Poika Isokoski

Manual Text Input: Experiments, Models, and Systems

MANUSCRIPT FOR AN ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Information Sciences of the University of Tampere, for public discussion in the the A1 auditorium on April 23rd, 2004, at noon.

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

A-2004-5
TAMPERE 2004

Supervisor: Professor Roope Raisamo
Department of Computer Sciences
University of Tampere, Finland

Opponent: Dr. Shumin Zhai
IBM Almaden Research Center, USA

Reviewers: Professor Heikki Mannila
Basic Research Unit
Helsinkin Institute for Information Technology, Finland

Professor Ari Visa
Signal Processing Laboratory
Tampere University of Technology, Finland

Department of Computer Sciences
FIN-33014 UNIVERSITY OF TAMPERE
Finland

ISBN
ISSN

Tampereen yliopistopaino Oy
Tampere 2004

Abstract

Despite the emergence of speech controlled computers and direct manipulation that both have diminished the need to operate computers with textual commands, manual text entry remains one of the dominant forms of human-computer interaction. This is because textual communication is one of the main reasons for using computers. We use computers to write rather than write to use computers.

Mobile and pervasive computing have been popular research areas recently. Concordantly, these issues have a major part in the thesis at hand. Most of the text entry methods that are discussed are for mobile computers. The architecture that is one of the three main contributions of the work is a middleware system intended to support personalized text entry in an environment permeated with mobile and non-mobile computers.

The two other main contributions in this thesis are experimental work on text entry methods and models of user performance in text entry tasks.

The text entry methods tested in experiments were the minimal device independent text entry method (MDITIM), two methods for entering numbers using a touchpad, Quikwriting in a multi-device environment, and a menu-augmented soft-keyboard. MDITIM was found to be relatively device-independent, but not very efficient. The numeric entry experiment showed that the clock metaphor works with a touchpad, but with a high error rate. An improved "hybrid" system exhibited a lower error rate. Quikwriting was tested to evaluate the claims on its performance made in the initial publication. The claims were found to be exaggerated, but Quikwriting worked well with the three tested input devices (stylus, game controller, and a keyboard). The menu augmented soft keyboard was compared to a traditional QWERTY soft keyboard to verify modeling results that show significant performance advantages. No performance advantage was observed during the 20 session experiment. However, extrapolations of the learning curves cross suggesting that with enough practice the users might be able to write faster with the menu augmented keyboard.

The results of the modeling part are two-fold. The explanatory power of a simple model for unistroke writing time was measured and a model that combines two previously known models for text entry rate development was constructed.

Acknowledgements

List of publications

This thesis is based on the following research papers:

I Poika Isokoski and Roope Raisamo, Device Independent Text Input: A Rationale and an Example. *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI2000)*, ACM Press, 2000, 76-83. [Isokoski and Raisamo, 2000]

II Poika Isokoski and Mika Käki, Comparison of Two Touchpad-based Methods for Numeric Entry. *CHI2002, Human Factors in Computing Systems, CHI Letters*, 4(1), ACM Press, 2002, 25-32. [Isokoski and Käki, 2002]

III Poika Isokoski and Roope Raisamo, *Evaluation of a Multi-Device Extension of Quikwriting*. Report A-2003-5, Department of Computer Sciences, University of Tampere, Finland, 2003. [Isokoski and Raisamo, 2003b]

IV Poika Isokoski, Performance of Menu-augmented Soft Keyboards. *CHI 2004, Human Factors in Computing Systems, CHI Letters*, 6(1), ACM Press, 2004, (in press). [Isokoski, 2004]

V Poika Isokoski, Model for Unistroke Writing Time. *CHI 2001: Human Factors in Computing Systems, CHI Letters*, 3(1), ACM Press, 2001, 357-364. [Isokoski, 2001]

VI Poika Isokoski and Scott MacKenzie, Combined Model for Text Entry Rate Development, *CHI 2003 Extended Abstracts*, ACM Press 2003, 752-753. [Isokoski and MacKenzie, 2003]

VII Poika Isokoski and Roope Raisamo, Architecture for Personal Text Entry Methods. In Morten Borup Harning and Jean Vanderdonckt (editors), *Closing the Gaps: Software Engineering and Human- Computer Interaction*, IFIP, 2003, 1-8. [Isokoski and Raisamo, 2003a]

Contents

Chapter 1

Introduction

1.1 Context

This thesis is about entering text into computers. In the sense that it is understood in the field of human-computer interaction, text entry began with the emergence of computers during the later part of the 20th century. At about the same time it became a field of technological innovation and topic of scientific study. There have been two waves of text input research activity. One began in the 1970s and another in the 1990s. According to MacKenzie [2002a] the first wave concentrated on desktop computing and the second on pen-based and mobile computing. This thesis belongs to the second wave.

