Zhang and Soukoreff 1999c; Zhai, Hunter and Smith 2000; Zhang 1998).

In 1995 we published a model (Soukoreff and MacKenzie 1995) of stylus typing that predicts typing rates for expert and novice users of a soft keyboard. The model is
comprised of five major components: (a) a digitised layout of the keyboard in question, (b) Fitts’ law for rapid aimed movements, (c) the Hick-Hyman law for choice selection time, (d) a linguistic table for the relative frequencies of letter pairs, or digrams, in common English, and (e) a spreadsheet in which the preceding components are combined. The result is a general behavioural description and predictive model of the task of text entry with a stylus and soft keyboard. The predictions of the model are approximate, but useful. This model is the subject of Chapter 2, and is presented in detail there.

This model has been subsequently used by others seeking to find an optimal keyboard for stylus typing (Hunter et al. 2000; MacKenzie et al. 1999c; Zhai et al. 2000; Zhai, Hunter and Smith 2002; Zhang 1998). Their efforts are reported in Chapter 4.

1.4.2 Language prediction

Predictive text input techniques strive to reduce the input burden by predicting what text the user is about to enter. This is accomplished by analysing a large collection of documents – a corpus – to establish the relative frequency of characters, digrams (pairs of characters), trigrams, words, or phrases in the language of interest. These statistical properties are used to suggest or predict letters or words as text is entered. The seminal publication in the area of text prediction is by Shannon (1951), and, although there are many ways to implement text prediction, most are based upon this paper.

Predictive input technologies have the capacity to significantly reduce the effort required to enter text – if the prediction is good. There are a few caveats to consider in basing a language model on a standard corpus, however. These include: (a) the corpus may not be representative of the user language, (b) the corpus does not reflect the editing process, and, (c) the corpus does not reflect input modalities. An explanation of these points follows.
1.4.2.1 Corpus not representative of the user language

The idea that a corpus is “representative of a language” is questionable when the domain is users interacting with computing technology. Users typically use a much richer set of characters and words than appear in any corpus, and the statistical properties in that set may be different from those in the corpus. A simple example is the space key, which is the most common character in English text (Soukoreff et al. 1995). Yet, the space character is typically missing in tables of letter or digram probabilities used to build language models (e.g., Mayzner and Tresselt 1965; Underwood and Schulz 1960). Typically, only alphabetic characters are included.

As well, punctuation symbols are rarely included in letter or digram tables. Both Isokoski (1999) and Zhai et al. (2000) observe that some punctuation symbols occur more frequently than some of the less common letters. The simple inclusion of the space character and punctuation symbols is the first step. We feel it is important to fully open the character set.

The characteristics of the text entered are dependent on the application. For example, we expect more formal prose to be entered using a word processor than an e-mail application. Additionally, the type of application depends upon the input device available – few people have the patience to enter volumes of text into a hand-held PDA device. The kind of text most likely entered in this context is short notes, phone numbers, URLs, acronyms, slang, etc., the statistical properties of which differ from formal English texts. Highly cryptic messages are common for text entry on cell phones.

1.4.2.2 Corpus ignores the editing process

A corpus contains no information about the editing process, and we feel this is an unfortunate omission. Users are fallible and the creation of a text message – or interaction with a system on a larger scale – involves much more than the perfect
sequential input of alphanumeric symbols. The input process is really the editing process.

Recently, we conducted a study to monitor and analyse keystroke-level interaction with desktop systems. Over a period of two months we logged all keystrokes (more than 400,000) for four desktop computer users. Table 1 shows the 15 most common keystrokes. Common editing keys, such as cursor up, down, and backspace, figure very prominently in the table. Although mobile users engage a much different interface, the data in Table 1 serve as a warning that input with computing technology, in general, is much richer than represented in a corpus.

Table 1 - Relative frequency (%) of the 15 most frequent keystrokes from four users

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.18</td>
<td>Space</td>
<td>10.42</td>
<td>Down</td>
<td>12.87</td>
<td>Space</td>
</tr>
<tr>
<td>7.14</td>
<td>Back</td>
<td>7.95</td>
<td>Space</td>
<td>8.69</td>
<td>Back</td>
</tr>
<tr>
<td>5.29</td>
<td>Down</td>
<td>5.57</td>
<td>Up</td>
<td>7.36</td>
<td>E</td>
</tr>
<tr>
<td>4.93</td>
<td>E</td>
<td>5.35</td>
<td>Shift</td>
<td>6.07</td>
<td>T</td>
</tr>
<tr>
<td>4.19</td>
<td>A</td>
<td>5.33</td>
<td>Right</td>
<td>5.05</td>
<td>O</td>
</tr>
<tr>
<td>3.85</td>
<td>Shift</td>
<td>4.49</td>
<td>Control</td>
<td>4.64</td>
<td>I</td>
</tr>
<tr>
<td>3.84</td>
<td>I</td>
<td>4.00</td>
<td>E</td>
<td>4.45</td>
<td>A</td>
</tr>
<tr>
<td>3.42</td>
<td>O</td>
<td>3.96</td>
<td>Left</td>
<td>4.18</td>
<td>S</td>
</tr>
<tr>
<td>3.28</td>
<td>T</td>
<td>3.73</td>
<td>Delete</td>
<td>4.16</td>
<td>N</td>
</tr>
<tr>
<td>3.27</td>
<td>R</td>
<td>3.25</td>
<td>T</td>
<td>3.79</td>
<td>R</td>
</tr>
<tr>
<td>3.22</td>
<td>N</td>
<td>3.12</td>
<td>S</td>
<td>3.46</td>
<td>Shift</td>
</tr>
<tr>
<td>2.98</td>
<td>Up</td>
<td>2.54</td>
<td>O</td>
<td>2.68</td>
<td>H</td>
</tr>
<tr>
<td>2.92</td>
<td>Right</td>
<td>2.54</td>
<td>A</td>
<td>2.32</td>
<td>L</td>
</tr>
<tr>
<td>2.72</td>
<td>S</td>
<td>2.42</td>
<td>Back</td>
<td>2.24</td>
<td>C</td>
</tr>
<tr>
<td>2.48</td>
<td>Delete</td>
<td>2.38</td>
<td>I</td>
<td>2.14</td>
<td>D</td>
</tr>
</tbody>
</table>

1.4.2.3 Corpus does not capture input modalities

Text documents do not reflect how they were created. For example, a corpus includes both capital and lowercase characters. In simple language models this distinction is ignored (e.g., “A” and “a” are considered the same). A more expansive
model can easily accommodate this distinction simply by treating capital and lowercase characters as distinct symbols. Yet, from the input perspective, both approaches are wrong. Uppercase and lowercase characters are never entered via separate keys on a keyboard; thus, the seemingly more accurate treatment of capital and lowercase characters as distinct symbols is just as wrong.

For the user’s interaction with the shift and caps-lock keys to be accommodated in a model of text input, activity with these and related keys should be included in the language model. In other words, it is the “language of interaction” that should be modelled. Note in Table 1 that the shift key fairs no worse than eleventh in the list of most-frequent keys.

### 1.4.3 Hybrid input techniques

Some text input techniques include both movement-minimizing and predictive features. *Dasher* (Ward et al. 2000) is a predictive text input technique using a pointing device to select from predicted options. The options are presented to the user in boxes sized according to their relative probabilities. The boxes scroll and expand as the pointing device hovers near them (using graphics somewhat like a video game) allowing rapid text entry. Thus, the technique is both movement-minimizing and predictive. An online demo is available ([http://wol.ra.phy.cam.ac.uk/mackay/dasher/](http://wol.ra.phy.cam.ac.uk/mackay/dasher/)). Although *Dasher* is an interesting technology, it is not clear that it is scalable for the small graphics screens common on mobile and handheld devices.

### 1.4.4 Key minimization techniques (modes)

Because space is limited on small devices, keyboards that minimise the number of keys are of interest. However, users desire a large set of characters including the alphabet, numbers, symbols, and editing keys. An example of this is the standard PC-compatible 101-key keyboard. Although the standard PC keyboard has 101 keys,
a user can produce closer to 800 individual keystrokes (each key can be hit by itself or with any combination of shift, control, or alt – and the num-lock key changes the mode of the numeric keypad). The keys on the standard PC keyboard are, therefore, ambiguous; disambiguation is accomplished with the various mode keys.

There is another way to disambiguate keystrokes. Some keyboards are designed with more than one letter on each key (e.g., the alphabetic characters on a standard telephone keypad). Text entered on these is inherently ambiguous, because different character strings correspond to the same key presses. For example, on a standard telephone keypad, both “GAP” and “HAS” correspond to the key sequence 4-2-7. Disambiguation technology takes key press sequences and uses an embedded database of language statistics to identify legal words that are presented to the user for verification. Automated disambiguation promises to increase the speed and accuracy of text input on ambiguous keyboards.

Conceptually, we can think of key ambiguity as a continuum (see Figure 4). At one extreme, we have a keyboard with a dedicated key for each symbol in the language (Figure 4a), while at the other we have just one key that maps to every symbol in the language (Figure 4d).

The keyboard in Figure 4d would be very fast\(^7\), because only one key is pressed. However, it is of no practical use, since each key press is ambiguous to the entire set of symbols in the language. Clearly, Figure 4d is little more than a theoretical curiosity. The Qwerty keyboard (Figure 4b) and telephone keypad (Figure 4c) represent two relevant points in the continuum.

---

\(^7\) In fact, the text entry rate for this keyboard would be about 78.4 words per minute (wpm). This figure is derived from the single finger key repeat time of 0.153 seconds reported by Soukoreff and MacKenzie (1995). The text entry rate is \((1 / 0.153) \times (60 / 5) = 78.4\) wpm. The key repeat time may be as low as 0.127 seconds (Zhai 2000), and, in this case, the upper bound is \((1 / 0.127) \times (60 / 5) = 94.5\) wpm.
(a) A fictitious alphabetic keyboard with distinct keys for capital and lower case letters, (b) Qwerty keyboard, (c) standard telephone keypad, (d) hypothetical single-key keyboard which would require either many mode keys or a near psychic disambiguation algorithm to be useful.

Figure 4 - The key-ambiguity continuum
The previous sections introduced many issues facing researchers in mobile text input, and we have delineated the design space within which this research takes place. The following section presents a survey of mobile text entry techniques as found in research papers and commercial products.

1.5 Survey of text entry techniques

The survey is divided into key-based and stylus-based text input methods.

1.5.1 Key-based text entry

Key-based text entry techniques range from those that use a keyboard where each key represents one or more letters, to those with as few as three keys.

1.5.1.1 Telephone keypad

The desire for effective text entry using the telephone keypad is fuelled by the increase in text messaging services, and the movement toward consolidation of technologies such as wireless telephony and PDAs. Text entry on a mobile phone is based on the standard 12-key telephone keypad (see Figure 5).

The 12-key keypad consists of number keys 0 to 9 and two additional keys (* and #). Characters A to Z are spread over keys 2 to 9 in alphabetic order. The placement of characters is similar in most mobile phones, as it is based on an international standard (Grover, King and Kuschler 1998). The placement of the space character varies among phones, however it is usually entered with a single press of the 0 key. Since there are fewer keys than the 26 needed for the characters A to Z, three or four characters are grouped on each key, and so as noted earlier, ambiguity arises. There are three main approaches to text entry on a phone keypad: multi-tap, two-key, and one-key with disambiguation.
The multi-tap method is currently the most common text input method for mobile phones. With this approach, the user presses each key one or more times to specify the input character. For example, the 2 key is pressed once for the character A, twice for B, and three times for C. The multi-tap approach suffers from the problem of segmentation, when a character is on the same key as the previous character (e.g., the word “ON” because both O and N are on the 6 key). To enter the word “ON” the user presses the 6 key three times, waits for the system to timeout, and then presses the 6 key twice more to enter the N. Another solution to the segmentation problem is to have a special key to skip the timeout (this is termed “timeout kill”) thus allowing direct entry of the next character on the same key. Some phone models use a combination of the two solutions. For example, Nokia (Nokia Group, Finland; http://www.nokia.com/) phones include both a 1.5 second timeout and the provision for a timeout kill using the down-arrow key. The user decides which strategy to use.

In the two-key method, the user presses two keys successively to specify a character. The first key selects the group of characters (e.g., the 5 key for J, K, or L). The second key specifies the position within the group. For example to enter the character K the user presses 5 followed by 2 (K is second character in JKL). Although the two-key method is quite simple, it is not in common use for entering Roman letters. However, in Japan a similar method (often called the “pager” input
method) is very common for entering Katakana characters.

A third way to overcome the problem of ambiguity is to add linguistic knowledge to the system. We call this technique one-key with disambiguation. An example is T9 by Tegic Communications, Inc. (Seattle, WA; http://www.tegic.com/). When using T9 each key is pressed only once. For example, to enter “THE”, the user enters 8-4-3-0. The 0 key represents the space character – delimiting words and terminating disambiguation of the preceding keystrokes. T9 compares the word possibilities to a linguistic database to guess the intended word.

Naturally, linguistic disambiguation is not perfect, since multiple words may have the same key sequence. In these cases the most common word is the default. A simple example follows using the well-known “quick brown fox” phrase: (words are shown top-to-bottom, most probable at the top)

```
843 78425 27696 369 58677 6837 843 5299 364
the quick brown fox jumps over the jazz dog
tie stick crown lumps muds tie lazy fog
vie vie
```

Of the nine words in the phrase, eight are ambiguous, given the required key sequence. For seven of the eight, however, the most probable word is the intended word. The most probable word is not the intended word just once, with “jazz” being more probable in English than “lazy”. In this case, the user must press additional keys to obtain the desired word. Evidently, the term “one-key” in “one-key with disambiguation” is an oversimplification!

Silfverberg et al. (2000) present predictive models of these three text input methods based on the model of Soukoreff and MacKenzie (1995). They report that the disambiguation of T9 works reasonably well, with expert predictions ranging from 41 to 46 wpm. These figures are coincident with rather broad assumptions,
however. These include (a) all words entered are unambiguous, (b) users are experts (i.e., no typing, spelling, or other errors), and, (c) all words entered are in the dictionary. Their predictions are, at best, an upper-bound.

Many mobile phone manufacturers have licensed the T9 input technology, and since 1999 it has begun to surface in commercial products, for example, the Mitsubishi (Tokyo, Japan; http://www.mitsubishi.com/) MA125, the Motorola i1000Plus, and the Nokia 7110. There is also a touch-screen version of T9 that is available for PDAs. Bohan et al. (1999) describe an evaluation of the touch screen version.

T9 is the first disambiguating technology to work with a standard mobile phone keypad, but not the only such technology. Motorola’s iTAP is disambiguating technology similar to T9. Both iTAP and T9 support multiple languages. The Chinese version of iTAP uses a nine-key input method for writing the various strokes, that offers users more keystroke choices and is easy to learn (Sacher 1998). Another similar technology is eZiText by Zi Corp. (Calgary, Alberta, Canada; http://www.zicorp.com/). No published evaluations exist of iTAP or of eZiText.

A slightly different approach is presented in WordWise by Eatoni Ergonomics (New York, NY; http://www.eatoni.com/). To aid in disambiguation a mode shift is used to explicitly choose one character from each key, the other characters remain ambiguous; this achieves partial disambiguation. Figure 6 illustrates the WordWise keypad.

The mode shift is implemented either with the 1 key (shown in Figure 6) or using a thumb-activated key on the side of the mobile phone. For example, to enter the letter C, the shift (1) key is pressed followed by the 2 key. To enter the letter A, the 2 key is pressed by itself and automatic disambiguation determines whether the user intended to enter A or B. The letters chosen for the mode shift are C, E, H, L, N, S, T, and Y, which are the most popular letters in each group (on each key).
These letters were chosen to provide maximum separation for the disambiguation algorithm. One beneficial side effect of the mode shift is that words that are explicit (e.g., “THE” which is entered by holding shift while entering 8-4-3) can be omitted in the internal database. This greatly reduces the memory requirements of the implementation – a critical factor for mobile phones.

![Figure 6 - Eatoni Ergonomics WordWise keypad](image)

The 1 key acts as a shift to explicitly select one letter on each of the other keys.

All text input methods based on the telephone keypad require the attention of the user, especially when disambiguation is used. A typical text creation task has two FOA. (The user attends to both the keypad and the display.) With practice, it may be possible for users to become familiar enough with the multi-tap and two-key input methods that they would not need to look at the telephone. Telephone keypad text entry is much easier with two hands – one to hold the device and the other to type with. The models created by Silfverberg et al. (2000) predict about 21 to 27 words per minute for the multi-tap method and the two-key method.

1.5.1.2 Small Qwerty keyboards

Another common text input technology is the miniature Qwerty keyboard, which is especially prevalent in the low-end mobile computers, and personal information
managers (PIMs). There are many examples, such as the HP Jornada, the Sharp (Osaka, Japan; http://sharp-world.com/) Zaurus, the Sharp Mobilon, and the Psion (London, United Kingdom; http://psion.com/) Revo. Two-way pagers support text input and at least two companies have pager products with miniature Qwerty keyboards.

The Blackberry by Research In Motion is a two-way pager with a small Qwerty keyboard. (See Figure 7a.) The keyboard is too small for touch typing, but it is suitable for one or two finger typing. Motorola has a similar product called the PageWriter. (See Figure 7b.)

The Nokia Communicator is a mobile phone with text messaging functionality. It looks like a typical mobile phone when operated as a phone, but opens to reveal a large LCD screen and miniature Qwerty keyboard inside. (See Figure 8.)

The Blackberry, PageWriter, and Communicator are representative of small devices that have stayed with the Qwerty paradigm, and they are by no means alone. There are many similar devices on the market, including the personal information managers noted earlier.

There is another way to reduce the size of a Qwerty-like keyboard. Matias and colleagues proposed a clever way to half the size of the keyboard and still leverage touch-typing skills (Matias et al. 1993; Matias, MacKenzie and Buxton 1994; Matias et al. 1996a; Matias, MacKenzie and Buxton 1996b). The Half-Qwerty keyboard, commercialised by the Matias Corporation (Rexdale, Ontario, Canada; 8 The distinction between a personal digital assistant (PDA), and a personal information manager (PIM) is dubious. For the purposes of this document, a PDA refers to a general-purpose mobile computer; a PIM is a device with a static set of applications, that typical users cannot write software for. PDAs are more expensive, more powerful, and usually employ more elaborate text input methods than PIMs.
Figure 7 - Two devices with miniature Qwerty keyboards
(a) Research in Motion BlackBerry (RIM 957) (the actual size is 79 × 117 mm), (b) Motorola Pagewriter 2000X (the actual size is 95 × 71 mm)

Figure 8 - Nokia Communicator 9110
The actual size is 158 × 112 mm.
http://www.halfquerty.com/), is a regular Qwerty keyboard that is split in half. Therefore, there are two possible Half-Qwerty keyboards (see Figure 9), one corresponds to the left half of the Qwerty keyboard, the other to the right. When using the Half-Qwerty keyboard, characters that happen to appear on the half of the keyboard being used are typed in the normal fashion – by simply hitting the appropriate key. Keys from the missing half of the keyboard are entered by holding down the space key while hitting the mirror image of the appropriate key.

Figure 9 - Matias Corporation Half-Qwerty keyboard
If implemented using a desktop keyboard, either half may be used.
Characters from the missing half of the keyboard are arranged as the mirror image of their normal arrangement; this means that the relative finger movements used for one-handed typing are the same as those for two-handed typing – but are made with the opposite hand. Hitting the space bar alone types a space. Because there are two possible Half-Qwerty keyboards, either hand can be used.

Matias and colleagues report the results of a rigorous user evaluation of the Half-Qwerty (Matias et al. 1993; Matias et al. 1996a). Right-handed subjects using their left hands reached 50% of their two-handed typing speed after approximately 8 hours of practice, and after 10 hours all subjects typed between 41% and 73% of their two-handed speed, ranging from 24 to 43 words per minute.

The Half-Qwerty keyboard is unique among solutions to the mobile and handheld text entry problem, because the keyboard is small, familiar to users, supports fairly rapid text entry, and has some significant applications. There are many industrial jobs that require a worker to enter text with one hand while doing another task with the other. The Half-Qwerty keyboard is also useful in rehabilitation situations where a user has lost the use of one hand. In both cases, software can be installed on a regular desktop computer that enables Half-Qwerty functionality. Recently, a small stand-alone version for handheld devices has been introduced.

Although lugging around a full-sized Qwerty keyboard to use with the PDA in one’s shirt pocket seems ridiculous, collapsible Qwerty keyboards allow users to do just that. In 1999, Think Outside Inc. (Carlsbad, Ca; http://www.thinkoutside.com/) released the Stowaway, a full-size Qwerty keyboard that collapses to a 91 × 130 × 20 mm volume. Originally released as a third-party add-on for the Palm, the Stowaway was later adopted by Palm Computing becoming the Palm Portable Keyboard. Think Outside also produce collapsible keyboards for other families of PDAs.
1.5.1.3 Three-key and five-key text entry

By way of introduction to five-key text input, we mention the date stamp method (also known as the three-key text input method). This method can be implemented using very limited hardware: technology to display at least one character, two buttons (or a wheel) to scroll through the alphabet, and an enter key. It is called the date stamp method, because, similar to a date stamp, the desired character is selected by rotating through the character set. Video arcade games often use this technique for players to enter their name when they achieve a high score. The technique is also common for entering text into some electronic musical instruments. Although the three-key method is reasonable for entering small amounts of text into devices with a simple interface, the method is rather slow, about 5 – 10 words per minute according to a recent experiment (MacKenzie 2002b).

Five-key text entry uses an interface with four cursor keys (up, down, left, and right) and an enter key. See Figure 10. The alphabet, number, and symbol characters are presented on a LCD display with typically three to five rows and ten to twenty columns, and the five keys are used to move a cursor to select one letter at a time.
The characters are presented in alphabetic order or in the familiar Qwerty arrangement. The five-key input method is typically used on very small devices like the recent generation of pagers, which only have enough space for a small LCD screen and five keys. An example of one such device is the AccessLink II pager from Glenayre Technologies Inc. (Charlotte, NC; http://www.glenayre.com/).