In this thesis I will discuss details of some of the recent developments in text entry. Before that, however, I will briefly introduce some themes that will recur later in the thesis.

One of these recurring issues is the long history of writing and the effects that the established traditions have on text entry research. Writing in general is as old as history itself. The time before writing as we all know is called prehistoric. Throughout times writing systems have interacted with other technology and the societies that have used them. For example the Sumerian cuneiform writing was tightly interwoven with the clay tablet and reed stick technology as well as with the needs of the society. It seems apparent that in this case the writing system evolved to fit the technology. Other cases, such as the Egyptian use of papyrus, exemplify a situation where a whole industry is set up to manufacture material suitable for writing. When a new piece of technology comes along, sooner or later somebody will try to use it for writing. Similarly, when a new writing task emerges, people will try to find the most suitable tools for accomplishing it.

The most influential new piece of technology in our days is the computer. In the light of the historical tendency of trying out new things, it was likely that somebody would try to use the computer for writing. This has indeed happened.

Generally speaking the work in this thesis consists of experiments on how to use computers for writing. Because computers are well established writing tools it could be argued that the whole work is pointless. This is not the case because the recent proliferation of embedded and mobile computers has

led to many situations where traditional text entry systems are ineffective and difficult to use. A persistent skeptic could still argue that although new devices and usage situations have emerged, developing text entry methods for them is relatively simple. Based on the mature knowledge of coding schemes developed in computer science and engineering one should be able to develop optimal systems without much trouble. The counter-argument in this case is the same as in most user interface issues: text entry would indeed be trivial if people were as easily programmed as computers are. Because this is not the case, we need to resort to laborious methods such as experiments to find out how things work when humans are involved.

Branches of science dealing with humans are, of course, not purely experimental. Based on experimental results we can build models and construct theories that can then be used to formulate new questions that can be answered experimentally. A suitably simple and robust theory can also be used to find answers to questions without doing expensive experiments. Some hope that in the future HCI theory can be developed to a level where building user interfaces would indeed be a trivial engineering exercise [Scutliffe, 2000]. We are still far away from that goal. Years of experimental work lay before us.

Although computers offer new possibilities for writing, they do not change everything. The human body is the same as it was 6000 years ago when first writing systems were developed. The motivation for writing is also the same. The need for writing arises when people need to remember things precisely over long periods of time or to communicate over distance [Woolley, 1963]. Deals have to be written down so that all parties can agree on what was agreed upon even after circumstances have changed over time. Records on prices and debts have to be kept in writing when the economy gets complex enough.

Once writing emerges for a reason or another, it tends to spread to other areas of human activity. Serving as a memory for economic activities is just one example. People start writing letters for their loved ones, they write down stories for others to enjoy, and decorate their tombs and other monuments with words that they want to be remembered with.

1.1.1 Language Issues

All writing is not equal. Character sets and writing systems interact with languages in complicated ways. The importance of the language and its effect on the writing systems is exemplified by the chase of Chinese. Writing down the pronunciation of Chinese words in the Latin alphabet simply does not suffice. Chinese has many words that produce the same Latin transliteration, but have different meanings that the Chinese writing system conveys, but the European does not [Sacher, 1998, Wang *et al.*, 2003]. Language issues are important and should be considered in work related to writing. However, a researcher must also recognize his limitations. Verifying that everything in this thesis applies to all languages is clearly beyond my capabilities. Thus, I mostly ignore language issues and confine the discussion within the languages that can conveniently be written with the Latin alphabet. The reader should be aware that generalizing beyond this scope may lead to false conclusions.

1.1.2 History of the Latin Alphabet

Of a particular interest for writers of modern western European languages is the history of the Latin alphabet. The early part, which is the development of the proto semitic script, happened roughly simultaneously with the development of the Chinese writing systems [Gaur, 1987, Woolley, 1963, Grimberg, 1967]. The sites where early semitic texts have been found are close enough to both Mesopotamia and Egypt to make it safe to assume that these older systems were not completely unknown to early developers of the Semitic scripts. Semitic scripts were phonemic, that is the sounds were written instead of ideas or words. They were also consonantal, which means that vowels were not written at all.

The next step after the proto Semitic scripts was the Phoenician trade empire that spread their version of the north semitic script throughout the Mediterranean. Later the Greek added some vowels and adapted the script to their use giving it in turn to the Romans who left the alphabet in the hands of the christian church that was the main practitioner of writing in Europe for much of the middle ages. The use of printing press and the industrialization finally lifted the Latin alphabet to the position that it has today in the western industrialized world.

The name of the Latin alphabet comes from the Latin speaking Roman culture. The monumental script that can be observed in Roman ruins still lives in the capital letters of the Roman family of computer fonts. Sometimes the term Roman alphabet is used instead of Latin and it is not uncommon to see people speaking using modern day associations such calling it the English alphabet.

1.1.3 Terminology

By *text entry* I mean the activity performed to transfer text from the user's brain to computer memory. *Text input* is synonymous with text entry and often used interchangeably. A *text entry method* is the abstract description of how to accomplish text entry. A *text entry system* is a concrete implementation of a text entry method. As is apparent, text entry is a subset of the activities that are usually referred to by the term *writing*.

Text entry does not include the language related issues of syntax, neither are semantics of the text an issue in text entry. *Error correction*, however is a part of text entry by necessity. The way that humans operate always produces errors. This is somewhat analogous to a generic information transmission channel in engineering. There is always noise that must be coped with. The way that human users cope with the noise is first to keep the text entry rate slow enough not to exceed the channel capacity. Secondly, when an error occurs, it is noticed through the feedback channel and corrected unless there is an error in the feedback channel in which case the error goes unnoticed, or in some cases unnecessary correction activity is launched.

1.1.4 Separation of input, storage, and output

The reason for text entry being a more interesting topic than some other writing technology such as the ballpoint pen is that computers are different in so many ways. They can take many shapes and sizes and be operated with different input devices. The short history of computers shows both development of input devices for easier writing with a given text entry method and development of text entry methods for easier writing with a given input device. The peculiar thing about computers is that the physical writing motion is separated from the shape of the resulting characters. Mechanical typewriters and the printing press have similar qualities, but in the case of computers the separation is cleanest. Finger motion in pressing the "H" key on the keyboard is very similar to pressing any other key and very different from the shape of the letter "H". In handwriting the pen motion is exactly the same as the shape of the resulting character and consequently different for all characters. The separation of input activity and character shapes is a powerful feature of computerized writing that could, if utilized properly, solve the problem of having to learn many writing systems that bothered the ancient Egyptian scribes and still burdens us.

On the other hand, the separation means that any physical activity can be translated to text. Computer manufacturers have utilized this opportunity and developed computers with very different input devices. Commonly keyboards such as desktop-sized ones, telephone keypads and mini QWERTY keyboards are used for text input. In addition styli and even speech can be used. These all require different skills from the user. The benefit gained from learning some of these skills is added efficiency. For example it is not uncommon for people to touch-type twice as fast as they can write with a pen.

One of the issues that a thesis in this field should address is how individual users can cope with the multitude of writing systems and input devices. My position is that neither computers nor manual text entry are passing flings. Both are likely to persist until the end of our civilization. Consequently everybody must develop a text entry strategy. Text entry method developers should strive to make this as easy as possible.

1.2 Method

The work reported in the following chapters is done within the paradigms of constructive and empirical research. Because something is being constructed, the work is applied rather than pure science. Constructive research happens in cycle with two phases. One phase is the construction of a system and the other the evaluation of that system. The order and breadth of these phases may vary, but the idea is to develop artefacts with potential practical value and knowledge on these artefacts. In human-computer interaction the artefacts are user interfaces and the targeted knowledge is knowledge on human performance with these interfaces. Most of the work has a heavy empirical emphasis. The reason for this is that I agree with the title of Shumin Zhai's recent outburst on the state of the affairs in human computer interaction. Because, "Evaluation

is the worst form of HCI research except all those other forms that have been tried" [Zhai, 2003], I too have to evaluate my systems in order to learn useful things about them.

Within this overall framework I have used snippets of what other branches of science call the scientific method. These include building thorough descriptions such as taxonomies to understand the problem area, doing evaluations following the experimental research methods largely developed by psychologists, and most importantly use of common sense for example in recognizing situations where an experiment or a prototype cannot solidify knowledge beyond what can be achieved through carefully explained reasoning.