The main problem with the five-key method is that many key presses are required to move between characters and this significantly retards text input. In view of this, Bellman and MacKenzie (1998) devised a technique known as fluctuating optimal character layout, or FOCL. The idea is that since the input device knows the last character the user has entered, it can subsequently present the characters ordered so as to place the most likely characters closer to the cursor’s home position. Characters are rearranged after each character entered so as to minimise the number of cursor movements to select the most likely next character. They show that the average KSPC can be reduced by over 50%, from just over 4 KSPC for the alpha layout to less than 2 KSPC using FOCL.

Bellman and MacKenzie (1998) report the results of an exploratory study comparing FOCL to the five-key input method using the Qwerty arrangement of letters. Their study with ten participants found that after ten sessions of fifteen minutes each there was no statistically significant difference between the average text entry speeds nor accuracies. The average speed they report for both Qwerty and FOCL is 10 wpm. Although the study was longitudinal in nature, evidently subjects did not have enough exposure to FOCL to approach their maximum text entry speeds. Although fewer keystrokes were required to enter each character, more visual scan time was required to find the next character. In short, as with many other optimised text entry methods, the advantage of the input technique is not realised until users invest considerable time to become familiar with the new technology.
The three-key, five-key, and FOCL text input techniques all require the user’s attention on the screen to scroll around and select characters. Therefore, text creation is a two FOA task with these techniques. Single-handed input is possible with all of these input techniques.

1.5.1.4 Other small keyboards

Some researchers have proposed alternatives to the telephone or Qwerty key arrangements. The single hand key card or SHK is a small card with a keyboard and joystick proposed by Sugimoto and Takahashi (1996). The SHK is held in one hand, pinned between the palm and the thumb in such a way that the four fingers of the hand can manipulate the keyboard and joystick on the top face of the device. SHK is a small keyboard with multiple characters on each key. It employs disambiguation technology. The keyboard arrangement of SHK appears in Figure 11. The joystick and three function keys appear in a row above the keyboard on the device (not shown). The AR key in Figure 11 toggles through the word possibilities generated by the ambiguity resolution feature.

<table>
<thead>
<tr>
<th>P</th>
<th>N</th>
<th>G</th>
<th>T</th>
<th>C</th>
<th>R</th>
<th>Z</th>
<th>K</th>
<th>W</th>
<th>J</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>E</td>
<td>H</td>
<td>I</td>
<td>S</td>
<td>O</td>
<td>AR</td>
<td></td>
<td></td>
<td></td>
<td>Enter</td>
</tr>
<tr>
<td>U</td>
<td>D</td>
<td>X</td>
<td>F</td>
<td>Y</td>
<td>M</td>
<td>V</td>
<td>L</td>
<td>Q</td>
<td>B</td>
<td>Space</td>
</tr>
</tbody>
</table>

Figure 11 - The SHK device (from Sugimoto et al. 1996)

Sugimoto and Takahashi report that the keys were arranged so as to reduce the average motion of the fingers, although they do not report details of how they arrived at this arrangement, and they have not published an evaluation of their device. Once the arrangement is learned by the user, the device could support single FOA text creation, and single-handed text input.
Another important class of keyboard is chording keyboards, where text is entered by pressing multiple keys simultaneously. Because multiple keys may be pressed, fewer keys are needed on a chord keyboard (resulting in smaller devices) and chords not being used for entering single letters can be used to enter words. The Twiddler by HandyKey Inc. (Mt. Sinai, NY; http://www.handykey.com/) is a chord keyboard that is popular with researchers in the wearable computing, and ubiquitous computing fields. The Twiddler is operated with one hand and has four mode keys depressed by the thumb (number, alt, control, and shift), and twelve keys for the fingers. The Twiddler keyboard appears in Figure 12, notice that the layout of the characters is somewhat alphabetical. However, the Twiddler is user configurable; the user may change the characters (or words) entered by each chord (the keyboard appearing in Figure 12 is the recommended layout), and other character mappings for the chords have been proposed for the Twiddler which are claimed to map common characters to easier chords. The Twiddler also has chords defined for common small English words and parts of words (e.g., “the”, “and”, “ion”, and “ing”, etc.).

The Twiddler is intended to be used with one hand (zero FOA, once the chords have been learned), and anecdotal reports of typing speed as fast as 50 wpm have been reported (Hjelm, Tan, Fabry, Fanchon and Reichert 1996) (also see http://lcs.www.media.mit.edu/projects/wearable/keyboards.html).

The Twiddler is by no means the only chord keyboard, for example, the Bat by Infogrip (Ventura, CA; http://www.infogrip.com/), and MonoManus by ElmEntry Enterprises (Minneapolis, MN; http://www.hankes.com/eee/), and many others. However most of these keyboards interface to desktop computers, and are not specifically for mobile computing platforms. For example, the Twiddler interfaces to the Palm only if the Happy Hacking Cradle (by PFU America Inc; San Jose, CA; http://www.pfuca.com/) is used (available separately).
Letters with a white background are entered by pressing the key by itself. Letters with a light grey background are entered by pressing the key and the E key simultaneously. Letters with a dark grey background are entered by pressing the key and the A key simultaneously.

1.5.2 Stylus-based text entry

Stylus-based text entry uses a pen-shaped pointing device to select characters through tapping or gestures. Our discussions here are limited to stylus input, but there are several related examples of research in mobile text entry using finger or touch input, wherein the user’s finger is used instead of a stylus (e.g., Enns and MacKenzie 1998; Fukumoto and Suenaga 1994; Goldstein, Book, Alsio and Tessa 1999; Sears 1994). All of the stylus-based text entry techniques require two hands, unless the user can support the device on a table while using it.

1.5.2.1 Traditional handwriting recognition

Handwriting recognition was once touted as the solution for mobile text entry, but as already noted, early systems received considerable bad press. To be fair, handwriting recognition is a difficult problem, and the technology has improved.
since the early days. There are two problems that handwriting recognisers must solve, segmentation and recognition. The input to a recogniser is a series of ink trails, with each stored as a set of digitised points representing the stylus travel between pen-down and pen-up actions. Segmentation is the process of determining which segments are parts of which characters. With the goal of supporting “natural handwriting”, input is often a mixture of block printing and cursive handwriting. As one might imagine, segmenting the strokes in the scrawl of a sloppy user is very difficult indeed. One way to reduce the complexity is to constrain input, for example, to support block printed characters only. Entry like this is by no means “natural”, however. Generally, the more relaxed the constraints, the more difficult the segmentation and recognition process; recognition accuracy usually suffers. To compensate, recognisers are made more complex and, unfortunately, require more memory. See Tapert, Suen, and Wakahara (1990) for a detailed survey of recognition techniques and technologies.

One obstacle for recognition-based technologies is high user expectations. LaLomia (1994) reports that users are willing to accept a recognition error rate of only 3% (a 97% recognition rate), although Frankish (1995) concludes that users will accept higher error rates depending upon the text editing task. Several researchers have published studies evaluating or comparing the recognition rate of various recognition systems. Chang and MacKenzie report a recognition rate of 87% - 93% for two recognisers (Chang et al. 1994; MacKenzie and Chang 1999a). Wolf et al. (1991) report a recognition rate of 88% - 93%. Santos et al. (1992) report a novice recognition rate of 57%, although this improved to 97% after three hours of practice. These studies suggest that recognition technology is close to matching user expectations for expert users, but that novices may be discouraged by their initial experiences. Perhaps the acid test, an observation suggesting that handwriting recognition does not yet perform adequately, is that there are no mobile consumer products in the market today where natural handwriting recognition is the sole text
input method. The products that do support stylus-based text input work with constraints or stylised alphabets (see below).

There is an important observation that can be made about text entry speed and handwriting recognition. In Gibbs’ (1993) summary of thirteen recognisers, the recognition speed of the systems was at least 4 cps, which translates into 48 wpm. However, human hand printing speed is typically on the order of 15 wpm (Card et al. 1983; Devoe 1967; Van Cott et al. 1972). In other words, speed is a function of human limitations, not machine limitations. Even with perfect recognition, therefore, entry rates can never reach those of, for example, touch typing.

1.5.2.2 Stroke-based text input

Unistrokes is a stylised single-stroke alphabet developed by Goldberg and Richardson at the Xerox Palo Alto Research Center (Goldberg and Richardson 1993). At the time of the invention (1993) the handwriting recognition technology was not in good stead with users, as the problems noted above were rampant in existing products. To address these, Goldberg and Richardson developed a simplified set of strokes that is both easier for software to recognise, and quicker for users to write. The Unistrokes alphabet appears in Figure 13.

![Unistrokes Alphabet](image)

Figure 13 - The Unistroke alphabet (from Goldberg et al. 1993)
The name *Unistrokes* describes the most significant simplification that Goldberg and Richardson made – each letter is written with a single stroke. This greatly simplifies recognition, as the segmentation problem is essentially eliminated. The strokes are so simple that users can write *Unistrokes* without watching the stylus. Goldberg and Richardson observed that *Unistrokes* afford what they termed heads-up text entry (i.e., reduced FOA). The *Unistrokes* alphabet does not contain numbers, punctuation, or symbolic characters, although the original publication (Goldberg et al. 1993) suggests ways of supporting these, for example, using a dedicated stroke as a mode shift.

Although a comparative study of *Unistrokes* has never been undertaken, some experimental results are given (Goldberg et al. 1993). Ignoring errors, a text entry rate of 2.8 cps (34 wpm) was reported.

Although an interesting and promising idea, *Unistrokes* did not catch on, and the most likely reason is that the strokes are not similar enough to regular handwritten or printed letters – the strokes had to be learned. Palm Inc., designed a single stroke system called *Graffiti* that is used in their *Palm* product. *Graffiti* has been credited as a significant reason for the commercial success of the *Palm* (Blickenstorfer 1995). The *Graffiti* alphabet appears in Figure 14.

*Graffiti* has strokes for punctuation, numbers, symbolic characters, and mode switches (capital versus lowercase). (These are omitted from Figure 14 for brevity.) Capital and lowercase characters are supported with mode switching, accomplished by entering a dedicated stroke. Basic editing (backspace) is also supported with a special stroke.

---

9 A stroke begins when the stylus first touches the tablet, and ends when the stylus is lifted from the tablet. Any amount of stylus motion, while remaining in contact with the tablet, is considered part of a single stroke.
The great advantage that Graffiti has over Unistrokes is its similarity to normal hand-printed characters. MacKenzie and Zhang (1997) performed a study of the immediate usability of Graffiti. They observed that 79% of the Graffiti strokes match letters of the Roman alphabet. Under experimental conditions they measured the accuracy with which subjects could enter the alphabet following one minute of studying the Graffiti reference chart, following five minutes of practicing with Graffiti, and following a one week lapse with no intervening practice. The accuracies they reported were very high, 86%, 97%, and 97%, respectively.

Isokoski (1999) presents a single stroke alphabet that can be entered using a wide range of pointing devices. He observed that the easiest motions to make with pointing devices are the four primary compass directions: up, down, left, and right. A design objective was finding the optimal mapping between the four directional strokes and the characters of the alphabet (more frequent characters should have shorter strokes). He called the result minimal device-independent text input method, or MDTIM. The MDTIM alphabet appears in Figure 15.

Isokoski evaluated his single stroke alphabet with a variety of pointing devices. The measured average text entry speed using a touchpad was 7.5 wpm. The study was not longitudinal and the subjects were still showing improvement at the end of the trials. The MDTIM alphabet suffers from the same affliction as Unistrokes and many other text input methods – the alphabet is not familiar to the average user,
and considerable investment is required to attain fast entry speeds. However, Isokoski’s results do indicate that the *MDTIM* alphabet is indeed device independent.

![MDTIM alphabet](image)

**Figure 15** - The *MDTIM* alphabet (from Isokoski 1999)

Until recently *Windows CE* devices were without a similar easy-to-learn handwriting recognition technology. This changed in 1998 when Microsoft licensed *Jot* from Communication Intelligence Corp. *Jot* recognizes many of the *Graffiti* strokes and a number of alternative strokes similar to normal handwriting and printing as well. The *Jot* alphabet appears in Figure 16.

*Jot* also includes strokes for numbers, symbolic characters, and common editing functions (omitted for brevity). The different cases (capital versus lowercase) are selected by where the user writes the stroke on the touch screen of the device. *Jot* also allows some customisation – users can indicate writing preferences for some characters.

All the alphabets just described have the potential to support single FOA text entry, once the user is familiar with the stylised alphabet.
1.5.2.3 Gesture-based text input

Gestures are informal motions used to communicate. We have classified the text entry methods in this section as gestural because of their informality and fluidity. Character recognition-based and soft keyboard-based input techniques have fixed characters that are entered in a certain way, or the stylus must be tapped in a certain location to select characters for input. Gesture-based text input technologies do not have a fixed set of strokes that a recogniser turns into characters; gestural text input methods have a framework in which informal stylus motions are interpreted as characters.

An example of this is Cirrin, a technology invented by Mankoff and Abowd (1998).
The letters of the alphabet are arranged along the perimeter of a circle. Figure 17 shows the word “cirrin” written on the Cirrin interface. Characters are selected by moving into and out of the appropriate segment of the circle. Mankoff and Abowd choose the annular arrangement and the order of the letters around the annulus to minimise the distance between likely consecutive characters. In Figure 17, notice how the final two letters, “in” are selected, the stylus can be moved directly from one letter into the neighbouring letter.

![Figure 17 - The Cirrin interface (from Mankoff et al. 1998)](image)

The pointing device motion necessary to enter the word “cirrin” has been indicated.

Cirrin is not a “heads up” text input method; users must attend to the interface when entering text. As presented, only alphabetic characters are supported. A space is entered by lifting the stylus, and punctuation and mode shifts are accomplished by using an auxiliary technique, such as a keys operated by the non-dominant hand.

Mankoff and Abowd did not report a user evaluation in their publication, however
they state that *Cirrin* “is about as fast as existing pen entry systems” (1998, page 214) but no indication is given of specifically what pen entry systems they were comparing *Cirrin* to.

*Quikwriting* is an input technology described by Perlin (1998). The idea is to have a 3 × 3 grid where characters are entered with strokes that begin in the centre “home” position and move through one to three adjoining positions, returning back to the home position. Figure 18 illustrates the *Quikwriting* lowercase menu. *Quikwriting* has similar displays and modes for numbers, capitals, and symbols (omitted for brevity). The symbols in the top centre and bottom centre positions represent the different modes. Letters that occur more frequently in English are given the shortest strokes. For example, “i” in Figure 18 is selected by moving into the bottom right position and then returning back to the home position. Infrequent letters have longer strokes (“k” requires a move into the upper left position, then to the upper right position, and finally back to home). There is an online demonstration available (http://www.mrl.nyu.edu/perlin/demos/quikwriting.html). *Quikwriting*, like *Cirrin*, requires the user to look at the interface and so is a two FOA interface, if users correct errors as they go.

At the time of Perlin’s publication (1998) a user evaluation had not been performed, although he writes that users familiar with *Graffiti* find *Quikwriting* about three times faster.
The pointing device motion necessary to enter the word “quik” as been indicated.

To the uninitiated Quikwriting can be confusing. Two gestures are necessary to enter a letter, the first selects a group of characters, and the second selects the individual character within the group.

Consider the upper-left corner. By moving into the upper-left corner and directly back to the home position, the letter “a” is entered. By moving into the upper-left corner, then to the left-middle section, and then to the home position, the letter “m” is selected. Finally, moving into the upper-left corner, then to the lower-left corner, and then home, enters the letter “q”. Similarly, moving to the upper-left corner and then to the top-centre or upper-right sections, selects the letters “s” or “k” respectively.

Another gestural text input technology is T-Cube, described by Venolia (1994). T-Cube is similar to a two-tier pie menu system. Figure 19 shows the pie menu structure of T-Cube. The location where the stylus is first placed indicates which of the pie menus the user will select from. The user places the stylus within one of nine starting positions (arranged in a 3 × 3 grid – see Figure 19a), this indicates which of the submenus (which appear in Figure 19b) the user shall choose the next character from. One of the eight characters in the submenu is chosen by flicking the stylus in one of the eight compass directions (up, down, left, right, and the four diagonal directions). The interface does not display the pie menus (Figure 19b), unless the user hesitates. T-Cube includes numbers, many symbol characters, and
basic backspace editing. Like the other gestural input techniques, \textit{T-Cube} requires the attention of the user, making standard text entry a two FOA task.

Venolia presents the results of a user study of \textit{T-Cube} (1994) indicating that reasonably fast text entry can be achieved; one of his subjects achieved a rate of 106 characters per minute (21 wpm). However, he also acknowledges that the interface is difficult to learn.

![Figure 19 - T-Cube pie menu structure (from Venolia et al. 1994)](image)

(a) 1st level menu (b) 2nd level menus
1.5.2.4 Soft keyboards

A soft keyboard is a keyboard implemented on a display with built-in digitising technology. Text entry is performed by tapping on keys with a stylus. Eyes-free entry is not possible. Advantages of soft keyboards include simplicity and efficient use of space, because the system only needs to display the soft keyboard when text is being entered. When no text entry is occurring, the soft keyboard disappears thus freeing screen space for other purposes.

Different arrangements of the keys of a soft keyboard affect both the speed and accuracy with which text can be entered. MacKenzie et al. (1994a) report a text entry task comparing a Qwerty soft-keyboard, an alphabetically-arranged soft keyboard, and hand printing. The Qwerty soft keyboard was both faster and more accurate than hand printing (see Figure 3 on page 19). The effect that key arrangement has on soft keyboard performance is discussed in detail in Chapter 2, and optimisation of key arrangements is discussed in Chapter 4.

1.5.3 Predictive input techniques

One early predictive input technology is the Reactive Keyboard (Darragh 1988; Darragh, Witten and James 1990; Darragh and Witten 1991). The Reactive Keyboard monitors what a user enters and presents text predictions that the user can choose from using the mouse. The predictions are generated by finding longest matching sub-strings in the previously entered text. The Reactive Keyboard adapts to users’ input and hence is not limited to a static set of words or phrases. Other related work is hereby cited (Jakobsson 1986; Masui and Nakayama 1994; Raita and Teuhola 1987).

POBox (Masui 1998; Masui 1999) is predictive input technology that allows users to enter part of a word and then search for similar words by spelling, pronunciation, or shape (for pictograph-based languages). It is not limited to alphabetic languages.
POBox uses a static database coupled with another primary input technique, such as a soft keyboard or telephone keypad. Search results appear on the screen as the user types. A tap or key press selects the desired word. When embedded in a mobile phone, text entry via the multi-key method yields a list of search results that the user scrolls through using a scroll wheel on the side of the device.

Lewis and colleagues (Lewis 1999; Lewis et al. 1999a) experimented with a predictive soft keyboard technology for extremely limited screen sizes. Their system presents the user with keys for the six to eight most likely characters, and an “other” key revealing the rest of the alphabet. Lewis reports (1999) text entry speeds for a Qwerty soft keyboard, his predictive keyboard, and handwriting (as a control condition) at 14 wpm, 6 wpm, and 22 wpm, respectively. The speed Lewis reports for soft keyboard entry is approximately half that reported by others (e.g., MacKenzie et al. 1994a). Lewis observes that the uncertain arrangement of the keys of the predictive keyboard significantly hindered performance.

1.6 Conclusions

There are many text entry methods available to designers of mobile systems, and without a doubt more are forthcoming. However, deciding which is best for an application is difficult, in part, because of the lack of publications giving empirically measured text entry speeds and accuracies. This chapter has brought together many of the techniques in use or under investigation in this exciting area of mobile computing. The result is a snapshot of the current state of the art in mobile text entry.

Movement and language are omnipresent in human-computer interaction. We have hinted that Fitts’ law and a language corpus can work together in exploring, a priori, design alternatives for stylus input on soft keyboards or single-finger input on small physical keyboards. These ideas will be explored in detail in the following
Additionally, we have examined many issues in methodology and evaluation, and have identified factors, such as: text copy versus text creation tasks, foci of attention, novice versus expert performance, confusion matrices, and the speed/accuracy trade-off, that impact user performance. Clearly, evaluation is critical, and it is by no means simple. A number of issues are particularly tricky such as the measurement and treatment of errors and the types of tasks used in text entry studies. These and other topics, such as internationalisation, are the subject of ongoing and future work.
Chapter 2
Modelling Stylus Typing$^{10}$

In this chapter a theoretical model is presented to predict upper-bound and lower-bound text-entry rates using a stylus to tap on a soft Qwerty keyboard. The model is based on the Hick-Hyman law for choice reaction time, Fitts’ law for rapid aimed movements, and linguistic tables for the relative frequencies of letter-pairs, or digrams, in common English. The model’s importance lies not only in the predictions provided, but in its characterization of text-entry tasks using keyboards. Whereas previous studies only use frequency probabilities of the $26 \times 26$ digrams in the Roman alphabet, our model accommodates the space bar – the most common character in typing tasks. Using a very large linguistic table that decomposes digrams by position within words, we established start-of-word (space-letter) and end-of-word (letter-space) probabilities and worked from a $27 \times 27$ digram table. The model predicts a typing rate of 8.9 wpm for novices unfamiliar with the Qwerty keyboard, and 30.1 wpm for experts. Comparisons are drawn with empirical studies using a stylus and other forms of text entry.

2.1 Introduction

The efficiency of different manual data entry methods has been of interest since the field of information processing began. Entry methods such as typing, printing, hand-writing, or selecting with a mouse have been investigated extensively (e.g. Card et al. 1983; Van Cott et al. 1972).

---

With the advent of pen-based computers, there is renewed interest in printing or hand-writing using a pen or stylus as a form of input (Gibbs 1993). In fact, the lure of cursive handwritten input is so strong that alternate forms of entry are often ignored. In particular, we feel an entry method worthy of serious evaluation is tapping with a stylus on a graphic representation of a Qwerty keyboard. We call this a “soft keyboard”.

In the following sections, we will attempt to answer the following question: “How fast can one type using a stylus to tap on a soft keyboard”. We have calculated a lower-bound, which is the typing speed expected for walk-up, novice users unfamiliar with the Qwerty layout; and an upper-bound, a rate attainable after considerable practice on a Qwerty soft keyboard. Our model is useful, not only due to the predictions provided, but also because of its behavioural description of text-entry tasks using various forms of keyboards.

To answer the question posed above, we draw on Fitts’ law for rapid aimed movements (Fitts 1954), the Hick-Hyman law for choice reaction time (Hick 1952; Hyman 1953), and linguistics tables for characterizing the text-entry task. Work by Epps (1986), Kerr (1977), MacKenzie, Sellen, and Buxton (1991) and others has demonstrated that Fitts’ law is applicable to the movement of a stylus as an input device. By extension, Fitts’ law is applicable when stylus movements are between “keys” on a Qwerty keyboard simulated on the display of a pen-based computer.

The applicability of the Hick-Hyman law has been established in numerous choice reaction tasks, such as pressing buttons in response to lights. Welford (1968, p. 64) notes that the act of sorting cards can be modelled as two separate acts: a movement time and a visual scan time modelled by the Hick-Hyman law. More recently, Landauer and Nachbar (1985) showed that a computer input task requiring a choice followed by a movement-plus-selection can be modelled using the
Hick-Hyman law for the choice component of the task followed by Fitts’ law for the movement-plus-selection component of the task.

These simple and intuitive relationships suggest that we can predict novice performance by adding a visual scan time to the movement component of each entry. As the novice gains familiarity with the Qwerty layout, the visual scan time diminishes (eventually to zero) and performance improves to expert levels, whereby the movement time for each entry is the sole component of the task. Although novice-to-expert migration may reveal an improvement in the motor component of the task, this is considered negligible in comparison to the reduction in the visual scan time. Evidence of this has been noted in novice-to-expert transitions for shorthand, wherein improvements follow from reductions in inter-stroke hesitations rather than reductions in stroking time (Gregg, Leslie and Zoubek 1972).

To express entry rate in “words per minute”, we assume that the task is text entry of common English. Linguistic tables compiled from large samples of representative English assist in characterizing the text-entry task. Briefly, a 26-character alphabet contains $26 \times 26$ letter-pairs, or digrams. For each digram, we can predict the movement time to enter the second letter given the first, and this prediction is weighted by the probability of occurrence of the digram. We will develop and refine this concept later, particularly with respect to the role of the space bar (which is of little relevance to linguists).

After deriving our upper-bound and lower-bound models, we will compare our predictions against data from empirical studies using a stylus and other forms of text entry. Weaknesses in our model are identified and assessed; and, in conclusion, qualitative comparisons are drawn between stylus-tapping and other forms of text entry.
2.2 Model derivation

2.2.1 Characterizing the text-entry task

Before modelling a task, we must define it. We characterize “text entry of common English” using tables for single-letter and letter-pair (digram) frequency counts in English. Our approach is similar to the study of touch typing speeds for different keyboards presented by Card et al. (1983), who used a table by Underwood and Shultz (1960). For reasons explained shortly, we used a different but similar table by Mayzner and Tresselt (1965). Mayzner and Tresselt’s table was compiled from 100 samples of 200 words. Each sample was drawn at random from a variety of sources, including newspapers, magazines, and fiction and non-fiction books. Mayzner and Tresselt established that the relative frequencies of letters in their sample correlated highly with the same frequencies in Underwood and Schulz’s sample. And so, the claim that the sample is representative of English is made, notwithstanding caveats for non-text input, cultural bias, etc.

Mayzner and Tresselt’s (1965) table, which spans 18 pages, is particularly useful because digram positions within words were also provided, unlike Underwood and Schulz’s (1960) table. This allowed us to identify digrams at the beginning and ending of words, and therefore, to extend the $26 \times 26$ digrams to $27 \times 27$ digrams. The “alphabet” in our model consists of 26 letters plus the space character. Table 2 gives the frequency counts of the $27 \times 27$ digrams used to characterize the text-entry task for our model.\(^{11}\) The data in Table 2 were entered into a large spreadsheet with $27 \times 27$ rows. Columns were added incrementally to build our model, as described herein.

\(^{11}\) A careful inspection of Table 2 reveals slight discrepancies in the row and column totals. These are due to small errors noted by Mayzner and Tresselt (1965) and attributed to human error in tabulating the original data.
The core 26 × 26 digrams are from Mayzner and Tresselt’s (1965) Table 2. The space digrams (shaded) were compiled from Mayzner and Tresselt’s frequency counts for start-of-word and end-of-word digrams.

Table 2 - 27 × 27 digrams for the text-entry task

The core 26 × 26 digrams are from Mayzner and Tresselt’s (1965) table 2. The space digrams (shaded) were compiled from Mayzner and Tresselt’s frequency counts for start-of-word and end-of-word digrams.
The importance of the space character is illustrated as follows. Mayzner and Tresselt's sample contained 20,000 words with a total of 87,199 characters. If Table 2 was constructed only showing $26 \times 26$ digrams, then the most-prominent letter would be “e” with a frequency probability of $11,612 / 87,199 = 0.133$ and the most-prominent digram would be “h-e” with a frequency probability of $3155 / 87,155 = 0.0362$. However, by adding 20,000 spaces to the sample (one space per word), the frequency probability of “e” drops to $11,620 / 107,199 = 0.108$ and the space character becomes the most prominent with a frequency probability of $20,000 / 107,199 = 0.187$. The most prominent digram becomes “e-space” with a frequency probability of $4904 / 107,199 = 0.0457$. Table 3 illustrates the calculations for the ten most-frequent digrams in the sample.

<table>
<thead>
<tr>
<th>Digram</th>
<th>Count</th>
<th>Probability‡</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-space</td>
<td>4904</td>
<td>0.0457</td>
</tr>
<tr>
<td>space-t</td>
<td>3912</td>
<td>0.0365</td>
</tr>
<tr>
<td>t-h</td>
<td>3774</td>
<td>0.0352</td>
</tr>
<tr>
<td>h-e</td>
<td>3155</td>
<td>0.0294</td>
</tr>
<tr>
<td>d-space</td>
<td>2627</td>
<td>0.0245</td>
</tr>
<tr>
<td>s-space</td>
<td>2348</td>
<td>0.0219</td>
</tr>
<tr>
<td>t-space</td>
<td>2228</td>
<td>0.0208</td>
</tr>
<tr>
<td>space-a</td>
<td>1882</td>
<td>0.0176</td>
</tr>
<tr>
<td>space-w</td>
<td>1787</td>
<td>0.0167</td>
</tr>
<tr>
<td>a-n</td>
<td>1576</td>
<td>0.0147</td>
</tr>
</tbody>
</table>

‡ Probability = Count / 107,199

### 2.2.2 The upper-bound
For each of the $27 \times 27$ digrams in Table 2, we used Fitts’ law to predict the movement time to select the second key, given the first. Fitts’ law is expressed as
\[ MT = a + b \times ID \]  \hspace{1cm} (1)

where \( MT \) is the predicted movement time, and \( ID \) is the index of difficulty of the task being modelled. The coefficients \( a \) and \( b \) are constants obtained through linear regression. The index of difficulty (with units of \( \text{bits} \)), is defined using the Shannon formulation (MacKenzie 1992) as

\[ ID = \log_2 \left( \frac{A}{W} + 1 \right) \]  \hspace{1cm} (2)

\( A \) and \( W \) are physical parameters in the task being modelled. Specifically, \( A \) is the amplitude of the movement and \( W \) is the width of the target (ideally, measured along the direction of motion).

We are interested in movements from an initial character on the keyboard \( i \), to a subsequent character \( j \). Combining Equation 1 and 2 results in

\[ MT_{ij} = a + b \times \log_2 \left( \frac{A_{ij}}{W} + 1 \right) \]  \hspace{1cm} (3)

where the variable \( A_{ij} \), the amplitude, represents the distance from the centre of the \( i \) key to the centre of the \( j \) key across the keyboard. The variable \( W \), the width, represents the width of the target key, which is the same for all alpha keys (see below). The variable \( MT_{ij} \) is the predicted time to move from the \( i \) key to the \( j \) key.

Notice that amplitude and width are measured in the same units and form a ratio in Equation 3. This implies that a Qwerty keyboard will yield the same movement time regardless of its size, so long as the same relative scale is maintained. This will be true only within limits. For example, as keyboard size decreases, precision will be compromised as size of the targets diminishes. As keyboard size increases, movement times will increase (and precision may decrease) because forearm motion
will be required where wrist-only motion was previously used. This observation has been confirmed by MacKenzie and Zhang (2001b).

There exists a problem with the predicted movement time as defined above. Fitts’ law is only valid for movements from a start location to a target some non-zero distance away. When entering a character twice in succession, the starting character \( i \) is the same as the target character \( j \), and the amplitude \( A \) is zero. Fitts’ law does not apply. To circumvent this difficulty we modify our model:

\[
MT_{ij} = \begin{cases} 
  a + b \times \log_2 \left( \frac{A_{ij}}{W} + 1 \right) & \text{if } i \neq j \\
  MT_{\text{Repeat}} & \text{if } i = j
\end{cases}
\]  

(4)

where \( MT_{\text{Repeat}} \) is the time to activate a key a second time after it has already been activated. A discussion of how a value for \( MT_{\text{Repeat}} \) was obtained is given later.

To weight each \( MT_{ij} \), we define \( P_{ij} \) as the probability of occurrence of the digram \( ij \), (the character \( i \) followed by the character \( j \)). Note that the sum of all probabilities must be one; that is

\[
\sum_{i, j \in \text{Alphabet}} P_{ij} = 1.
\]

(5)

We define

\[
\overline{MT} = \sum_{i, j \in \text{Alphabet}} (P_{ij} \times MT_{ij})
\]

(6)

where \( \overline{MT} \) is the mean time between keys with an alphabet of 27 symbols, as noted earlier. \( \overline{MT} \) is a minimum because it is based solely on the time to move between keys and assumes no pause for thinking or finding keys.
Taking the reciprocal of the average movement time yields the average number of characters per second, which is easily transformed into words per minute:

\[
CPS_{\text{max}} = \frac{1}{MT} \\
WPM_{\text{max}} = \frac{CPS_{\text{max}} \times 60}{5}
\]  \hspace{1cm} (7)

Note that typists define a word as five characters, including spaces and punctuation; also, there are 60 seconds in a minute. Equation 7 is our upper-bound model.

2.2.3 The lower-bound

To compute a lower-bound typing rate, we use the Hick-Hyman law to add a visual scan time onto each entry. The Hick-Hyman law is expressed as

\[
RT = a' + b' \times \log_2(n)
\]  \hspace{1cm} (8)

where \(n\) is the number of items to choose from, and \(RT\) is the choice reaction time, or the time required to make a choice among \(n\) items. The coefficients \(a'\) and \(b'\) are slope and intercept constants, similar to the coefficients \(a\) and \(b\) in Fitts’ law.

To calculate the lower-bound, we simply add the choice reaction time to the movement time per character:

\[
CPS_{\text{min}} = \frac{1}{MT + RT} \\
WPM_{\text{min}} = \frac{CPS_{\text{min}} \times 60}{5}
\]  \hspace{1cm} (9)

The minimum typing speed considers not only the movement time between keys, but the time it takes on average to find each key by someone who is unfamiliar with the
Qwerty layout. Equation 9 is our lower-bound model.

2.2.4 Coefficients in the model

The derivation presented thus far, culminating in Equation 7 and 9, expresses our upper-bound and lower-bound models in general terms, without assigning numerical coefficients. Predictions follow once the coefficients in the models are assigned values.

2.2.4.1 The Fitts’ law intercept, $a$

The form of Fitts’ law expressed by Equation 1 contains coefficients for the intercept, $a$, and slope, $b$. It is generally accepted that the intercept should be zero. If the intercept were positive then the movement time for a task with $ID = 0$ is not zero (see Equation 1). This seems unreasonable. Similarly, with a negative intercept, a task of zero difficulty has a negative predicted movement time, which also seems unreasonable. Fitts’ (1954) original relationship excluded the intercept altogether. For these reasons, we used $a = 0$.

2.2.4.2 The Fitts’ law slope, $b$

Choosing an appropriate value for the slope is a problem. The reciprocal of the slope is called the bandwidth, and its units are bits per second (bps). Fitts’ (1954) original work with a stylus reported bandwidth in the range 9.5 to 11.5 bps. In later work, Fitts and Peterson (1964) suggested a range of 14 to 22 bps. In our opinion, these values are too high. (A justification appears later.) The study by MacKenzie et al. (1991) measured the bandwidth for pointing tasks using a stylus as a computer input device and found a rate of 4.9 bps. The pointing task in the latter experiment was similar to the form of typing we are investigating.

Many reasons for the lack of consensus on bandwidth relate to diverse and inconsistent experimental methods, as identified by MacKenzie (1992). The
calculations that follow were performed in duplicate using 14 bps, as suggested by Fitts and Peterson (1964), and 4.9 bps, as suggested by MacKenzie et al. (1991).

2.2.4.3 Key distances, $A_{ij}$

We used a facsimile of a Qwerty keyboard taken from IBM hardware documentation (IBM Corporation, 1984, pp. 4-6) to find the inter-key distances. Using an arbitrary origin, $x$ and $y$ coordinates were assigned to each key and inter-key distances were calculated using the Pythagorean identity. As an example, the distance across the keyboard from Q to P in our template was 18.9 cm.

As noted previously the actual size of the keyboard is irrelevant, within limits. It is assumed that the simulated Qwerty keyboard matches the proportions of the keyboard taken from the IBM documentation.

The position of the space bar cannot assume a fixed $x$-$y$ coordinate, as with alpha keys, since the space bar spans the width of the keyboard. As illustrated in Figure 20, the distances for the digrams involving spaces depend on three keys, rather than two. For our lower-bound model we assume users do not optimise movements and proceed directly to the space bar at the end of each word (see Figure 20a). For our upper-bound model we assume a strategy that minimizes the total distance of the letter-space-letter sequence; that is, the angle of approach to the space bar is the same as the angle of departure (see Figure 20b).

Instead of computing distances explicitly for each letter-space-letter trigram, we computed each letter-space distance as a weighted average considering all possible letters to follow, and, similarly, computed each space-letter distance as a weighted average considering all possible preceding letters. Although this sounds messy, the formulas, once established, are easily added to the spreadsheet containing the $27 \times 27$ digrams.
Figure 20 - Distances to and from the space bar
(a) Non-optimised movements for novices, (b) Optimised movements for experts.

Note that the space bar of many modern computers does not extend from the edge of the left side of the alphabetic keys to the edge of the right side. In our model, we assume the space bar does span the width of the alpha keys. This is a minor adjustment; however, the assumption simplifies the calculations of letter-space-letter distances. Since the space character is the most common, it is reasonable to assume that a soft keyboard would accommodate the prominence of the space bar.

2.2.4.4 Key width, $W$
Recall that the width, $W$, is measured along the direction of motion. Because the keys of the Qwerty keyboard are rectangular, the magnitude of $W$ differs with the
angle that the stylus approaches the target key. This complicates the model; however, a pragmatic solution (with empirical support, MacKenzie and Buxton 1992) is to set $W$ to the minimum of the height and width of rectangular targets. This applies to all keys including the space bar. Thus, we used $W = 2.12$ cm for the width of the alpha keys and $W = 2.38$ cm for the width (actually, the height) of the space bar.

2.2.4.5 Hick-Hyman law intercept, $a'$

Most interpretations of the Hick-Hyman law associate the intercept, $a'$, with a transport lag resulting from the subjects' immediate reaction to the onset of the stimulus. However, when there is no uncertainty as to when the stimulus signal arrives, as in continuous text-entry, Welford (1968) suggests that the constant, $a'$, is zero. Therefore, we set $a' = 0$.

2.2.4.6 Hick-Hyman law slope, $b'$

As in Fitts' law, the reciprocal of the slope coefficient in the Hick-Hyman law, $b'$, is called the bandwidth and is measured in bits per second. Bandwidth, in this context, is the rate at which humans process choices.

Welford (1968) maintains that for subjects in their twenties using key presses to signal choices, the reciprocal of the slope in the Hick-Hyman law lies in the range 5 to 7 bps. Since we are searching for a lower-bound, we assume that the slowest choice processing speed is appropriate, and set $b' = \frac{1}{5} = 0.2$ seconds per bit. This is also the figure reported by Hick (1952).

With a 27-character alphabet, $n = 27$, so the visual scan time for novices in the lower-bound model is

$$RT = 0.2 \times \log_2(27) = 0.951 \text{ seconds.}$$

(10)
2.2.4.7 Key repeat time, $MT_{\text{Repeat}}$

A small informal experiment was conducted to determine a reasonable value for the mean key repeat time, $MT_{\text{Repeat}}$. Six subjects entered one character repeatedly into an editor for one minute. They used a stylus to tap on a soft keyboard displayed on a Wacom PL-100V digitising display tablet. Every ten seconds a carriage return was entered by the operator from the physical keyboard. (This was done to simplify the subsequent analysis of the number of characters entered in each ten second period.) Because the physical keyboard was used to enter the carriage returns, the participants were not disturbed from their task. Of the 2,347 characters entered by all subjects, the average number per second was 6.52, with a standard deviation of 1.15 characters. No significant difference was noted in the number of characters across the six different ten-second periods. The results indicate that

$$MT_{\text{Repeat}} = \frac{1}{6.52} = 0.153 \text{ seconds.}$$

(11)

This is close to the figure of 140 ms cited by Card et al. (1983, p. 60) for a typist repetitively pushing a key with a finger.

2.3 Results of model derivation

We have now completely defined all equations and coefficients for the upper-bound and lower-bound models. Since the models were developed using a spreadsheet, the results were available to us immediately upon entering the formulas. Table 4 summarizes the results. Using a bandwidth of 4.9 bps, we expect subjects can type (i.e., tap) at about 8.9 wpm without any prior experience with a Qwerty keyboard. With practice, this rate should increase to about 30.1 wpm. Using a bandwidth of 14 bps, the range is from about 11.0 wpm to 84.6 wpm. We feel the latter figure is much too high.


Table 4 - Upper-bound and lower-bound predictions

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Lower-Bound</th>
<th></th>
<th>Upper-Bound</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Characters</td>
<td>Words Per Minute</td>
<td>Characters</td>
<td>Words Per Minute</td>
</tr>
<tr>
<td>4.9 bps</td>
<td>0.74</td>
<td>8.9</td>
<td>2.51</td>
<td>30.1</td>
</tr>
<tr>
<td>14 bps</td>
<td>0.92</td>
<td>10.98</td>
<td>7.05</td>
<td>84.56</td>
</tr>
</tbody>
</table>

2.4 Discussion

Before comparing our results with empirical data from other studies, we present an observation that illustrates why a Fitts' law bandwidth of 14 bps is too high. Consider entering the sequence Z-X, which corresponds to neighbours on a Qwerty keyboard. Using $\frac{1}{b} = 14$ bps, the predicted time to tap X following Z is

$$MT_{ZX} = b \times \log_2 \left( \frac{A_{ZX}}{W} + 1 \right)$$

$$= \frac{1}{14} \times \log_2 \left( \frac{2.12}{2.12} + 1 \right)$$

$$= 0.071$$

$$= 71 \text{ ms}$$

Clearly this is unreasonable because it is less than half the $MT_{Repeat}$ time of 153 ms given earlier. For this and other reasons (see MacKenzie 1992), we feel the bandwidth figure of 4.9 bps is more accurate.

2.4.1 Comparisons with other text entry methods

Figure 21 compares the results of our model with empirical data from several text-entry experiments.
An experiment of stylus-tapping on a soft keyboard is reported in MacKenzie, Nonnecke, McQueen, Riddersma, and Meltz (1994b). Two different keyboard arrangements were used: a Qwerty layout and an alphabetic (ABC) layout. In the latter case, keys were arranged alphabetically in two horizontal rows with a space bar across and below the bottom row (see Figure 27d). The ABC layout was tested because of its good use of screen real estate. Typing speeds for the Qwerty layout ranged from 21 wpm for the first block to 24 wpm for the ninth and last block. With prolonged practice, subjects would, no doubt, improve, perhaps levelling off near 30 wpm, as predicted in our upper-bound model.
MacKenzie and Zhang (1999b) report stylus tapping speeds on a Qwerty soft keyboard ranging from 28 to 40 wpm. They compare these results with a keyboard arrangement that is optimised for stylus tapping (the OPTI I, presented in Chapter 4) that ranged in speed from 17 to 45 wpm in a longitudinal study. That the lower bound of the speed reported for the OPTI I keyboard is 17 wpm is cause for concern. The subjects in this experiment were unfamiliar with the keyboard, and so should have had a lower novice text entry rate. This paper (MacKenzie et al. 1999b) was the first to present evidence that the novice model may not reflect reality.

Although the movement component of our model is inaccurate for the ABC layout (because the key distances were different), it is the visual scan time that dominates for novices. Therefore, the results for the ABC layout, which was unfamiliar to users, merit comparison with our lower-bound model. Typing speeds ranged from 12 wpm for the first block to 14 wpm for the ninth block. The entry rate of 12 wpm is slightly above our model’s lower-bound rate of 8.9 wpm. We attribute this to the following. First, the alphabetic ordering of the ABC layout is not random; in fact, it gives subjects a good sense of where characters are located. This reduces the visual scan time that would occur for a random layout. Second, learning is “immediate”. That is, after the first few entries, subjects are already gaining familiarity with the layout and cease to be novices in the strictest sense. In fact, an experiment to test our lower-bound model would require a completely new and random key assignment after each key stroke. Such an experiment is reported by MacKenzie and Zhang (2001b). They report a rate of 20 wpm for the randomised keyboard, and present observations that suggest that the novice model presented in this chapter is flawed. Novices tend to make complicated motions with their hand while searching for keys on the keyboard. Typical movements include lifting their hand so as to have a clear view of the keyboard, or moving their hand back-and-forth over the keyboard mimicking the scanning motions of their eyes. In any case, it seems that the simple addition of the Hick-Hyman choice reaction time to account for novice unfamiliarity with the keyboard is insufficient.
The closest entry method to stylus-tapping is the use of a soft keyboard on a touchscreen. Gould and colleagues (1990) found entry rates of 12 wpm for experienced users. This rate seems low; however, the measurement included the extra touches to back-up the cursor, correct errors, and the time to review the input text mid-way through a block if subjects forgot the input string. Wiklund and Dumas (1987) found rates of 14 to 18 wpm in a similar experiment. Sears and colleagues (1993) measured entry rates with novice and expert users on two sizes of keyboards. Novice rates were 10 wpm on the small keyboard and 20 wpm on the large keyboard. Expert rates were 21 wpm on the small keyboard and 32 wpm on the large keyboard.

Sears (1991) found a rate of 25.4 wpm using expert subjects for a text-entry task with a touchscreen. The task was similar to that modelled here, since the only characters entered were the 26 alpha keys, a space bar, and a return key. The experiments described by Sears (1991), Sears et al. (1993), and Wiklund and Dumas (1987) allowed subjects to use two hands on the touchscreens, so comparisons with stylus-tapping are limited.

The text-entry method for pen-based computers that receives the most press is hand printing or hand writing with built-in recognition software. A “perfect” recogniser will be transparent to the user; so traditional research into hand printing and writing gives some insight into the best possible scenario for this provocative form of input. Hand printing speeds are well-known to lie in the range of 12 to 23 wpm (Bailey 1989; Card et al. 1983). So, even with highly accurate recognition software, entry speeds will not match experts tapping on a soft keyboard with a stylus. Although cursive handwriting rates can range from 16 wpm (Devoe 1967) to over 30 wpm (Wiklund et al. 1987), recognition errors exceed those for block printing (Wolf and Morrel-Samuels 1987) and therefore exacerbate the problems of automated recognition of input.
Goldberg and Richardson (1993) report that one user attained rates of 33.6 wpm for “peak error-free” input using Unistrokes, however, a formal experiment was not performed. Veniola and Neiberg (1994) reported entry rates from 12 to 21 wpm from an experiment using T-Cube.

Based on the results displayed in Figure 21, our upper-bound prediction of 30 wpm fares quite well as a text-entry technique.

2.4.2 Weaknesses in our model

Every effort was made to make our model comprehensive. Still, the predictions must be considered rough estimates. Although weaknesses in our model are easy to spot, we feel these represent less dominant aspects of the task than those that are accommodated. A few examples are offered.

Our model is limited to a 27-key soft keyboard with 26 alpha keys and a space bar. In practice, soft keyboards must accommodate the full spectra of text entry, including capital and lower-case letters, punctuation symbols, function keys, etc. We limited our model to 27 keys because it made the task easy to define. Implementing the full complement of Qwerty keyboard features in a soft keyboard is easy, but performance predictions are difficult since they are highly task-dependent and they require a much larger table of digrams. Adding the period and comma to our table would be possible only if we undertook a very arduous process of data collection and tabulation similar to that reported by Mayzner and Tresselt (1965). Adding the less common graphic symbols (e.g., ~ | _ { #) to our model is not worth considering, since they are not used in a consistent manner in “common English”.

To obtain capital letters, a shift function can be implemented several ways. The simplest is to include a shift key which is touched once to transpose the next alpha key to uppercase. This method was implemented by Plaisant and Sears (1992) for a touchscreen. Caps-lock may be implemented by a separate caps-lock key or by
double-tapping the shift key. In the latter case, caps-lock ceases upon the next tap of the caps-lock key.

The space bar on a Qwerty keyboard is much larger than the alpha keys and it is the most frequently entered key. Our model’s formula for choice reaction time (Equation 8) is a rough estimate; it assumes all choices are equally probable and are equally presented. Obviously, the space bar poses a problem in predicting visual scan time. Detailed treatments of choice reaction time replace $\log_2(n)$ in Equation 8 with an information metric for the entropy of the distribution of choices. This would tend to reduce $RT$ in our lower-bound model and increase the predicted entry rate for novices.

The predictions are also weakened by the fact that the user’s hand will partially obscure the keyboard after certain key-taps (for example, in the middle or upper-left regions of the keyboard for right-handed people). This has a slight impact on both the $RT$ and $MT$ predictions in the lower-bound model. Other weaknesses have already been noted, such as the validity and accuracy of the Fitts’ law model used to predict the movement time for each entry.

2.4.3 Alternate key layouts

Since our model is based on a spreadsheet in which each key is assigned an $x$-$y$ coordinate and movement distances are calculated using the Pythagorean identity, it is easy to predict the performance attainable with alternate key layouts. For example, it is often claimed that the Dvorak layout is preferable to the Qwerty layout because key positions were assigned along principles of time-motion studies and scientific measurements of efficiency for two-handed touch typing (Potosnak 1988). These benefits may not necessarily transfer to typing with a stylus on a soft keyboard, however. We tested this by reassigning the $x$-$y$ coordinate for each key to match the Dvorak layout. Using our preferred bandwidth figure of 4.9 bps, we
predict a lower limit of 8.6 wpm and an upper limit of 27.1 wpm, slightly below the predictions for the Qwerty layout.

Since our model is for a soft keyboard, we can go beyond simply rearranging the key positions, however. We can also adjust key sizes, with the most probable symbols assigned to “big keys” and the least probable assigned to “small keys”. Such a keyboard would have a large space bar in the middle, “big” keys around the space bar for the most probable letters (e.g., e, a, t, o), and “small” keys around the outside for the least probable letters (e.g., q, x, z).

Despite a lot of interesting research with alternate key layouts, or even chord keyboards, the momentum of the Qwerty layout is substantial. The promise of payoffs in terms of expert performance levels has had little effect in convincing users to shun the venerable Qwerty arrangement. Further discussion on alternative key layouts is presented in the following chapter.

2.4.4 Stylus tapping and soft keyboards

Stylus tapping on a soft keyboard is an entry method well-suited to pen-based computers. However, it is often ignored in the popular press which focuses on software that accepts the sloppy scrawl of the ambivalent user. The performance of recognition software is quite another issue. The gap between manufacturer’s claims and the goods delivered was expounded by Kurtenbach et al. (1994), for example.

From a qualitative viewpoint, one drawback of stylus tapping is the lack of kinaesthetic feedback: Users must maintain eye-fixation on the screen. This is also true of hand printing if an input grid is presented. Cursive handwriting recognition promises to ease this constraint by allowing input anywhere on the input device. The input is recognized, converted to ASCII, and delivered to the application’s insertion point. Another drawback of soft keyboards is that they consume screen real estate. This is particularly troublesome with the small form-factor of personal
digital assistants. As with current implementations on graphical user interfaces, the soft keyboard is usually displayed only while in use. There is also the “feel” of the LCD surface. Users often note that the touch is too slippery – unlike the texture of pen-on-paper. Parallax (the visual gap between the stylus’ contact point and the visual feedback when the display is viewed at an angle) is also a problem, as noted by Goldberg and Goodisman (1991).

In our view, pen-based computers are unsuited to applications that require extensive text input. Veniola (1994) also raises this point. Indeed, the promise of pen-based computers lies in vertical markets, wherein the application is so highly understood and constrained that input is primarily through on-screen buttons, menus, etc. For input-intensive tasks, the venerable Qwerty keyboard – like the one used to write this thesis – remains unchallenged. With input rates well beyond 40 wpm for expert users, the alternate forms of input cited herein are reduced to niche applications, such as information kiosks or banking machines, or to occasional input on pen-based computers.

2.5 Conclusions

We have presented a model that gives upper-bound and lower-bound predictions for entry rates using a pen or stylus to tap on a soft Qwerty keyboard. The model is based on a characterization of the text-entry task based on linguistic tables of frequency probabilities of digrams (including spaces) in common English. Visual scan time is predicted by the Hick-Hyman law for choice reaction time, and movement time is predicted by Fitts’ law for rapid aimed movements.

With predicted rates ranging from about 9 wpm for novices to 30 wpm for experts, stylus-tapping is a viable input method. Considering the accuracy problems that plague systems supporting hand writing recognition, stylus tapping on a soft
keyboard represents a fast and easy alternative for text entry on pen-based computers.

However, further work is needed to refine this modelling technique, for example, in determining the correct coefficient in the Fitts’ law model and in exploring and refining other aspects of the model, such as treatment of the space and punctuation characters, or sensitivity to changes in the language model.

The stylus model as presented in this chapter is theoretically sound. However, the quality of the model, particularly for novices, has yet to be determined. In the next chapter an evaluation of the novice stylus typing model is presented, weaknesses are discovered, and new avenues for improvement are identified.
Chapter 3
Comparing the Soft Keyboard Model with Empirical Data\textsuperscript{12}

The stylus model presented in the previous chapter predicts expert and novice typing rates for given keyboard arrangements. In this chapter, the novice model is evaluated. Text entry rates are explored for several variations of soft keyboards\textsuperscript{13}.

In a quick, novice, test, subjects achieved rates of 20.2 words per minute (Qwerty), 10.7 wpm (ABC – alphabetic), 8.5 wpm (Dvorak), 8.0 wpm (Fitaly), 7.0 wpm (JustType), and 8.0 wpm (telephone). At 8 to 10 wpm, the novice predictions were found to be low for all layouts because the dominant factor is the visual scan time, rather than the movement time. The Qwerty rate of 20.2 wpm is consistent with observations in other studies. The relatively high rate for Qwerty suggests that there is skill transfer from users’ familiarity with desktop computers to the stylus tapping task.

3.1 Introduction

The previous chapter developed a model of stylus typing in the context of the Qwerty layout. Since then, we have incorporated some minor improvements and refinements, for example, in accommodating the space bar\textsuperscript{14}. Our predictions for the Qwerty layout stand as follows:


\textsuperscript{13} We present an empirical test with 24 subjects. Six keyboards were examined: the Qwerty, ABC (alphabetic), Dvorak, Fitaly, JustType, and telephone.

\textsuperscript{14} The enhancements are discussed in section 4.1.1 of Chapter 4.
• Novice: 8.9 wpm
• Expert: 30.0 wpm

The novice prediction is difficult to validate empirically for two reasons. First, it is difficult to find subjects who have never used or seen a Qwerty keyboard and who, therefore, would exhibit visual scanning as predicted for novices. Second, in an empirical test the novice classification fades quickly since subjects gain familiarity with the key layout within a few taps. In fact, a real test of a novice prediction would require the keys to be randomly reassigned after each tap. So, we view the novice prediction as a lower threshold at which subjects begin to enter text, but from which they would rise quickly.

The expert prediction is also difficult to test for at least two reasons. First, proposing a single measure of expert performance is simplistic because users can always attain small improvements in performance, consistent with the power law of practice (DeJong 1957). Second, measuring expert performance requires a study conducted over many sessions, and this is very labour intensive. To our knowledge, no such study exists for text entry on a soft keyboard using a stylus.

There are a few studies of text entry with soft Qwerty keyboards, although the task was usually administered only for one or two hours. MacKenzie et al. (1994a) reported rates of 22.9 wpm for stylus tapping on a soft Qwerty keyboard. We expect that with practice entry rates would increase, levelling off near 30 wpm. Related studies used a soft keyboard on a touch screen with entry via the fingers rather than a stylus. Entry rates ranged from about 12 wpm (Gould et al. 1990) to 25 wpm (Sears 1991). In another study (Sears et al. 1993), subjects were allowed to use both hands on a touchscreen. Text entry rates as high a 32 wpm were recorded; however, the comparison with stylus tapping is weak.

15 Such an experiment is described by Zhang (1998), and MacKenzie and Zhang (2001b). The experiment was a simple text entry task with the letter-to-key assignment randomised after each tap. The average entry rate for 12 subjects was 5.52 wpm.
Finally, it is important to note the limitations in the model. These are elaborated later in the context of an empirical test with several alternate layouts for soft keyboards.

3.2 Soft keyboard layouts

From the work described above, we proceeded to test our model on other soft keyboard layouts. Although there are numerous layouts to test, we limit our discussion to several interesting possibilities.

3.2.1 The Dvorak keyboard

Since the Dvorak layout is well known as a Qwerty alternative, it was a logical starting point. In its physical form, the Dvorak keyboard (Figure 22) is similar to a Qwerty keyboard (Figure 4b). By simply renaming the keys, a Qwerty layout can be transformed into a Dvorak layout. The Dvorak keyboard was designed to optimise two-handed touch typing. The idea is that higher entry rates can be obtained if common digrams are entered by fingers on opposing hands instead of on the same hand (Potosnak 1988). As well, the most common letters (e.g., E, T, A, H) are positioned along the home (viz. middle) row.

After entering the x-y coordinates of the Dvorak keyboard into our spreadsheet, the novice and expert predictions were immediately available to us. Our predictions are as follows:

- Novice: 8.7 wpm
- Expert: 27.2 wpm
At 27.2 wpm, the expert prediction is below our 30.0 wpm prediction for a Qwerty layout. This illustrates the distinct difference between optimising for two-handed touch typing versus optimising for one-handed touch tapping with a stylus.

The novice rate of 8.7 wpm is slightly lower than our Qwerty prediction of 8.9 wpm. In general, the novice prediction is dominated by the visual scan time, so any layout permutation that minimizes movement has only a minor impact on the novice entry rates. This point is emphasized in the following example: If we consider, in the extreme, that the novice visual scan time of 951 ms is the only component of the task, then this alone represents an entry rate of \((1 / 0.951) \times (60 / 5) = 12.6 \text{ wpm}\). Therefore, novice predictions will always be lower than 12.6 wpm by an amount determined by the movement component of the task.

3.2.2 The ABC (alphabetic) layout

In an effort to minimize screen real estate, soft keyboards can be streamlined, for example, by using short and wide or tall and narrow layouts. We investigated a variety of such possibilities. Figure 23 illustrates an example which we call the ABC layout.

There are certain advantages to grouping the keys as shown in Figure 23, (or in other alphabetic arrangements, see Figure 27) and having a space bar that spans the full keyboard. The alphabetic ordering gives novices a good indication of the
location of each key, and this should reduce the visual scan time. Since space is the most prevalent character in text-entry tasks, the size and proximity of the space bar to the other keys should reduce overall movement time.

After entering the key coordinates in our spreadsheet, the following predictions emerged:

- Novice: > 9.6 wpm
- Expert: 28.8 wpm

We indicate “greater than” 9.6 wpm for the novice prediction for the simple reason that the alphabetic ordering of keys precludes anyone from being a novice, provided they know the alphabet (a reasonable assumption).

Figure 23 - An alphabetic soft keyboard
The expert prediction is slightly lower than our 30.0 wpm prediction for the Qwerty layout. The close proximity of all keys to the space bar is a definite advantage for the ABC layout; however, this appears to be offset by placing letters in two columns. Words like “satisfaction” require substantial up-down-up pen travel, and this tends to push the prediction down. A three-column version would alleviate this, but there is a cost, since shifting some keys to the third column increases their distance to the space bar. A three-column arrangement was not tested.

3.2.3 The *Fitaly* keyboard

The *Fitaly* soft keyboard is a commercial product (Textware Solutions, Burlington, MA; US Patent Number 5,487,616) designed to optimise text entry with a stylus (see Figure 24). The most striking feature of the layout is the presence of two space bars. The proximity of the most common letters in English (e.g., E, T, A, H) to the space bars is also immediately apparent. The keyboard’s name is taken from the letter sequence along the second row of keys.

<table>
<thead>
<tr>
<th>z</th>
<th>v</th>
<th>c</th>
<th>h</th>
<th>w</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>i</td>
<td>t</td>
<td>a</td>
<td>l</td>
<td>y</td>
</tr>
<tr>
<td>space</td>
<td>n</td>
<td>e</td>
<td>space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>d</td>
<td>o</td>
<td>r</td>
<td>s</td>
<td>b</td>
</tr>
<tr>
<td>q</td>
<td>j</td>
<td>u</td>
<td>m</td>
<td>p</td>
<td>x</td>
</tr>
</tbody>
</table>

Figure 24 - Textware Solutions *Fitaly* keyboard

The *Fitaly* keyboard was designed to minimize the travel from one letter to the next. According to the developer, the average travel is 1.8 keys compared to 3.2 keys for a Qwerty layout. These figures were obtained using a corpus of digram probabilities
similar to that described in the previous chapter, and by Soukoreff and MacKenzie (1995).

After entering the $x$-$y$ coordinates of the *Fitaly* keys into our spreadsheet (and introducing a few modifications to accommodate the two space bars), we obtained the following predictions:

- Novice: 9.7 wpm
- Expert: 42.0 wpm

The novice prediction at 9.7 wpm is close to the predictions given earlier. The expert prediction of 42.0 wpm is impressive. This is a full 40% higher than the 30.0 wpm expert prediction for the Qwerty layout. To our knowledge, no empirical evaluation of the *Fitaly* soft keyboard exists.

### 3.2.4 The telephone keyboard

Although extensive text entry via a telephone keyboard is arguably impractical, some text entry occasionally occurs using this familiar product (see Figure 5).

Examples of text entry via a telephone include automated directory assistance, automated facsimile request (also known as “fax back”), or programming names and numbers into home and cellular telephones. Since the telephone keyboard is small (12 keys), it is an appealing choice for other mobile products that require text entry. So, the idea of modelling and predicting the text entry task for the telephone keyboard is perhaps worthwhile. We have chosen two techniques to explore here.

#### 3.2.4.1 With disambiguation

Since there are three or four letters assigned to each key, the telephone keyboard poses a special problem: disambiguating the input. One technique uses a dictionary
and a built-in disambiguating algorithm to determine each word. This will work especially well if the dictionary is small, as, for example, in finding a phone extension in a small company by entering an employee’s last name. Consider the following sequence:

5 6 6 3 7

Although this sequence has $3 \times 3 \times 3 \times 3 \times 4 = 324$ permutations of the letters on the keys of the telephone keypad, most are nonsense. If the dictionary includes only employees’ last names, the most likely name is “Jones”, illustrated as follows and easily verifiable by examining Figure 5:

J O N E S
5 6 6 3 7

Our novice and expert predictions for the telephone keyboard with built-in disambiguation are as follows:

- Novice: > 9.1 wpm
- Expert: 43.5 wpm

As with the ABC layout, we indicate “greater than” 9.1 wpm for the novice prediction since the letters are sequenced alphabetically. Although the expert figure is comparable to Qwerty, bear in mind that for unconstrained text entry, the method’s utility hinges on the disambiguating algorithm’s ability to provide reasonably accurate results. Implementation details are also extremely important, such as providing a mechanism to select from alternate plausible words (e.g., 228 = “bat” or “cat”), or to explicitly enter a word when none is found in the dictionary.

3.2.4.2 With explicit entry

Other text entry techniques for telephone keyboards adopt some mechanism to explicitly identify each letter. Two schemes previously discussed (in Chapter 1) are
(a) multi-tap (press each key one, two, three, or four times to select the 1\textsuperscript{st}, 2\textsuperscript{nd}, 3\textsuperscript{rd}, or 4\textsuperscript{th} letter on the key), or (b) the two-key method (press a key followed by the number 1, 2, 3, or 4 to select the 1\textsuperscript{st}, 2\textsuperscript{nd}, 3\textsuperscript{rd}, or 4\textsuperscript{th} letter on the key). We choose the latter technique to predict with our model. So, for example, the name “Jones” would be entered as the following sequence of numbers:

\[
\begin{array}{c}
J \\
5
\end{array}
\begin{array}{c}
O \\
1
\end{array}
\begin{array}{c}
N \\
6
\end{array}
\begin{array}{c}
E \\
3
\end{array}
\begin{array}{c}
S \\
6
\end{array}
\begin{array}{c}
2 \\
2
\end{array}
\begin{array}{c}
7 \\
3
\end{array}
\]

Our predictions are as follows:

- Novice: > 7.4 wpm
- Expert: 22.6 wpm

The comparatively low predictions are, of course, due to the need to tap two keys for each letter entered.

3.2.5 The \textit{JustType} keyboard

As with the telephone keyboard, the \textit{JustType} keyboard (Aiki Limited, Seattle, WA) places more than one letter on each key. The 26 letters of the alphabet are encoded on nine keys, with eight keys encoding three letters each, and one key encoding two letters (see Figure 25).

The \textit{JustType} keyboard works with a large dictionary and a disambiguation algorithm to determine the user’s intended words. Unlike the telephone keyboard, the letter groupings are distinctly non-alphabetic. The groupings in Figure 25 were chosen to optimise the performance of the disambiguating algorithm.\footnote{The layout in Figure 25 is a sample \textit{JustType} keyboard, as provided by Aiki Limited. Although the letter groupings are fixed, other key layouts are possible.} After inputting the key coordinates in our spreadsheet model, we arrived at the following predictions:
• Novice: 9.8 wpm
• Expert: 42.4 wpm

The expert entry rate is 41.3% faster than for the Qwerty keyboard.

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>P</th>
<th>Q</th>
<th>A</th>
<th>D</th>
<th>F</th>
<th>N</th>
<th>B</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>O</td>
<td>L</td>
<td>X</td>
<td>E</td>
<td>W</td>
<td>V</td>
<td>I</td>
<td>M</td>
<td>G</td>
</tr>
<tr>
<td>Space</td>
<td>C</td>
<td>Y</td>
<td>K</td>
<td>T</td>
<td>H</td>
<td>J</td>
<td>S</td>
<td>U</td>
<td></td>
</tr>
</tbody>
</table>

Figure 25 - Aiki Limited *JustType* keypad

### 3.3 Evaluating novice behaviour

In most evaluations with input devices or interaction techniques, learning effects are considered a confounding factor. There are two common ways to deal with this. The first is to use a very primitive interaction task – one that quickly yields a high level of proficiency (e.g., MacKenzie et al. 1991). The second is to sufficiently practice subjects until a criterion level of performance is attained (e.g., Card et al. 1978). Both methods are pragmatic. Indeed, expert behaviour is not a “state” that one captures, because human performance improves indefinitely with practice in both complex and simple tasks (DeJong 1957).¹⁷

Measuring novice behaviour, however, presents a different problem. The goal is to capture a snapshot of human performance at the onset of learning. We call this the *novice experience*. This is particularly worthwhile for user interfaces targeted at consumers, since immediate usability is important. New methodologies must be

---

¹⁷ If both human performance and practice are plotted in logarithmic scales, a continuous linear improvement in performance appears. That there is a diminishing return in the performance improvement versus practice is inherently accommodated in the logarithmic transform of the measures.
devised, because substantial learning may occur within a few minutes of exposure, and this effectively contaminates the novice status of subjects.

In a study, MacKenzie and Zhang (1997) captured the novice experience with the Graffiti alphabet of unistroke symbols for handwriting recognition. They measured subjects’ performance following one-minute and then five minutes of controlled exposure to the technology. In the following section, we describe a simple methodology, seeking to similarly capture the novice experience with various soft keyboards. We refer to our method as a “quick test” rather than a “novice test”, because of the simple technique employed, and because of the special problem presented by layouts such as Qwerty where prior exposure is inevitable. Since fewer measures are gathered per subject, we used more subjects than are commonly employed in evaluations with input devices. (In the study by Card and colleagues, 1978, for example, only five subjects were used.)

3.4 Method

3.4.1 Subjects

We recruited twenty-four volunteer subjects (18 male, 6 female), all of whom were regular users of desktop computers with Qwerty keyboards. None had previously used pen-based computers on a regular basis.

3.4.2 Apparatus

We used a paper facsimile for each of following keyboards:

- Qwerty
- Dvorak
- ABC
- Fitaly
- Telephone (with built-in disambiguation)
Subjects sat at a desk with the keyboard layout in front of them on the surface of the desk. Text was entered by tapping on the paper image of the keyboard using a stylus. The stylus was borrowed from a Wacom graphics tablet. Prior to beginning, subjects were briefly shown the keyboard layout. The operation of each keyboard was explained to each subject. This was particularly important for the JustType and Telephone keyboards, since these have multiple characters per key.

3.4.3 Procedure

The following 45-character phrase of text was used:

\[ \text{the quick brown fox jumped over the lazy dogs} \]

Each subject entered the phrase on each of the six keyboard layouts. The order of keyboards was counterbalanced using a $6 \times 6$ Latin square, with four subjects receiving each ordering. Since we sought to capture the novice experience, no practice trials were given and the text phrase was entered once only. We felt that this procedure, combined with data from 24 subjects, would provide stable measurements of the novice experience with each soft keyboard. The most serious flaw in this reasoning follows from the subjects’ prior experience with Qwerty keyboards. More will be said about this later.

The total entry time was measured with a stop watch. Entry time (measured in seconds) was converted to entry speed (words per minute) as follows:

\[ \text{Entry Speed} = \left( \frac{\text{Entry Time}}{44} \right)^{-1} \times \left( \frac{60}{5} \right) \]

The text phrase had 45 characters; however, since entry was timed from the first tap, there was no movement time to the first character. Therefore, entry time was
divided by 44, rather than 45. Taking the reciprocal transforms the measure into “characters per second”. Multiplying by 60 and dividing by 5 transforms the measure to “words per minute”.

Subjects were instructed to tap the phrase as quickly as possible while trying to avoid making mistakes. They were also reminded to tap spaces between words. As the experiment was a simulation of real soft keyboards and ran without data capture software, error rates were not recorded.

3.5 Results and discussion

The mean entry speed across all subjects and keyboards was 10.5 wpm. There was a significant effect of keyboard on entry time ($F_{5,23} = 184.3$, $p < .0001$). The Qwerty layout was the fastest (20.2 wpm), while the JustType layout was the slowest (7.3 wpm). The results are summarized in Table 5. The relatively low standard deviations suggests that the mean scores for each layout were consistent across subjects.

That our test was a quick test rather than a novice test is evident in Table 5. Since all subjects were experienced desktop computer users, they were by no means “novices” with the Qwerty layout. A similar, although less emphatic statement, can be made for the ABC and Telephone keyboards, since the sequential ordering of letters gives subjects a reasonable clue to each letter’s location. Furthermore, over the duration of each test, the novice status fades as subjects become familiar with the arrangement of keys.
Table 5 - Entry speed versus keyboard layout

<table>
<thead>
<tr>
<th>Keyboard Layout</th>
<th>Entry Speed (wpm) (\dagger)</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qwerty</td>
<td>20.2</td>
<td>4.9</td>
<td></td>
</tr>
<tr>
<td>Dvorak</td>
<td>8.5</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>ABC</td>
<td>10.6</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>Fitaly</td>
<td>8.2</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>Telephone</td>
<td>8.1</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>JustType</td>
<td>7.3</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>10.5</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

\(\dagger\) n = 24 subjects

It is instructive to compare the observations from our quick test with the novice and expert predictions presented earlier. Since the observed rates should fall between the novice and expert predicted rates, they are presented in Figure 26 as the middle of three bars for each keyboard.

Figure 26 - Comparison of novice, quick test and expert predictions for six soft keyboard layouts
At 20.2 wpm, our quick test measurement for the Qwerty layout illustrates subjects’ familiarity with this keyboard. This figure is only slightly below the 22.9 wpm figure measured in a test with a real Qwerty soft keyboard (MacKenzie et al. 1994a), and is 67.3% of the predicted expert rate of 30.0 wpm.

The ABC layout, with its familiar arrangement of keys and with the large space bar, did not score as high as expected. The average entry speed of 10.6 wpm is about half that of the Qwerty layout, and is only 17.8% higher than the novice prediction of 9.0 wpm. This implies that subjects’ daily experience with a Qwerty keyboard is an important advantage for the stylus tapping task.

We were surprised at the scores for the other four keyboard layouts. Ranging from 7.3 wpm (JustType) to 8.5 wpm (Dvorak), all rates were below those predicted for novices. This is problematic since, as noted earlier, the novice predictions should be a lower threshold from which subjects rise quickly when working with a new keyboard layout. We have considered several reasons for the low entry rates. First, our text phrase included every letter of the alphabet. On the one hand, this is good because it ensures subjects visit every key during the task. On the other hand, the appearance of all 26 letters of the alphabet in a 45-character phrase means the phrase is not typical of common English. Thinking this might push the observed speeds down, we generated for each keyboard a novice prediction for the specific phrase, “the quick brown fox ...”. These predictions differed very little from the novice predictions in Figure 26 (typically 0.2 wpm), so we ruled out this explanation. Other explanations are explored in the next section.

3.5.1 The novice experience

Although the predictions are less striking for novices than experts, it is the novice experience that often determines the overall acceptance of new technology. Hence, it is important to capture aspects of the task that affect or determine novice performance.
3.5.1.1 Visual scan time

For a soft keyboard with an unfamiliar layout, the visual scan time is one aspect of the novice experience that must be examined. Our figure of 951 ms is the visual scan time for 27 choices, predicted using Equation 10. This estimate is sensitive to the slope coefficient in Equation 10, which we set to 200 ms/bit (Hick 1952). However, in Welford's extensive review of choice reaction time studies (1968, pp. 60-104), slopes vary from about 160 ms/bit to about 320 ms/bit. Using Equation 10, this implies that the visual scan time could range from about 760 ms to about 1.52 s. Therefore, one explanation for the low observed entry rates for the four “novice” keyboards (Figure 26) is that our 951 ms estimate of visual scan time is too low. Since the predicted vs. observed discrepancies range from 2.3% (Dvorak) to 28.6% (JustType), it is not clear how much of an adjustment may be warranted. It is our feeling, however, that other, more significant factors are at play, as discussed in the next section.

3.5.1.2 Movement time

Other possibilities for the discrepancies between our observations and the novice predictions arise from a re-examination of the movement component of the task. Our model assumes that text entry consists of a visual scan time – set to zero for experts – followed by a movement, with each movement beginning where the previous movement ended. This may be too simplistic. Since soft keyboards lack kinaesthetic and tactile feedback, on-going visual feedback is required, even for experts. Consider, as an example, a right-handed expert subject. Following a tap on the left side of the keyboard, much of the keyboard is obscured by the hand. If the following key is on the right, movement will proceed in that direction and the spatial arrangement of keys will be revealed as the hand moves toward the key. There may be a slight degradation in performance – one that is not accounted for in our model.
For novice subjects, the situation is quite different. We observed a definite pattern of lifting the hand above the keyboard after each tap. This behaviour was not limited to situations where the previous tap obscured the keyboard; it was a pattern that occurred consistently following each tap. Since novice subjects do not know the location of the next key, lifting the hand to a neutral, revealing, position above the keyboard is a natural strategy. This is likely a parallel action, overlapping the visual search. Clearly, the motor component of the prediction is affected and this weakens our model’s predictive power for novices.

Another limitation pertains to the size of the keyboard. According to our model a keyboard can be scaled up or down without affecting the movement time prediction. This is due to a simple characteristic in Fitts’ law: if the movement distance is doubled (or halved) but target width is also doubled (or halved), the predicted movement time remains the same (see Equations 2 and 3). In other words, a keyboard may be scaled up or down without affecting the movement component of the model. Although this may be true within limits, it is probably not true in the extreme, or when other identifiable factors enter into the task. There are a variety of such factors. For very small keys, the contact area of the stylus tip may be an appreciable percentage of the size of the keys. This would tend to reduce the effective area of the key. Drury and Hoffmann (1992) showed that an optimal data-entry keyboard will have an inter-key gap equal to the width of the data entry probe (viz., stylus tip). For very small soft keyboards, this effect may be important. This is likely confounded with parallax – the visual gap between the stylus’ contact point and the visual feedback when the display is viewed at an angle. However, further study is needed before the effect can be quantified.

Another factor is the limbs and muscles groups required for the movement. For a small keyboard, the majority of the moves occur with the wrist anchored; whereas for larger keyboards the wrist is airborne as movement engages the forearm as well as the wrist. The highly learned task of handwriting is performed primarily with
the wrist resting. The affect of different muscle groups on performance has been well studied (Balakrishnan and MacKenzie 1997; Gibbs 1962; Langolf, Chaffin and Foulke 1976).

3.6 Conclusions

Text entry on small mobile systems remains a challenge for computing systems of the future. Stylus tapping on a soft keyboard offers easy entry; however, rates are moderate at best and a keyboard must be presented on the system’s display, thus occupying valuable screen real estate. Expert entry rates may reach 30 wpm for the Qwerty layout, or in excess of 42 wpm for optimised layouts. Because eye fixation is a requirement of interaction with soft keyboards, fatigue may prove a factor with prolonged use.

Novice entry rates are in the 7 to 10 word per minute rate for most layout permutations. However, experienced users of desktop computers may enter text with an immediate rate of about 21 words per minute on a soft keyboard with a Qwerty layout. This suggests that the venerable Qwerty layout is here to stay, both for physical keyboards on desktop computers and for soft keyboards that support stylus tapping.
In Chapter 2, a mathematical model was presented that predicts the expert and novice typing rates for a given arrangement of keys on a soft keyboard. The purpose of the model was to evaluate soft keyboards, so it seems only natural that researchers would try to find an optimal soft keyboard using the model. This chapter presents a review of the considerable work done in this area.

4.1 Introduction

This chapter presents some variations of soft keyboards developed in industry and in research labs. There have been many soft keyboards proposed, and researchers have demonstrated great imagination and ingenuity in their designs. For example, soft keyboards have been proposed that have more than one space key (reflecting the high frequency of spaces), that have fewer keys than the 26 alphabetic letters (to conserve space), and that are alphabetic (and hence easy for beginners to become familiar with).

Although most of the keyboards covered in this chapter were created after, and in response to, the publication of the predictive model of Chapter 2, two keyboards designed for rehabilitation situations (Getschow and Cubon) predate our model.
4.1.1 Predicted text entry rates for several soft keyboards

Each of the keyboards discussed in this chapter will be described in its own section of this chapter, but we begin by giving the predicted expert entry rates according to the model of Soukoreff and MacKenzie (1995, and Chapter 2) for each of the keyboards we shall discuss. Our predications are given in Table 6 sorted by predicted entry rate from highest to lowest.

Several enhancements have been introduced in the model since it was introduced in 1995. (a) Multiple space keys are accommodated using trigrams and assuming that expert typists always use the most efficient (not always the nearest) space key. (b) Non-rectangular shaped keys are accommodated by choosing inter-key distances and key sizes that make sense in the context of Fitts’ law. (c) Keyboards with mode shifts can be accommodated (the DotNote keyboard described below consists of two small keyboards each containing half of the alphabet, with a mode key depressed to toggle between them). Where necessary, digrams are modelled as the Fitts’ law prediction for the motion from the first key to the mode shift key, followed by the prediction to move from the mode shift key to the second character.

The entries in Table 6 are updated from earlier publications to reflect these changes (and corrections have been applied to erroneous predictions published for the OPTI I and OPTI II keyboards, see Footnote 19). Also, see Zhai (2000) for a discussion of the model’s sensitivity to factors such as the coefficients in the Fitts’ law model and the corpus used in building the language model.

Table 6 also gives the improvement of each soft keyboard relative to Qwerty. At the top of the list is the Metropolis II keyboard, with a predicted text entry rate 42.9% higher than Qwerty. We will visit this shortly.
<table>
<thead>
<tr>
<th>Keyboard Layout</th>
<th>Expert Prediction (wpm)</th>
<th>Improvement Over Qwerty (%)</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metropolis II</td>
<td>42.94</td>
<td>42.9</td>
<td>Figure 30c</td>
</tr>
<tr>
<td>OPTI II</td>
<td>42.37</td>
<td>41.0</td>
<td>Figure 29b</td>
</tr>
<tr>
<td>OPTI I</td>
<td>42.16</td>
<td>40.3</td>
<td>Figure 29a</td>
</tr>
<tr>
<td>Metropolis I</td>
<td>42.15</td>
<td>40.3</td>
<td>Figure 30b</td>
</tr>
<tr>
<td>Fitaly</td>
<td>41.96</td>
<td>39.7</td>
<td>Figure 24</td>
</tr>
<tr>
<td>Hook's</td>
<td>41.15</td>
<td>37.0</td>
<td>Figure 30a</td>
</tr>
<tr>
<td>Getschow</td>
<td>40.89</td>
<td>36.1</td>
<td>Figure 31</td>
</tr>
<tr>
<td>Cubon</td>
<td>37.02</td>
<td>23.2</td>
<td>Figure 28</td>
</tr>
<tr>
<td>Lewis</td>
<td>34.65</td>
<td>15.3</td>
<td>Figure 32</td>
</tr>
<tr>
<td>ABC III</td>
<td>32.50</td>
<td>8.2</td>
<td>Figure 27c</td>
</tr>
<tr>
<td>ABC IV</td>
<td>30.18</td>
<td>0.5</td>
<td>Figure 27d</td>
</tr>
<tr>
<td>ABC II</td>
<td>30.13</td>
<td>0.3</td>
<td>Figure 27b</td>
</tr>
<tr>
<td>Qwerty</td>
<td>30.04</td>
<td>-</td>
<td>Figure 4b</td>
</tr>
<tr>
<td>DotNote</td>
<td>29.46</td>
<td>-1.9</td>
<td>Figure 33</td>
</tr>
<tr>
<td>ABC I</td>
<td>28.79</td>
<td>-4.2</td>
<td>Figure 27a</td>
</tr>
</tbody>
</table>

### 4.2 Fifteen soft keyboards

Each of the soft keyboards appearing in Table 6 is discussed in its own section below.

#### 4.2.1 Alphabetic keyboards

There are two keyboard arrangements which most users are generally familiar with, Qwerty and alphabetic. The Qwerty keyboard was shown in a previous chapter (see Figure 4b). A few alphabetic arrangements appear in Figure 27. An experiment reported by MacKenzie et al. (1999c) found that subjects achieved rates of 20 wpm
Figure 27 - Some alphabetic keyboard arrangements

(a) ABC I, (b) ABC II, (c) ABC III, (d) ABC IV
on a Qwerty soft keyboard and 11 wpm using an ABC layout (ABC I in Figure 27a). The predicted expert entry rates are 30.04 wpm for a Qwerty soft keyboard and 28.79 wpm for the ABC I arrangement. Predictions for the ABC II, ABC III, and ABC IV arrangements are 30.13 wpm, 32.50 wpm, and 30.18 wpm, respectively (see Table 6).

4.2.2 The Cubon keyboard

Little information is available on the Cubon keyboard, except that it seems to have been proposed by R. A. Cubon and is used in rehabilitation situations for persons with the use of only one finger, or with a head-mounted pointing device. (Although not originally proposed as a soft keyboard, it is included here because it was optimised for text entry using a single pointing device.) We know of no published user studies. The Cubon keyboard arrangement that appears in Figure 28 is taken from Zhai (2000). The expert prediction for Cubon is 37.02 wpm.

![Cubon keyboard](from Zhai et al. 2000)

4.2.3 The Fitaly keyboard

Textware Solutions Incorporated, the inventors of the Fitaly keyboard (Burlington, MA; http://www.textwaresolutions.com/), used an ad hoc optimisation approach to minimise the distance between common character pairs. The resulting
keyboard (see Figure 24, on page 82) contains two space bars and the letters are arranged so that common pairs of letters are often on neighbouring keys. MacKenzie et al. (1999c) reported a walk-up (i.e., subjects did not have previous experience, and did not get much practice) typing rate for the Fitaly keyboard of 8 words per minute. The expert prediction for the Fitaly layout is 41.96 wpm (see Table 6).

4.2.4 The OPTI keyboards

MacKenzie and Zhang (1999b) used Soukoreff and MacKenzie’s model to produce an optimised keyboard arrangement. The OPTI I keyboard appears in Figure 29a. They reported a predicted expert typing rate of 58 wpm, however this prediction includes an error pointed out by Zhai and colleagues19. Our prediction for the OPTI I layout stands at 42.16 wpm.

MacKenzie and Zhang (1999b) performed a longitudinal study over 20 sessions comparing the Qwerty and OPTI I arrangements and found that the average typing rate for OPTI I increased from 17 wpm initially to 44 wpm after eight hours of practice (see Figure 1). For the Qwerty layout, rates increased from 28 wpm to 40 wpm, over the same interval. The average rates for OPTI I exceeded those for the Qwerty layout after about four hours of practice.

19 The model works by using digrams to model the users’ transitions from key to key as they enter text. However, a long space key (such as in the Qwerty keyboard) or multiple space keys are best modelled with trigrams. The error made by MacKenzie and Zhang (MacKenzie et al. 1999b; Zhang 1998) was a miscalculation involving the relative probabilities of trigrams containing a space character. Typically trigram frequencies are not explicitly represented, but, rather, are derived from digram frequencies. The probability of a trigram (i.e., the probability of the character sequence i-j-k) is found with the expression:

\[ P(i,j,k) = P(i,j) \sum \frac{P(j,k)}{P(j,s)} \]

where \( P(i,j) \) is the probability of digram i-j. MacKenzie and Zhang omitted the denominator from their calculations. This error was first reported by Zhai and Hunter (Hunter et al. 2000; Zhai et al. 2000).

100
The alert reader will notice that something is amiss; the observed rates actually exceeded the expert predictions! The most likely explanation is that the slope coefficient in the Fitts’ law prediction model is too conservative. The slope coefficient used in the predictions is 0.204 seconds per bit (in Equation 1 and 3, \( b = 1 / 4.9 = 0.204 \) seconds per bit, also see Section 2.2.4.2), a value obtained from a pointing device study using a stylus on a Wacom tablet in a serial tapping task (MacKenzie et al. 1991). The reciprocal of the slope coefficient is commonly known as the Fitts’ law bandwidth, and, in this case, is \( 1 / 0.204 = 4.9 \) bits per second. A discrepancy even of 1 bit/second is enough to raise the predicted rate above the observed rates.\(^{20}\) Although the model is clearly sensitive to the slope coefficient in Fitts’ law, adjustments do not change the rank order of the predictions in Table 6. The reader is directed to Zhai et al. for further discussion (2000).

---

\(^{20}\) The bandwidth coefficient in MacKenzie et al. (1991) was measured in an “indirect” task: subjects manipulated the stylus on a Wacom tablet while attending to the system’s display. Stylus tapping on a soft keyboard is a “direct” task: subjects manipulate the stylus on the soft keyboard while also visually attending to the soft keyboard. This, alone, is cause to suspect that the bandwidth coefficient used in Soukoreff and MacKenzie’s model is conservative. Although no experiment measuring the bandwidth coefficient for stylus tapping on a soft keyboard has been published, we suspect such an experiment would yield a higher bandwidth. The effect would be to increase all the predictions in Table 6. This would not change the rank order of the soft keyboards in Table 6.
In follow-up work, Zhang (1998) proposed a slight modification to OPTI I. The OPTI II appears in Figure 29b, and yields an expert prediction of 42.37 wpm (see Table 6).

4.2.5 The Hook’s and Metropolis keyboards

Zhai, Hunter and Smith (Hunter et al. 2000; Zhai et al. 2000) applied two physics inspired techniques to the model of Soukoreff and MacKenzie and generated optimal keyboards. They used a mechanical simulation of a mesh of springs, where the springs were stretched between the characters of the alphabet and tensioned proportionally to digram probabilities in English. The technique is an application of a greedy algorithm to reduce the physical distance between more likely character pairs.21 The result is a keyboard they call Hook’s keyboard after Hook’s law22 (see Figure 30a). It yields a predicted expert entry rate of 41.15 wpm.

A second approach they took was to apply the Metropolis algorithm, which is theoretically more sound because it uses a random-walk instead of a greedy algorithm.23 The Metropolis I and Metropolis II keyboards have higher predicted expert typing speeds, and appear in Figure 30b and Figure 30c. In their first

21 The term “greedy algorithm” refers to a class of algorithms for solving minimisation (or maximisation) problems. Greedy algorithms try to find the minimum solution of a problem by always moving in the direction of steepest descent. However, greedy algorithms can become trapped in “local minimums”. This is analogous to searching for the deepest point in a valley by always walking downhill – but becoming trapped in a hole in the side of the valley because a step upwards (to a slightly higher altitude) would be required to leave the hole.

22 Hook’s law states that the force due to tension in a spring is proportional to its extension, i.e. how much it has been stretched from its equilibrium length. Mathematically, Hook’s law states that: $F = -kx$, where $F$ is the force of the spring, $x$ is the distance that the spring has been stretched (or compressed), and $k$ is the spring constant, which depends upon the mechanical construction of the spring.

23 The Metropolis algorithm is a well-known approach to solving complex minimisation (or maximisation) problems, inspired by thermodynamics (Press, Teukolsky, Vetterling and Flannery 1992, page 444). When a liquid is slowly cooled until it becomes solid, the resulting crystal is very orderly and has almost the minimum energy possible. Metropolis takes a function representing the energy of a system and applies simulated annealing, solving the minimisation problem by modelling the effect slow cooling has on the energy of the system. While the Metropolis algorithm does not suffer from the local minimum problem, and it does find solutions with very low energy levels, it is still not guaranteed to find the solution with the absolute minimum energy level.
Figure 30 - Hook's and Metropolis I & II keyboards

(a) Hook's keyboard, (b) Metropolis I keyboard, and, (c) Metropolis II keyboard (from Hunter et al. 2000; Zhai et al. 2000)
publication reporting preliminary results (Hunter et al. 2000), they present a Metropolis keyboard, which we denote *Metropolis I*. In a later publication (Zhai et al. 2000) another Metropolis-derived keyboard is presented, which we call *Metropolis II*. The predicted expert entry rates for the *Metropolis I* and *Metropolis II* keyboards are 42.15 wpm and 42.94 wpm, respectively. *Metropolis II* has the distinction of yielding the fastest predictions of any soft keyboard tested (see Table 6). No user evaluation of the *Metropolis I* or *Metropolis II* keyboard has been published.

4.2.6 The Getschow keyboard

The Getschow keyboard (by Getschow, Rosen and Goodenough-Trepagnier 1986) is another keyboard designed for use in rehabilitation situations. Getschow et al. used a block greedy\textsuperscript{24} approach to exhaustively search for an optimal keyboard consisting of the 26 alphabetic letters, and the period and space characters. An important simplification made by Getschow et al. is the optimisation of the distance between frequent letters of the keyboard, instead of optimising the time to move between keys as predicted by Fitts’ law. However, because all of the keys in their design were identical in shape and size (see Figure 31), the effect is basically the same.

\textsuperscript{24} The term “Block Greedy” algorithm refers to an algorithm that breaks a minimisation (or maximisation) problem into sequential partitions. The partitions are chosen to be small enough that they can be solved by some means. The partitions are then solved one at a time, at each step assuming the results of the previous partitions are immutable.

To use the example of walking down-hill to find the deepest point in a valley, consider breaking the walk into 10 minute segments. Many people start walking down the hill. After 10 minutes, everyone compares their positions to find who got the furthest down the hill. All of the people then convene at that location, and start walking down hill from there for another 10 minutes. The process is repeated until a satisfactory minimum is found.

Although not guaranteed to find the universal minimum (or maximum), block greedy algorithms have a greater chance of avoiding local minimums, and they produce results faster, and can solve larger problems, than is possible with an exhaustive search.
First, a $3 \times 3$ keyboard containing the nine most frequent letters was optimised by exhaustive search (every possible arrangement of keys was generated and compared). Then, the nine keys were frozen in their positions, as the eight next most frequent letters were optimised around them. Once optimised, the seventeen ($9 + 8 = 17$) letters block-optimised so far were frozen in position, while the next eight letters were added and optimised. After one final round that added the remaining three keys, Getschow et al. had produced their keyboard.

Figure 31 - The Getschow keyboard (from Getschow et al. 1986)

No user studies exist of the Getschow keyboard.

4.2.7 The Lewis soft keyboard

Lewis et al. (Lewis et al. 1999b; Lewis et al. 1999c) also tried to optimise entry rates for a soft keyboard. They applied network analysis to character pair probabilities to determine the most strongly associated pairs. Then, using an *ad hoc* method to minimise distances for the strongly associated character pairs, they produced the keyboard arrangement in Figure 32. Lewis et al. performed a comparative user evaluation but they did not report their results; estimating from their published report (Lewis et al. 1999b, Figure 1) suggests that they measured typing speeds of 25 wpm for the Qwerty control condition, and 13 wpm for their keyboard design.
They also report that when asked, subjects indicated a preference for the Qwerty layout. Our expert prediction for the Lewis keyboard is 34.65 wpm (see Table 6).

<table>
<thead>
<tr>
<th>q</th>
<th>r</th>
<th>w</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>u</td>
<td>a</td>
<td>o</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>h</td>
<td>e</td>
<td>n</td>
<td>g</td>
</tr>
<tr>
<td>v</td>
<td>d</td>
<td>i</td>
<td>s</td>
<td>p</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>m</td>
<td>j</td>
<td>k</td>
</tr>
</tbody>
</table>

Figure 32 - Lewis keyboard (from Lewis et al. 1999b; Lewis et al. 1999c)

4.2.8 The DotNote keyboard

The DotNote keyboard by Útilware (http://www.utilware.com/) was designed to support single-handed text entry on the Palm as an alternative to the built-in Graffiti handwriting recognition which requires two hands (one to hold the device, the other to manipulate the stylus). In order to support finger or thumb typing, the DotNote keyboard fills most of the display with relatively large keys, however, this allows only half of the alphabet to appear at once. The most common letters appear on the default DotNote keyboard (Figure 33a), and a mode shift key switches to the second keyboard arrangement which contains the less common letters (Figure 33b). The arrangement of the keys in each soft keyboard is alphabetic.

Our expert prediction of 29.46 wpm for the DotNote keyboard is low compared to the other soft keyboards listed in Table 6, because of the need for the user to switch between the two keyboards as they enter text – generating more keystrokes than is required using other soft keyboards. However, this arrangement results in smaller keyboards with fewer buttons, which could be useful in some applications.
4.3 Conclusions

The model of text entry via tapping on a soft keyboard appearing in Chapter 2, and published as Soukoreff and MacKenzie (1995) has generated great interest in soft keyboard design and optimisation. Soft keyboard optimisation remains an active research area today, however, it remains doubtful whether even a greatly superior soft keyboard design could supplant the ubiquitous Qwerty keyboard.
Chapter 5
A Model of Two-Thumb Text Entry

Although text entry has been extensively studied for touch typing on standard keyboards, and single finger, thumb, or stylus input (for example, Chapters 2 through 4, and Silfverberg et al. 2000; Soukoreff et al. 1995), no such work exists for two-thumb text entry. In this chapter, we propose a model for this mode of text entry. The model is based solely on the linguistic and motor components of the two-thumb typing task; thus, it predicts the peak rate for expert text entry.

We present predictions based on two different Fitts’ law models. Employing a published Fitts’ law model of thumb motion on a cellular telephone key pad (Silfverberg et al. 2000), the prediction for peak expert text entry rate is 60.74 words per minute. Using our own Fitts’ law models of thumb motion on a miniature Qwerty keyboard, the prediction obtained is 58.06 words per minute. We also describe the study undertaken to generate Fitts’ law models of both thumbs on a miniature Qwerty keyboard.

A detailed sensitivity analysis is included to examine the effect of changing the model’s components and parameters over a broad range (±50% for the parameters). The model demonstrates reasonable stability – predictions remain within about 10% of predicted nominal values.

---

5.1 Introduction

Current research in text entry includes significant interest in the use of small physical keyboards. Some devices allow text entry with as few as five keys, such as the AccessLink II by Glenayre Electronics. Others sport a complete, but miniature, Qwerty keyboard, such as the Blackberry by Research In Motion. These are both examples of two-way pagers. As well, text entry using the mobile phone keypad has grabbed the attention of users and researchers. While most mobile phones support text entry via the conventional telephone keypad, Nokia has recently introduced the 5510, a mobile phone with a full Qwerty keyboard.

Much of the interest is spurred by the remarkable success of so-called SMS messaging on mobile phones. The ability to discretely, asynchronously, and at very low cost, send a message from one mobile device to another has proven hugely successful, particularly in Europe. The statistics are staggering; message volumes are now approaching one billion messages per day. (Various SMS statistics are available at http://gsmworld.com/technology/sms.html) Given the limited capability of the mobile phone keypad for text input, it is not surprising, therefore, that the current wave of mobile text entry research includes numerous researchers and companies working on new ideas to improve text entry techniques for mobile phones or other anticipated mobile products supporting similar services.

In this chapter, we propose a model of two thumb text entry on small physical keyboards. The model provides both a behavioural description of the interaction plus predicted peak text entry rates for expert users. In the following sections, the model is presented. We include a detailed analysis examining the model’s sensitivity to changes in the various components and parameters that affect the prediction. A study is discussed that was used to generate Fitts’ law models of two-thumb motion on a miniature Qwerty keyboard. Predictions using these models and a previously published model are presented.
Two-thumb text entry is depicted in Figure 34.

![Figure 34 - Two-thumb text entry](image)

The device shown is a Sharp *EL-6810* organizer (also shown in Figure 35a). Other devices for which a similar interaction style is expected include the Motorola *PageWriter 2000* two-way pager (Figure 35b), the Research In Motion *Blackberry* two-way pager (Figure 35c), and the Nokia *5510* mobile phone (Figure 35d). These are all small devices bearing a complete, but miniature, Qwerty keyboard.
5.2 Model overview

To model two-thumb text entry, the following steps are proposed:

1. Obtain a word-frequency list derived from a language corpus.
2. Digitise the miniature keyboard of interest.
3. Determine the assignment of the left and right thumbs to letters and keys.
4. Given the information in steps 1-3, compute the predicted entry time for each word in the corpus, including the time to enter a terminating space character after each word.
5. Multiply the predicted entry time for each word by the frequency of the word in the corpus, then sum the values. The result, $t_{\text{CORPUS}}$, is the time to reproduce the entire corpus.

6. Multiply the size of each word (including a terminating space character) by the frequency of the word in the corpus, then sum the values. The result, $n_{\text{CORPUS}}$, is the number of characters in the corpus.

7. Compute $t_{\text{CHAR}} = t_{\text{CORPUS}} / n_{\text{CORPUS}}$. The result, $t_{\text{CHAR}}$, is the mean time to enter each character in the corpus. The units are “seconds per character”.

8. Compute $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) \times (60 / 5)$. The result, $t_{\text{WPM}}$, is the text entry throughput in “words per minute”. The scaling factors are 60 seconds per minute, and 5 characters per word.

The steps above are similar to those in prior work on text entry on soft keyboards using a stylus (Chapter 2, MacKenzie et al. 1999b; MacKenzie et al. 1999c; Soukoreff et al. 1995; Zhai et al. 2000) and single-finger or thumb text entry on a mobile phone keypad (Silfverberg et al. 2000). There are, however, two significant departures. First, the unit of linguistic analysis is the word. The models in prior work are based on digrams (character pairs). Second, the motor component of the model must accommodate two thumbs rather than a single finger or stylus. Fitts’ law is used to predict movement times for individual thumbs where applicable; however, cases where successive keys are pressed by opposing thumbs are modelled with a different approach (described below).

Each step above is detailed in the following sections, but before we continue, some comments about Fitts’ law are necessary.

5.2.1 Fitts’ law

Fitts’ law predicts the time required for a person to make a rapid aimed movement (Fitts 1954; Fitts et al. 1964). It is expressed as,
\[ t_{\text{Fitts}} = a + b \times ID , \]  

(14)

where \( t_{\text{Fitts}} \) is the movement time (measured in seconds), \( ID \) is the index of difficulty of the movement task (measured in bits), and \( a \) and \( b \) are constants, obtained through experimentation and linear regression.

The index of difficulty reflects the difficulty of the movement task, and it is defined by the Shannon formulation (MacKenzie 1992) to be,

\[ ID = \log_2 \left( \frac{A}{W} + 1 \right) , \]  

(15)

where \( A \) and \( W \) define the physical characteristics of the movement task. Specifically, \( A \) is the amplitude of the task (the distance from the starting position to the centre of the target area), and \( W \) is the width of the target (preferably, but not necessarily, measured tangential to the direction of motion). The units for both \( A \) and \( W \) are distance (in our case, millimetres).

Previous work has established the applicability of Fitts’ law to single finger typing tasks, where \( A \) is the distance between two keys of a keyboard, and \( W \) is the width of the target key; specifically, Soukoreff and MacKenzie (in Chapter 2, and 1995) and Silfverberg et al. (2000) used Fitts’ law in their text entry models.

We will measure numerical values for \( a \) and \( b \) with an experiment presented later in this chapter, however, values for single thumb typing were published by Silfverberg et al. (2000). They found \( a = 0.176 \) seconds, and \( b = 0.064 \) seconds per bit, for single-thumb typing on a cellular telephone. We expect to measure similar values.

A final point about Equation 14 is necessary. The intercept, \( a \), has units of time (seconds) and represents the amount of time required to perform a movement task of no distance, viz. where \( A \) is zero. The physical interpretation of this is the time to
perform a repeat key press (as when typing the second “o” in the word “look”, because the thumb is already hovering over the “o” key after typing the first “o”). We will return to this point later.

5.3 Word-frequency list

Our intention is for a model to predict the average typing speed for common English. Previous models of text entry have accomplished this by using digram or trigram based language models (for example, Chapter 2, Silfverberg et al. 2000; Soukoreff et al. 1995). Digrams and trigrams are not suited to our model (for reasons that will soon become apparent), so instead we employ a word-level model of English, based in the British National Corpus.

The British National Corpus is a collection of 4,124 texts, of which 863 are transcribed from spoken conversations or monologues. The following description is taken from the British National Corpus home page (http://info.ox.ac.uk/bnc/):

The Corpus is designed to represent as wide a range of modern British English as possible. The written part (90%) includes, for example, extracts from regional and national newspapers, specialist periodicals and journals for all ages and interests, academic books and popular fiction, published and unpublished letters and memoranda, school and university essays, among many other kinds of text.

Our word-frequency list contains the 9022 most common words in the British National Corpus. It is the same list used by Silfverberg et al. (2000) in developing their text entry model for mobile phone keypads. The frequencies of the words total

26 Although a model of the English language is employed, any language could be modelled by using a suitable corpus.
67,962,112. The shortest word is “a” (frequency: 1,939,617), while the longest word is “telecommunications” (18 letters, frequency: 1221). The average word size is 7.088 characters if a simple mean is calculated, or 4.427 characters if weighted by the word frequencies.

Although our model’s predictions are generated using a word-frequency list, digram-frequency and letter-frequency lists are also useful to facilitate certain analyses, for example, space key usage and word transitions. Both digram and letter frequency lists are easily built from the word-frequency list. The letter-frequency list has 27 letters (a-z, and space) with frequencies totalling 368,832,032. The digram-frequency list has $27 \times 27 = 729$ unique digrams, with frequencies again totalling 368,832,032. Table 7 presents frequency information for the space character calculated using the letter-frequency list.

<table>
<thead>
<tr>
<th>Letters</th>
<th>Frequency</th>
<th>% of Letters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>67,962,112</td>
<td>18.43%</td>
</tr>
<tr>
<td>All others</td>
<td>300,869,920</td>
<td>81.57%</td>
</tr>
<tr>
<td>Total</td>
<td>368,832,032</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

As seen in Table 7, spaces constitute about 18.43% of all letters. Similarly, 18.43% of all digrams are of the form space-letter (start of word), and an additional 18.43% of all digrams are of the form letter-space (end of word). It is useful to partition the start-of-word and end-of-word digrams into two groups – those involving a letter usually found on the left side of the Qwerty keyboard, and those from the right side. The results are shown in Table 8 and Table 9.
As seen in Table 8, 65.75% of words begin with a letter on the left side of the keyboard, with the remaining 34.25% beginning with a letter on the right side. A similar breakdown for word endings is seen in Table 9. 70.49% of words end with a letter on the left, while 29.51% end with a letter on the right. Thumb-to-key assignments are discussed in more detail shortly.

### 5.4 Digitised keyboard

Digitising a keyboard is straightforward. Working with an image of a keyboard, the x-y coordinate and the size of each key is measured and entered into a table along with the letter assigned to the key. For rectangular or elliptical keys, the smaller of the width and height dimensions is entered as the size of the key, as suggested in prior Fitts’ law research (MacKenzie et al. 1992). The units are arbitrary. Our
measurements were gathered using the pixel coordinates of an image processing application.

We used the Sharp *EL-6810* as a representative keyboard for testing our model (see Figure 35a). The digitised rendering is shown in Table 10.

<table>
<thead>
<tr>
<th>Letter</th>
<th>X Coordinate</th>
<th>Y Coordinate</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>46.0</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>w</td>
<td>119.4</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>e</td>
<td>192.8</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>r</td>
<td>266.2</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>t</td>
<td>339.6</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>y</td>
<td>413.0</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>u</td>
<td>486.4</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>i</td>
<td>559.8</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>o</td>
<td>633.2</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>p</td>
<td>706.6</td>
<td>314</td>
<td>35</td>
</tr>
<tr>
<td>a</td>
<td>80.0</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>s</td>
<td>153.4</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>d</td>
<td>226.8</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>f</td>
<td>300.2</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>g</td>
<td>373.6</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>h</td>
<td>447.0</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>j</td>
<td>520.4</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>k</td>
<td>593.8</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>l</td>
<td>667.2</td>
<td>366</td>
<td>35</td>
</tr>
<tr>
<td>z</td>
<td>118.0</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>x</td>
<td>191.4</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>c</td>
<td>264.8</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>v</td>
<td>338.2</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>b</td>
<td>411.6</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>n</td>
<td>485.0</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>m</td>
<td>558.4</td>
<td>418</td>
<td>35</td>
</tr>
<tr>
<td>_</td>
<td>416.0</td>
<td>470</td>
<td>35</td>
</tr>
</tbody>
</table>
5.5 Assignment of thumbs to letters and keys

To determine the assignment of thumbs to letters and keys, a few assumptions are necessary. A reasonable assumption is that each thumb presses keys normally pressed by the corresponding hand during touch typing. This is illustrated in Figure 36.

![Figure 36 - Partition of keyboard for left and right thumbs](image)

Although it is uncertain whether the thumb assignments in Figure 36 occur in practice, this is a reasonable start. Changes are easily introduced, if necessary, to accommodate different thumb-to-key assignments. Given the assignments in Figure 36, we now know which thumb is used to enter each letter (the space key will be discussed in the next section). Figure 37 shows an example where L represents the left thumb, and R the right thumb.

```
Letter:  t h e _ q u i c k _ b r o w n _ f o x
Thumb:   L R L R L R L R L L L R L R L L R L L
```

![Figure 37 - Example phrase with thumb assignment](image)

See text for discussion of space key usage.
5.5.1 Space key policy

The size and position of the space key varies among devices with miniature keyboards. If the space key is centrally located as with standard keyboards, then it is equally accessible to the right or left thumb. Since spaces constitute about 18% of English text entry, it is important to embed in our model an appropriate behavioural description of space key usage. We call this the space key policy. The following three space key policies seem tenable.

Alternate Thumb. One possibility is that the space key is activated by the alternate thumb to that used for the last letter in a word. This is a common behaviour exhibited by many, when two-thumb typing. This behaviour is shown in the example in Figure 37.

There is a certain optimality to the alternate thumb policy. For example, for two-handed touch typing, keying time is reduced when the preceding key is pressed by a finger on the opposite hand (Kinkead 1975). (To be truly optimal, however, the first letter in the following word would also have to be considered. This will not be considered at the present time.)

Left Thumb. The left thumb space key policy assumes simply that the space key is always pressed by the left thumb.

Right Thumb. With a right-thumb space key policy, the space key is always pressed by the right thumb.

The left-thumb and right-thumb space key policies are particularly appealing if the space key is positioned on either the left or right side or the keyboard, as seen, for example, in Figure 35b and Figure 35d where the space key is on the left. When modelling these cases, one should adopt a left-thumb space key policy.
5.5.2 Thumb transitions

Given our three space key policies and our assumptions of the assignment of thumbs to letters and keys, it is possible to categorize two-thumb text entry by thumb transitions for each digram in our corpus. This is shown in Figure 38.

Figure 38 - Thumb transitions by space key policy
(a) alternate thumb, (b) left thumb, (c) right thumb
The ratios in Figure 38 are of 368,832,032 total frequencies in the digram-frequency list cited above. Among the insights in Figure 38 is the identification of key actions characterized by Fitts’ law. These are the key sequences “Left-Left” or “Right-Right”. For the alternate thumb space key policy (Figure 38a), about 36.8% of the actions are of this type, whereas 63.2% of the key actions are of the form “Left-Right” or “Right-Left”. Also note that the proportions appearing for the alternate space key policy (Figure 38a) are similar to those of the right thumb policy (Figure 38c). Our method of modelling the key actions and thumb transitions is explained in the next section.

5.6 Predicted entry times

A few comments on key repeat times and thumb alternation are necessary before we can launch into the description of our two-thumb typing model.

5.6.1 Key repeat

In the section above that introduced Fitts’ law (section 5.2.1), Equation 14 was described as predicting the time taken to move a thumb from one key to another. However, as previously mentioned, when typing repeat keys, no movement is necessary. There are two approaches to modelling this. In Chapter 2 (and in Soukoreff et al. 1995) we chose to define a constant, \(t_{\text{REPEAT}}\), that represents the time to repeat a keystroke. A different approach was taken by Silfverberg et al. (2000), who used Fitts’ law to model all motions – even the repeat key strokes, so the intercept from the Fitts’ law regression model effectively became the key repeat time. Both approaches have merits, and this issue remains undecided. For the moment, we will model key repeat with Fitts’ law in the manner of Silfverberg et al., however, we will return to this point later.

Table 11 lists several values reported in the literature for key repeat time. Most of the values in Table 11 are for index fingers, and the single value for thumb repeat
time is for single thumb typing on a cellular telephone, while holding the phone in
the hand doing the typing. Although we expect a two thumb key repeat time to be
similar to the single thumb value, we will measure key repeat time with an
experiment described later in this chapter.

Table 11 - Values for $t_{\text{REPEAT}}$ from other publications

<table>
<thead>
<tr>
<th>Value (seconds)</th>
<th>Publication</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.127</td>
<td>MacKenzie &amp; Zhang, 1999</td>
<td>Index finger</td>
</tr>
<tr>
<td>0.140</td>
<td>Card et al., 1983</td>
<td>Standard keyboard</td>
</tr>
<tr>
<td>0.153</td>
<td>Soukoreff &amp; Mackenzie, 1995</td>
<td>Stylus on touchscreen</td>
</tr>
<tr>
<td>0.154</td>
<td>Hughes et al., 2002</td>
<td>Stylus on touchscreen</td>
</tr>
<tr>
<td>0.165</td>
<td>Silfverberg et al., 2000</td>
<td>Index finger</td>
</tr>
<tr>
<td>0.176</td>
<td>Silfverberg et al., 2000</td>
<td>Thumb (dominant-hand)‡</td>
</tr>
</tbody>
</table>

‡ Subjects used the thumb of their choosing, right-handed subjects used their
right thumb, most left handed subjects used their left thumb.

5.6.2 Thumb alternation

We shall use $t_{\text{MIN}}$ to represent the minimum time between keystrokes when
alternating thumbs (that is, typing a letter with one thumb, followed immediately by
typing another letter with the other thumb). The value of $t_{\text{MIN}}$ can be approximated
with

$$t_{\text{MIN}} \approx \frac{1}{2} \times t_{\text{REPEAT}}$$

(16)

where $t_{\text{REPEAT}}$ is the time to press one key repeatedly with the same finger. The
rationale for this approximation is based on research in two-handed touch typing, as
reported in Card et al. (1983, page 60). The idea is depicted in Figure 39. The time
between keystrokes when using one thumb to repeatedly type the same key is $t_{\text{REPEAT}}$
(depicted in Figure 39a). When using two thumbs to repeatedly alternate between
two keys, the keystroke rate almost doubles because the movement of the two
thumbs overlaps (Figure 39b). The value of \( t_{\text{MIN}} \) will be discussed and measured later in this chapter.

![Diagram showing thumb motion and repeat time](image)

**Figure 39** - Illustration of key repeat time
(a) single thumb (b) alternating thumbs

5.6.3 Calculating predicted entry time

Our method to compute the predicted entry time for each word is explained through an example. Figure 40 illustrates an arbitrary sequence of letters followed by space, entered as \texttt{LLRLRL}. Each circle represents a keystroke. Entry proceeds left-to-right as two separate coordinated streams of input, one for the left thumb (top line) and one for the right thumb (middle line). The combined effect is shown in the bottom line. The time to enter the word is \( t_6 \).
The open circles on the left represent the space character terminating the previous word. Since our model considers words only, and is based on a specific space key policy (see above), we do not know which thumb was used for the space key preceding a word. However, this uncertainty can be accommodated as follows. Our earlier analysis of end-of-word digrams reveals that 70.49% of words end with a letter on the left side of the layout in Figure 36. Based on our space key policy, this implies that 70.49% of the time, the space key is pressed by the right thumb, and 29.51% of the time the space key is pressed by the left thumb.

We can use the values just cited as weighting factors in determining $t_1$. The example word in Figure 40 begins with a left-thumb keystroke. If the left thumb was used for the preceding space, the movement time for first letter is $t_{FITS}$, where $t_{FITS}$ is the time for the left thumb to move to and press the key bearing the first letter in the word, having just pressed the space key. If the right thumb was used for the preceding space, we assume the left thumb is poised to enter the first letter with negligible movement. In this case, movement time is $t_{MIN}$. We combine these descriptions with the weighting factors to accommodate uncertainty on which interaction takes place. Since the example word in Figure 37 begins with a left-thumb keystroke, we use

$$t_1 = 0.2951 \times t_{FITS} + 0.7049 \times t_{MIN}$$  \hspace{1cm} (17)$$

For words beginning with a right-thumb keystroke, we use the same formula except the weighting factors are reversed.
Time $t_2$ in Figure 40 is simply

$$t_2 = t_1 + t_{\text{FITTS}}$$  \hspace{1cm} (18)

where $t_{\text{FITTS}}$, in this case, is the time for the left thumb to move to and acquire the key bearing the second letter, having just entered the first. A similar calculation is used throughout a word if the same thumb is used for the preceding letter.

The third letter in the example is entered with the right thumb. There is again uncertainty on the preceding interaction. For the sequence in Figure 40, we use

$$t_3 = \max(t_2 + t_{\text{MIN}}, t_0 + t_{\text{FITTS}})$$  \hspace{1cm} (19)

In this case, $t_{\text{FITTS}}$ is the time for the right thumb to press the key bearing the third letter having previously pressed the space key (which occurs at $t_0$ in the example). At the very least, $t_3$ should be $t_2 + t_{\text{MIN}}$, so we choose the maximum of these two possibilities. A similar calculation is used throughout a word if the opposite thumb was used for the preceding letter (i.e., thumb alternation).

To complete the example,

$$t_4 = \max(t_3 + t_{\text{MIN}}, t_2 + t_{\text{FITTS}})$$  \hspace{1cm} (20)

$$t_5 = \max(t_4 + t_{\text{MIN}}, t_3 + t_{\text{FITTS}})$$  \hspace{1cm} (21)

$$t_6 = \max(t_5 + t_{\text{MIN}}, t_4 + t_{\text{FITTS}})$$  \hspace{1cm} (22)

This completes our example walk-through for the key sequence in Figure 40. Restating the procedure in general terms, for the first letter in a word, we use

$$t_1 = 0.2951 \times t_{\text{FITTS}} + 0.7049 \times t_{\text{MIN}}$$  \hspace{1cm} (23)

if entered with the left thumb, or
\[ t_1 = 0.7049 \times t_{\text{FITTS}} + 0.2951 \times t_{\text{MIN}} \] \hspace{1cm} (24)

if entered with the right thumb. For subsequent letters, we use

\[ t_n = t_{n-1} + t_{\text{FITTS}} \] \hspace{1cm} (25)

if the same thumb is used for the previous letter, or

\[ t_n = \max(t_{n-1} + t_{\text{MIN}}, t_{\text{RECENT}} + t_{\text{FITTS}}) \] \hspace{1cm} (26)

if the opposite thumb is used for the previous letter. The time stamp of the most recent use of the same thumb is represented by \( t_{\text{RECENT}} \), which precedes the current keystroke by at least two keystrokes. Of the four equations above, equation 26 is used most often (about 57% of the time). It is for this reason – the need to consider more than one preceding keystroke – that our model is based on words rather than digrams.

### 5.6.4 Model coefficients

An important component of the model is missing. Fitts’ law models have not been reported for pressing keys with thumbs, as shown in Figure 34. Two models are needed: one for the preferred hand, and one for the non-preferred hand. A related model is reported by Silfverberg et al. (2000) for the thumb on the preferred hand pressing keys on a mobile phone keypad:

\[ MT = 176 + 64 \times \log_2(A / W + 1) \] \hspace{1cm} (27)

where \( A \) is the amplitude of the movement and \( W \) is the width of the destination key. We can tentatively use this model for both thumbs. As well, by Equation 27, \( t_{\text{REPEAT}} = 176 \text{ ms} \) (the intercept of the Fitts’ law equation represents the time to complete a movement of zero length, or in this case, a repeat key press). So, a tentative value for \( t_{\text{MIN}} \) is 88 ms.
We will revisit these values later in this chapter.

5.7 Model predictions

With these model coefficients and the behavioural description above, all the components of the model are in place. A Java program was written to generate a prediction, as per the procedure and coefficients just described. The program works with a space key policy, a word-frequency list and a digitised rendition of a keyboard. Our default invocation uses the alternate thumb space key policy, the 9022 word-frequency list from the British National Corpus, and a digitisation of the Sharp EL-6810 keyboard in Figure 35a. Our program provides the following prediction for two-thumb text entry:

\[ t_{WPM} = 60.74 \text{ wpm} \]  

Previous predictions for key-based mobile text entry fall into the range 20.8 wpm to 45.7 wpm (MacKenzie, Kober, Smith, Jones and Skepner 2001a; Silfverberg et al. 2000). Although our prediction of 60.74 wpm seems quite high, it is important to remember that it is a peak rate for experts, and it is for dual-stream input using two thumbs. Rates of 80 wpm, or beyond, are readily attained by expert touch typists on standard keyboards, so our prediction is not unreasonable.

5.8 Sensitivity analysis

There are numerous factors influencing our model’s prediction. A useful exercise, therefore, is to test the sensitivity of the model to changes in the components and parameters contributing to the prediction. Such an exercise is known as a sensitivity analysis. For examples, see (Card et al. 1983; Silfverberg et al. 2000).
5.8.1 Slope coefficient

A good start is to vary the slope coefficient in the Fitts’ law model and observe the effect on the model’s predictions. Earlier we tentatively used Silfverberg et al.’s (2000) model for pressing keys with the thumb, using the same model for both thumbs. The slope coefficient in their model is 64 milliseconds per bit (see Equation 27). Table 12 illustrates the effect of systematically altering the slope coefficient. For this, we generated six additional predictions: three with higher slope coefficients (+10%, +20%, and +50%) and three with lower slope coefficients (-10%, -20%, and -50%).

<table>
<thead>
<tr>
<th>Slope Coefficient (ms/bit)</th>
<th>WPM Prediction</th>
<th>% of Nominal</th>
<th>% of Nominal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Value</td>
<td>% of Nominal</td>
<td>% of Nominal</td>
</tr>
<tr>
<td>32.0</td>
<td>76.44</td>
<td>50%</td>
<td>125.8%</td>
</tr>
<tr>
<td>51.2</td>
<td>66.18</td>
<td>80%</td>
<td>109.0%</td>
</tr>
<tr>
<td>57.6</td>
<td>63.35</td>
<td>90%</td>
<td>104.3%</td>
</tr>
<tr>
<td>64.0‡</td>
<td>60.74‡</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>70.4</td>
<td>58.34</td>
<td>110%</td>
<td>96.0%</td>
</tr>
<tr>
<td>76.8</td>
<td>56.12</td>
<td>120%</td>
<td>92.4%</td>
</tr>
<tr>
<td>96.0</td>
<td>50.37</td>
<td>150%</td>
<td>82.9%</td>
</tr>
</tbody>
</table>

‡ Nominal values

The relationship is inverse, as expected, since increasing the slope coefficient increases the predicted Fitts’ law movement time which, in the end, reduces text entry throughput in words per minute. A 10% change in the slope coefficient, for example, yields a change of about 4% - 5% in the word-per-minute prediction. This
effect is readily seen in Figure 41. The 50% increase and decrease in slope coefficient values represent extremes that are presented for completeness. Reasonable (up to ±20%) variation of the slope results in a less than 10% change in our nominal prediction.

![Figure 41 - Sensitivity to the Fitts’ law slope coefficient](image)

Dashed line shows nominal value.

5.8.2 Sensitivity to $t_{\text{MIN}}$

Our model makes frequent use of $t_{\text{MIN}}$, the assumed minimum time between key presses with alternate thumbs. We nominally set $t_{\text{MIN}} = 88$ ms, or one half the intercept in the Fitts’ law equation, as explained earlier. However, it is not clear that users will exhibit such behaviour during normal or high speed text entry. And so, examining the influence of $t_{\text{MIN}}$ on the model is worthwhile. Table 13 shows this influence, replicating the procedure in the preceding section.
Clearly the influence is much less than for the slope coefficient. Changes of ±10% yield just a 1% change in the word-per-minute prediction produced by the model. Even changes of ±50% in the slope coefficient yield changes of only about 5% in the predicted text entry rate. The effects are more-clearly seen in Figure 42.

<table>
<thead>
<tr>
<th>Value</th>
<th>% of Nominal</th>
<th>Value</th>
<th>% of Nominal</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.0</td>
<td>50%</td>
<td>63.84</td>
<td>105.1%</td>
</tr>
<tr>
<td>70.4</td>
<td>80%</td>
<td>61.97</td>
<td>102.0%</td>
</tr>
<tr>
<td>79.2</td>
<td>90%</td>
<td>61.36</td>
<td>101.0%</td>
</tr>
<tr>
<td>88.0‡</td>
<td>-</td>
<td>60.74‡</td>
<td>-</td>
</tr>
<tr>
<td>96.8</td>
<td>110%</td>
<td>60.12</td>
<td>99.0%</td>
</tr>
<tr>
<td>105.6</td>
<td>120%</td>
<td>59.51</td>
<td>98.0%</td>
</tr>
<tr>
<td>132.0</td>
<td>150%</td>
<td>57.47</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

‡ Nominal values
5.8.3 Key widths

As well as sensitivity to the Fitts’ law coefficients, our model is sensitive to the assumed width of the keys, which is confounded with the width of the thumb. Our model uses the key heights as $W$ in the model, since key height is the smaller of the width and height dimensions of the keys. This assignment for target width was used by Soukoreff et al. (1995) and Silfverberg et al. (2000) and is recommended in prior Fitts’ law research (MacKenzie et al. 1992). However, the input “device” is a thumb, not a stylus, so the “effective key width” may be somewhat larger if we also consider the width of the thumb. This was noted by Drury (1992) in a study of keying times on calculators with various inter-key gaps. If the assumed key widths are increased by 10%, 20%, and 50%, for example, the word-per-minute prediction increases by 1.9% (61.89 wpm), 3.7% (62.96 wpm), and 8.3% (65.76 wpm), respectively.
5.8.4 Corpus effect

We used the same word-frequency list as Silfverberg et al. (2000). To test for a possible “corpus effect” we also generated predictions with three other word-frequency lists. The first is a much larger list from the British National Corpus that includes numerous additional low-frequency words. The second is a word-frequency list derived from the Brown Corpus (available from numerous on-line sites). The third is a word-frequency list derived from a set of 500 phrases constructed in-house for our text entry evaluations. The results are given in Table 14.

Clearly, the corpus effect is minimal. The first two additional predictions are extremely close to the original prediction of 60.74 wpm. Even the prediction generated with the very limited word-frequency list from our phrase set is within 2% of the nominal value.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Unique Words</th>
<th>Total Frequencies</th>
<th>WPM Prediction</th>
<th>% of Nominal</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNC1</td>
<td>9,022</td>
<td>67,962,112</td>
<td>60.74‡</td>
<td>-</td>
</tr>
<tr>
<td>BNC2</td>
<td>64,588</td>
<td>90,563,847</td>
<td>60.21</td>
<td>99.1%</td>
</tr>
<tr>
<td>Brown</td>
<td>41,532</td>
<td>997,552</td>
<td>60.18</td>
<td>99.1%</td>
</tr>
<tr>
<td>Phrases</td>
<td>1,163</td>
<td>2,712</td>
<td>59.81</td>
<td>98.5%</td>
</tr>
</tbody>
</table>

‡ Nominal value

5.8.5 Space key policy

Our nominal prediction assumes a specific policy on space key usage; namely, that the user always presses the space key with the alternate thumb from that used for the last letter in a word. Again, it is not clear that this will occur in practice. We
generated word-per-minute predictions for the two other space key policies described earlier. The results are shown in Table 15.

<table>
<thead>
<tr>
<th>Space Key Policy</th>
<th>WPM Prediction</th>
<th>% of Nominal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate thumb</td>
<td>60.74‡</td>
<td>-</td>
</tr>
<tr>
<td>Left thumb</td>
<td>49.92</td>
<td>82.19%</td>
</tr>
<tr>
<td>Right thumb</td>
<td>56.54</td>
<td>93.09%</td>
</tr>
</tbody>
</table>

‡ Nominal value

Using the alternate thumb policy is highly preferential to the other two policies. There are significant performance costs (7 - 18%) depending on whether the left or right thumb policy is used. We consider these differences more the result of changes in user behaviour, rather than a sensitivity effect in the model. Importantly, this exercise demonstrates the utility of our model for a priori analyses.

The predictions in Table 15 are for the Sharp EL-6810 keyboard which includes a centrally located space key (see Figure 35a). If the space key is offset to the left or right, then the effect of space key policy may be different. For example, the keyboards on the Motorola PageWriter 2000 two-way pager (Figure 35b) and the Nokia 5510 mobile phone (Figure 35d) position the space key to the left of centre. The effect of space key positioning and space key policy are the focus of continuing work in modelling two-thumb text entry.

5.9 Measuring coefficients for the model

Up to this point the values for the Fitts’ law model and for $t_{\text{MIN}}$ and $t_{\text{REPEAT}}$ were
taken from figures published by Silfverberg et al. (2000). However, the Silfverberg study was of single-thumb text entry using a cellular telephone. The difference in form factor between a keypad on a mobile phone and the miniature Qwerty keyboard on the Sharp EL-6810 causes some concern. An experiment was carried out to construct Fitts’ law models of both thumbs during text entry using a miniature Qwerty keyboard. Values for $t_{\text{MIN}}$ and $t_{\text{REPEAT}}$ were also measured.

5.9.1 Materials and method

5.9.1.1 Participants

Nine volunteers participated in this study (four females, five males). They ranged in age from 25 to 32, with an average age of 29.7 years. Eight were right-handed; one was left-handed (as reported by the subjects).

5.9.1.2 Apparatus

A Sharp EL-6053 pocket organiser was used for this experiment. This is not the same device described earlier (the Sharp EL-6810 organizer, shown in Figure 35a), but it is physically indistinguishable from the EL-6810. In particular, the miniature Qwerty keyboards on the devices are physically identical.

A PIC micro-controller (Microchip Technology, http://www.microchip.com/) was interfaced to the keyboard hardware of the EL-6053, and programmed to emit ASCII characters in real time as keys were typed on the keyboard. The PIC micro-controller and associated circuitry were too large to fit inside the case of the EL-6053 and so they were bolted onto the back of the device, with a flat cover bolted over the circuitry. Due to the added circuitry, the thickness of the device was increased to 39 mm. See Figure 43. The EL-6053 is a clam-shell device with a protective cover; the cover was removed for the experiment.
The ASCII characters from the PIC micro-controller were transmitted through a serial cable at 1200 baud to a 400MHz Pentium II computer. A Java program on the Pentium computer time-stamped and recorded the ASCII characters. The Java program provided the look-and-feel of a simple text editor, so subjects typing on the keyboard could receive visual feedback and confirmation of their keystrokes. See Figure 44. (Note that the LCD screen of the EL-6053, visible in Figure 43, was not used.) Particular attention was paid to lag to ensure the accuracy of the final time-stamps.
5.9.1.3 Procedure

Subjects sat in front of the monitor of the Pentium computer and were encouraged to arrange themselves in a comfortable position. The chair was adjusted if required. They held the device in both hands in a manner similar to that in Figure 34. Subjects were not allowed to place the device on the table in front of them (the table supported the monitor of the Pentium computer), however they were allowed to rest their arms in any comfortable manner they chose.

The subjects were instructed to use both thumbs to perform a series of artificial typing tasks lasting ten seconds each. We say “artificial” because subjects did not enter English text; rather, the tasks were to enter repeating patterns of characters necessary for the measurement of the desired Fitts’ law values (the character patterns are described below). The subjects were allowed to stop to rest or adjust themselves throughout the experiment. The specific instructions given to subjects were to enter the repeating pattern of characters as fast as possible, without making any errors. However, subjects were also told to ignore any errors they made; in the event of an error, they were simply to continue the movement task at hand.

5.9.1.4 Design

Several tasks were used to measure the required movement times over a range of movement distances. Descriptions of the movement tasks are organised into sections below, one for each parameter of the model.

5.9.1.5 The Fitts’ law models

The subjects performed a series of single thumb alternation tasks, over index of difficulty ($ID$) values ranging from 1.60 to 3.71 bits. The tasks were chosen to cover a representative range of $ID$ values, and to cover a range of directions of motion. The specific keystroke patterns used are listed in Table 16. For example, to determine the time to perform a task with a difficulty of 1.60 bits with their left
thumb, the subject was asked to enter the pattern: “DEDEDEDEDEDE…” as quickly as possible for ten seconds, using only their left thumb. An experimenter was present and monitoring the subject throughout these trials to ensure the subjects used the correct thumb for each task. Note that the experimental software stopped collecting data once the first keystroke past the ten second mark was entered, therefore the total time for one trial was slightly longer than ten seconds. The tasks were presented to the subjects in a random (not counter-balanced) order.

To calculate the index of difficulty values appearing in Table 16, Fitts’ law amplitudes ($A$) and widths ($W$) had to be assigned. The keys on the keyboard measured $8 \times 5$ mm; the target widths were taken to be the smaller of the two dimensions – so $W = 5$ mm. (This is identical to the approach taken by other researchers (Hunter et al. 2000; MacKenzie et al. 1999b; MacKenzie et al. 1999c; Soukoreff et al. 1995; Zhai et al. 2000; Zhai et al. 2002; Zhang 1998). The amplitudes were measured directly from the keyboard.

<table>
<thead>
<tr>
<th>ID (bits)</th>
<th>Left Hand Patterns</th>
<th>Right Hand Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.60</td>
<td>D-E</td>
<td>J-I</td>
</tr>
<tr>
<td>1.73</td>
<td>D-F</td>
<td>J-H</td>
</tr>
<tr>
<td>2.12</td>
<td>E-X</td>
<td>I-M</td>
</tr>
<tr>
<td>2.50</td>
<td>D-G</td>
<td>H-K</td>
</tr>
<tr>
<td>2.99</td>
<td>S-G</td>
<td>H-L</td>
</tr>
<tr>
<td>3.12</td>
<td>T-Z</td>
<td>M-T</td>
</tr>
<tr>
<td>3.71</td>
<td>Q-B</td>
<td>P-V</td>
</tr>
</tbody>
</table>

5.9.1.6 Measuring $t_{\text{REPEAT}}$

Two tasks were used to measure $t_{\text{REPEAT}}$, one task for each thumb. For the left
thumb, subjects were asked to repeatedly type the D key using the left thumb for ten seconds. Similarly, the J key was typed with the right thumb. The trial ended upon the first keystroke after 10 seconds.

5.9.1.7 Measuring $t_{\text{MIN}}$

One task was used to measure $t_{\text{MIN}}$. The subjects were asked to alternately type D with the left thumb and J with the right (so the subjects typed “DJDJDJ...”), for ten seconds. Again, the software ended the trial once the first keystroke past the 10 second mark was entered.

5.9.2 Results

5.9.2.1 The Fitts' law models

The average time required to perform each task listed in Table 16 was measured from the time-stamped keystroke data. Although subjects were instructed not to make errors, upon inspection of the data some errors were apparent. To overcome this, some pre-processing of the data was necessary. A thumb transition was defined as a movement from a key near one target, to another key nearer the other target. In this way, keys near (but possibly other than) the intended key were accepted as evidence that the thumb had moved from one target to another. Duplicate key presses were ignored. By counting the transitions, an estimate of the number of rapid aimed movements was made.

As well as the number of transitions, the total time (approximately ten seconds) for the series of thumb transitions was measured.

For example, when subject #1 completed the 1.60 bit movement task with his right hand, he made 58 alternations between the J and I keys in 10.380 seconds. The average movement time was $10.380 / 58 = 178.966$ milliseconds. The movement times for all subjects were averaged, and linear regression was used to calculate the
slope and intercept Fitts’ law constants. Note that the intercept data values (where the index of difficulty is 0 bits) were not used in the linear regression. The data and regression lines appear in Figure 45.

![Figure 45 - Fitts’ law data and regression lines](image)

The Fitts’ law models appear in Table 17. The correlations were high for both thumbs, indicating that Fitts’ law predicts the movement time with high accuracy for both thumbs.

As previously mentioned, the ID = 0 data points were not included in the linear regression used to calculate the constants for the Fitts’ law models. However, for completeness, in Table 17 we report the results of linear regression including the ID = 0 data points as well. Note that the regression correlation values are lower for the models that include the ID = 0 data point.
Table 17 - Fitts’ law models

<table>
<thead>
<tr>
<th>Model</th>
<th>Intercept (ms)</th>
<th>Slope (ms / bit)</th>
<th>Throughput (bits / s)</th>
<th>n</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant Thumb:</td>
<td>98.53</td>
<td>92.02</td>
<td>10.87</td>
<td>7</td>
<td>0.976</td>
</tr>
<tr>
<td>including ID = 0 data:</td>
<td>153.14</td>
<td>72.11</td>
<td>13.86</td>
<td>8</td>
<td>0.962</td>
</tr>
<tr>
<td>Non-Dominant Thumb:</td>
<td>98.62</td>
<td>98.79</td>
<td>10.12</td>
<td>7</td>
<td>0.993</td>
</tr>
<tr>
<td>including ID = 0 data:</td>
<td>170.73</td>
<td>72.50</td>
<td>13.79</td>
<td>8</td>
<td>0.956</td>
</tr>
</tbody>
</table>

It is apparent from Figure 45 that while a wide range of index of difficulty values were used (1.60 to 3.71 bits) there appears to be a gap in the values, from 0 to 1.60 bits. This is an artefact of the physical geometry of the keyboard. Recall that the width of the keys (for the purposes of Fitts’ law) was 5 mm, and that the keys are separated by a 3.33 to 3.6 mm gap. Therefore, the shortest Fitts’ law amplitude possible is $5 + 3 = 8$ mm. These physical characteristics imply that other than the (0 bit) repeat key task, the smallest index of difficulty value possible is 1.60 bits. See Figure 46, and Equation 29.

![Figure 46 - The minimum distance between two keys on our miniature Qwerty keyboard](image)

The physical arrangement of keys depicted in Figure 46, corresponds to an index of difficulty value of 1.60 bits,
\[ ID = \log_2 \left( \frac{A}{W} + 1 \right) = \log_2 \left( \frac{10.15}{5} + 1 \right) = 1.60 \text{ bits}. \] 

(29)

So, when constructing the Fitts’ law models there is a lack of data for low index of difficulty values, and hence we have a lack of confidence concerning predictions in the low index of difficult region. Although circumstantial to our reasoning, note that others have suggested that Fitts’ law does not apply when index of difficulty values are small (Gan and Hoffmann 1988).

5.9.2.2 Measuring \( t_{\text{REPEAT}} \)

The value for \( t_{\text{REPEAT}} \) was calculated by dividing the elapsed time by the number of keystrokes made by the subject. The results from all subjects were averaged. See Table 18.

<table>
<thead>
<tr>
<th>Thumb</th>
<th>( t_{\text{REPEAT}} ) (ms)</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant</td>
<td>181.57</td>
<td>23.74</td>
</tr>
<tr>
<td>Non-Dominant</td>
<td>208.28</td>
<td>31.88</td>
</tr>
<tr>
<td>Average</td>
<td>194.93</td>
<td>-</td>
</tr>
</tbody>
</table>

Although we report two \( t_{\text{REPEAT}} \) values, one for each hand, our intention was to have one value for the model. To obtain this value we average the values measured for the dominant and non-dominant hands. Therefore we calculate,

\[
t_{\text{REPEAT}} = \frac{181.57 + 208.28}{2} = 194.93 \text{ milliseconds.} \tag{30}
\]

5.9.2.3 Measuring \( t_{\text{MIN}} \)

The time-stamped keystroke data recorded to measure \( t_{\text{MIN}} \) suffered from the same
problem that the earlier Fitts’ law data did – subject typing errors. The data were processed in the same way as the Fitts’ law keystroke data. Transitions were counted so an estimate of how many alternating keystrokes were made by each thumb was obtained. The value for $t_{\text{MIN}}$ was calculated by dividing the elapsed time by the number of alternating keystrokes made by the subject. The results from eight subjects were averaged. The final value measured for $t_{\text{MIN}}$ was 117.80 milliseconds, with a standard deviation of 22.67 ms.

The data from one subject were not used in the average because the subject performed extremely poorly at this task, with a measured alternation time that was more than five and a half standard deviations from the mean of the rest of the subjects.

5.9.3 Discussion

5.9.3.1 The Fitts’ law models

The values in Table 17 for the linear regression calculated excluding the $ID = 0$ data point are comparable to those reported by Silfverberg et al. (2000) for single thumb typing. Silfverberg et al. report an intercept value of 176 milliseconds, and a slope value of 64 milliseconds per bit. However, the values in our model are those obtained when the $ID = 0$ data point were not included in the regression model. Unfortunately, Silfverberg does not report values calculated in a similar way. Other reported Fitts’ law models of finger movement do not use the same formulation of Fitts’ law, and so do not provide a useful comparison for our results, for example, Card et al. (1991).

5.9.3.2 Fitts’ law versus $t_{\text{REPEAT}}$

Due to the distribution of the index of difficulty values used to generate the Fitts’ law coefficients, there is less data for small $ID$ values. This may mean that our models do not predict movement times for small $ID$ values very accurately.
We feel that the Fitts’ law predictions for the intercept condition \((ID = 0)\) are too low to be used to model key repeat times. The intercepts are 98.53 milliseconds, and 98.62 ms when the regression does not include the \(ID = 0\) data point, and 153.14 ms and 170.73 ms when the \(ID = 0\) data point is included (see Table 17). These values are lower than the repeat time we measured, 194.93 ms, and our best fitting Fitts’ law values (obtained when the \(ID = 0\) data point is not included in the linear regression) are only slightly more than half of the measured \(t_{repeat}\) value.

Since the regression models match the data well for index of difficulty values greater or equal to 1.60 bits, and a separately calculated \(t_{repeat}\) provides the best fit for the \(ID = 0\) value, we feel that a combined approach should be used. Fitts’ law should be used to model thumb motion where \(ID > 0\) (viz., all thumb motions except key repeats), and \(t_{repeat}\) should be used to model key repeats.\(^{27}\)

5.9.3.3 The value of \(t_{MIN}\)

We initially predicted that the value of \(t_{MIN}\) was approximately 88 milliseconds based upon values reported by Silfverberg et al. (2000), and a certain amount of guesswork (see Equation 16, and the discussion surrounding it). The value we measured experimentally was 117.80 ms with a standard deviation of 22.67 ms. The predicted value lies slightly more than one standard deviation away from our measured value. Although we can not directly test whether our results are statistically significant, we note that the probability of observing a \(t_{MIN}\) value at least as far away as 88 ms is from 117.80 ms, given the standard deviation we measured, is \(P(X \geq 1.31 \mid X \text{ is normally distributed}) = 0.1902\). This value is not excessive. This could be due to some difference in single thumb typing versus two thumb typing, familiarity of the subjects in the Silfverberg et al. study with single thumb typing, or errors in our guesswork.

\(^{27}\) Although this observation is of theoretical importance, repeat keystrokes account for only 1.5% of all keystrokes during entry of typical English text, and so this point is of little practical value.
5.10 The predicted expert typing speed

All the necessary values for the model have been found. We used the same Java program described above to calculate our updated prediction using the two Fitts’ law models we constructed (Figure 45 and Table 17).

The only deviation from the model as described above was the handling of repeat key presses. These were modelled simply by using the $t_{\text{REPEAT}}$ value of 194.93 milliseconds measured experimentally.

Entering the measured parameters and running the program on the British National Corpus data, we arrived at our prediction of 58.06 words per minute for expert typing using our miniature keyboard. This value is only 5% less than the value we arrived at using the Silfverberg Fitts’ law model (2000).

5.11 Conclusions

We have presented a model of two-thumb text entry on a miniature Qwerty keyboards. We have provided a detailed behavioural description of the interaction as well as a predicted rate for English text entry. Our predictions are based solely on the linguistic and motor component of the interaction, and so represent peak rates for expert users.

Our model’s prediction is relatively stable. In a sensitivity analysis, we examined the effect of changes in the various components and parameters that influence the predictions. We generated new predictions after changing various components of the model, including: corpora, assumed key widths (accounting for thumb width), the minimum time between key presses by alternate thumbs, and slope coefficients in the movement time prediction equations. In most cases, the predicted text entry rate changed by less than 10%.
A change in text entry throughput of between 7 and 18% is expected if the user adopts a non-preferential space key policy such as always using the left or right thumb to press the space key. This expectation is coincident with a centrally located space key. The effect may be somewhat different for other keyboard geometries.

5.11.1 Validity of the prediction

At first glance the predicted expert text entry speed of 58.06 wpm seems fast, and without a longitudinal study, it is impossible to validate. However, we have some anecdotal evidence that suggests that it is valid. In preparation for another experiment being planned, the authors have built some software to perform a typing test using typical English phrases. This software presents a subject with a phrase and records keystroke time-stamps and elapsed time as text is entered. The same modified miniature keyboard described earlier has been interfaced to this experimental software, and the author has performed several blocks of trials in the course of testing the software and the miniature keyboard. A typical average text entry speed achieved during a half hour block of typing using the miniature keyboard is 34 wpm, but the fastest trial ever recorded was 48.3 wpm\(^{28}\). Although not an expert, the author has a considerable amount of experience typing with the miniature keyboard, and his fastest entry speed rate suggests that 58.06 wpm is realistic for experts.

5.11.2 Other contributions

We have provided measurements of several key features of two thumb typing such as the time for repeat and alternating keystrokes. We have also provided Fitts’ law models for thumbs on both dominant and non-dominant hands. These measurements describe thumb motion when engaged in two thumb typing on a small personal information manager, equipped with a miniature Qwerty keyboard.

\(^{28}\) An elapsed time of 10.440 seconds was measured while the phrase “travel at the speed of light is a good idea” was entered. This corresponds to a text entry speed of 48.3 wpm.
We have also presented a model of two thumb text entry using these measurements, Fitts’ law, and a word-based model of the English language, that predicts the expert (upper bound) typing rate for two thumb typing. We have used the model to calculate the predicted expert typing rate, 58.06 words per minute.

If an unsuspecting person is handed a device equipped with a miniature keyboard, and they try to enter text while holding the device, there are three ways that that person is most likely to use the device. They will either: hold the device in one hand and use their index finger (as described in Chapter 2, and Soukoreff et al. 1995), hold the device in one hand and use the thumb on the same hand (as described in Silfverberg et al. 2000), or they will use both hands to hold the device, and both thumbs to type. This chapter completes the set, by presenting a model for two thumb text entry.

Table 19 presents a comparison of the maximum predicted text entry rates for the three keyboard-based text entry methods. The slowest rate is achieved by stylus tapping on a Qwerty soft keyboard. The reduced number of keys on a cellular telephone reduces the distance that must be travelled by the thumb (or index finger) during text entry. This is reflected in the predictions for these text entry methods. (The values provided by Silfverberg for index-finger text entry on a cellular telephone are included in Table 19 for completeness.) Note that the figures taken from the Silfverberg et al. paper are for text entry in the presence of T9 disambiguation functionality. The values reported when disambiguation was not used (using the multi-tap and two-key text entry methods) fall into the 20 to 28 wpm range. Cellular telephone keypads are only competitive when coupled with disambiguation. The two-thumb text entry method described and modelled in this chapter achieves the highest expert text entry rate. This is not surprising because the use of two thumbs increases the means of communication, and reduces the average distance from an appendage to the keys.
Table 19 - Comparison of predicted expert typing rates for three methods of mobile text entry

<table>
<thead>
<tr>
<th>Publication</th>
<th>Description</th>
<th>Expert Typing Rate (wpm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soukoreff &amp; MacKenzie 1995</td>
<td>Stylus tapping on a soft Qwerty keyboard</td>
<td>30.0</td>
</tr>
<tr>
<td>Silfverberg et al. 2000</td>
<td>Single-thumb on cell phone with T9 disambiguation</td>
<td>40.6</td>
</tr>
<tr>
<td>Silfverberg et al. 2000</td>
<td>Index finger on cell phone with T9 disambiguation</td>
<td>45.7</td>
</tr>
<tr>
<td><em>This chapter</em></td>
<td>Two-thumb text entry on mini Qwerty keyboard</td>
<td>58.1</td>
</tr>
</tbody>
</table>

5.11.3 Further work

It would be interesting to experimentally observe the thumb-to-key assignments (which thumb is used to type which keys) and the space key policy used by typical miniature keyboard users.

How best to model small Fitts’ law index of difficulties remains an open question, although we have presented fairly convincing evidence that key repeat times should be modelled separately from Fitts’ law.

A question that remains of theoretical interest, but probably of little practical interest, is the optimisation of a keyboard for two-thumb typing.
Chapter 6
Conclusions and Future Work

This chapter presents conclusions and identifies open questions still remaining in the area of mobile text entry.

6.1 Conclusions

6.1.1 The Fitts’ digram model

The first model presented in this thesis – the Fitts’ digram model of text input, was published quite a while ago (Soukoreff et al. 1995). It has been a fascinating experience in the intervening time, to watch the effects that this publication has had upon the text-input research area. Several groups (as reviewed in Chapter 4) have used the model to try to find so-called “optimal” soft keyboard arrangements. This is, perhaps, the best application of the model. The model was not intended to replace user evaluation of soft keyboards; rather, the intent was to provide a fast and effective means to evaluate and compare novel soft keyboard arrangements, deferring costly user evaluation until a promising candidate arrangement was found. This allows researchers to “narrow the field” of possible keyboard arrangements and identify efficacious arrangements worthy of further study. Used in this way, the Fitts’ digram model has been immensely successful.

The Fitts’ digram model is really two models – one of expected expert performance, and one of novice performance. In attempting to verify the accuracy of these two facets of the model, it has become clear that it is difficult to verify the accuracy of the expert prediction, while it is relatively easy to evaluate the novice prediction.
The first difficulty that arises in evaluating the expert model lies in defining the term “expert”, and finding subjects who are experts. Expert performance does not exist as a metric quantifiable by a single value. Expert users continue to improve their performance indefinitely, even after thousands or tens of thousands of hours of practice.

Even if expert subjects are found and an evaluation performed, interpretation of the results is difficult. The prediction reported in Chapter 3 was 30 words per minute for expert stylus typing on a Qwerty soft keyboard. What would it mean if an experiment is performed that results in an expert actually typing 20 wpm? What if an expert types at 40 wpm? Observations such as these do not invalidate the model – they just indicate that the parameters of the model should be revisited, and a new prediction calculated. Regardless, the strength of the expert prediction lies in its application to comparative analyses, not in its ability to predict an absolute numerical prediction.

Evaluating the novice prediction of the model is easier. We employed the Hick-Hyman law for choice reaction time to model the time required for a novice to scan an unfamiliar keyboard searching for the desired key. In order to evaluate this facet of the stylus model, we would require novice subjects – people that are unfamiliar with the keyboard arrangement under study. However, people quickly learn, and the novice experience fades with each successive keystroke. To perform a fair evaluation of the novice model predication, a soft keyboard would have to be constructed that randomly jumbled the arrangement of keys after each keystroke – ensuring that the novice subject remained unfamiliar with the keyboard arrangement as they typed. The results of such a study are reported in Zhang (1998), and MacKenzie and Zhang (2001b). The results indicate that our model does a poor job of modelling novice soft-keyboard stylus typing rates. MacKenzie and Zhang report that, typically, while scanning the keyboard searching for the next key, subjects perform complex movements with the stylus. For example, some subjects
raised their hand holding the stylus above the keyboard so as not to block their view of the keyboard; other subjects moved the stylus back-and-forth over the keyboard following the scanning motions of their eyes. In any case, the simple addition of the Hick-Hyman choice reaction time and the Fitts’ law movement time does not suffice to model novice behaviour. Hughes et al. (2002) report a novel analysis technique that makes it possible to model novice behaviour. There remains work to be done in applying their technique and evaluating the results.

It is worth noting that an empirical study of novice typing on a Qwerty keyboard would be extremely difficult to perform – due to the ubiquity of the Qwerty keyboard, and the widespread familiarity of the population with the Qwerty keyboard arrangement.

6.1.2 The two-thumb typing model

Following the publication of the stylus typing model (Soukoreff et al. 1995), a model of single-thumb text entry on cellular telephones was published (Silfverberg et al. 2000). This model is similar to the stylus typing model, but allowances are made for the specialised text entry techniques employed on cellular telephones, such as multi-tap text entry, or disambiguation.

There are three ways that one is most likely to enter text into a hand-held device. (a) Holding the device in one hand and using an index finger to type. (b) Holding the device in one hand while using the thumb to type. Or, (c) using both hands to hold the device, and both thumbs to type. Models existed for the first two methods of text entry (Silfverberg et al. 2000; Soukoreff et al. 1995), but not the third. Chapter 5 completes the set by presenting a model of two thumb text entry.

The contents of Chapter 5 have only recently been published (MacKenzie et al. 2002b), and so the effects of the model on the text input research community cannot yet be gauged. Even so, the contributions of this work are significant. The model
provides insight into the two-thumb typing activity. Fitts’ law models of both thumbs (on the dominant and non-dominant hands) are constructed and reported. Additionally, the two important non-Fitts’ law activities – key repeating, and thumb alternation, are experimentally measured and reported. All of these observations constitute novel contributions to the literature.

Chapter 5 also presents the results of a sensitivity analysis of the model. This is an interesting and effective way to verify that the model produces a useful characterisation of two-thumb text entry, and that accurate predictions are produced.

6.2 Future work

The text input research area is by no means closed. The recent public interest in all things mobile – cellular telephones, personal digital assistants (PDAs), ubiquitous computing, and miniature and mobile connectivity tools, all guarantee that text entry will remain an important area of study.

It is interesting to note that since the early days of computing, keyboards (miniature or full-sized) have always been at the forefront of text entry. Other technologies have appeared (pen-based computing with handwriting recognition, cording keyboards, speech recognition) and in some cases supplanted the keyboard in certain highly specialised niche application areas, but for general-purpose text entry, the keyboard is here to stay for the foreseeable future. It is worth repeating a point made in the introduction.

Understanding and meeting user expectations is paramount in creating an acceptable text input technology. Users’ expectations for text entry are set by current practice. Touch typing speeds in the range of 20 to 40 words per minute are modest and achievable for
hunt-and-peck typists. Rates in the 40 to 60 words per minute range are achievable for touch typists, and with practice, skilled touch typists can achieve rates greater than 60 words per minute. Handwriting speeds are commonly in the 15 to 25 words per minute range. ... Users, perhaps unrealistically, expect to achieve text input rates within these ranges on mobile devices.

(Excerpt taken from page 8 of this document)

The text entry rates obtainable with a keyboard and a moderate amount of practice are quite fast when compared to many of the new technologies that have come along. However, until a new input technology is created that can match the speed of touch typing, the keyboard will likely retain its preferred position among the text input technologies.

Creation of a technology that can match the ease and speed of speech (172.6 – 197.4 words per minute, Walker 1988) should remain the goal of text input research.

6.2.1 Ongoing work

Although omitted from the preceding paragraphs, there is another element of text entry that is as important as speed – accuracy. In Chapter 1 we presented several ideas concerning error rate measurement using the minimum string distance, and keystrokes per character metrics. This is a promising and neglected area of research. How is it that text input research has come as far as it has without a standardised definition of, and method to calculate, error rate? The minimum string distance and keystrokes per character metrics described in Chapter 1 constituted the most advanced approach to error measurement known at the time that Chapter 1 was written (in 2001). Since that time we have constructed a theory of error measurement and analysis, that continues to be an active research topic for us
at the present time. When complete, this will be an important contribution to the field.
Bibliography