One central methodological issue in applied work is the time perspective that is used to motivate the work. Dealing with this issue is a balancing act between aiming for results with lasting value and aiming for results that are immediately useful. Results that may be found very useful or theoretically interesting in the future are not necessarily immediately useful in practice. On the other hand results that are not immediately useful may indeed be completely useless. Because text entry methods are so tightly interwoven in the culture and technology of a time and geographcal region, any significant change will take a long time. This makes it very difficult to see how the change could happen at all. In retrospect, however, we can observe historical developments that have changed writing systems completely. A recent example of a surprising development is the widespread use of the telephone keypad for text entry. Such changes are likely to continue in the future.

However, placing a particular piece of work in the context of long term developments is challenging. I have attempted this in the case of the notion of device independence addressed in papers I, III, and VII. Faced with that argumentation some people say "maybe" and others say "rubbish". Each according to their present mindset regarding the time perspective. "Maybe" is really the best we can hope for given the general difficulty of predicting the future. In the other end of the scale are the experimental results such as those in papers I, II, III, and IV. They are useful immediately. By producing both immediately applicable results and long reaching theoretical observations, I have hoped to keep the center of mass of the whole body of work in the right place. That is, beyond product development work done in the industry, but with enough ties to reality not to get lost in potentially useless visions.

1.3 Overview of the Thesis

The main content of this thesis consists of seven papers that have been published in various scientific forums. The other parts fill in the gaps between the papers and provide more extensive introductory material than what could be included in the papers themselves. Most importantly Chapter 2 gives an overview of the current state of manual text entry including a new framework of classifying and combining text entry techniques.

In the papers I present three kinds of results. First, the results of evaluations of text input systems, second, models that describe human performance in

certain situations, and third software that solves certain practical problems. The papers are woven together in chapters 3, 4, and 5 that each concentrate on one type of contributions.

The text entry method evaluations in Chapter 3 include four systems. First, a minimal device independent text input method that was an attempt in building a text entry system that can be operated with almost any input device while maximizing skill transfer between the input devices. Second, a comparison of two touchpad based systems for entering numbers. Third, the evaluation of Quikwriting in a multi-device environment, and fourth an evaluation of menu-augmented soft keyboards.

The modeling part in Chapter 4 includes a model for unistroke writing time and work on a combined model of text entry rate development in longitudinal experiments.

The software part (Chapter 5) consists of the components of the Text Input Architecture (TIA). The architecture supports text input methods that follow the user rather than the device.

In Chapter 6 I describe and discuss the general limitations of the work. Finally, conclusions concerning the whole body of work are presented in Chapter 7.

1.4 Division of labor

Because most of the publications were made in cooperation with other researchers, it is necessary to give details on the division of labor in order to satisfy the requirement that the thesis demonstrates capability in independent research. Below I list those parts in the publications that were significantly contributed to by others. Participation in the writing process means discussing the most effective ways of presenting the material that I had generated and editing the paper to realize the chosen presentation.

Paper I is based on my Master's thesis. Professor Roope Raisamo supervised the thesis and the writing of this publication.

Paper II was written on the course for Scientific Writing in Human-Computer Interaction given by Professor Kari-Jouko Räihä. The writing process was influenced by Professor Räihä and some participants of the course. Mika Käki wrote the program for analyzing the results of the experiment and participated in the writing of the paper after the course.

Paper III was written with the participation of Professor Roope Raisamo.

Paper IV was written in two phases. The modeling part was written for a course given by Professor Scott MacKenzie (Research in Advanced User Interfaces: Models, Methods, Measures, University of Tampere, spring 2003). Prof. MacKenzie's comments on that part influenced the final presentation as well as the decision to undertake the experimental part of the work. Some of the ideas were developed based on discussions with Dr. Grigori Evreinov.

Paper V was written on the Advanced Course on Human Computer Interaction given by Professor Kari-Jouko Räihä. The writing process was influenced by Professor Räihä and some participants of the course.

Paper VI was written in cooperation with Professor Scott MacKenzie who suggested model 1 and participated in the writing process.

Paper VII was written with Professor Raisamo.

Chapter 2

Current state of manual text entry

There is only one way to enter text into computers. All text entry methods can be conceptually reduced to a menu of choices and the user selecting these in the desired sequence. Sometimes the process is implicit or obfuscated beyond recognition. For example an artificial neural network that recognizes handwritten words is a complicated intermediary between the user and the list of words. The user does not see the list, nor is she aware of its existence, but the list exists and interpreting the user's selection is the very purpose of the whole system. So, with some faith the selection abstraction can be seen to hold generally.

Description of the current state of manual text entry has no practical value if it consists of a classification of only one class. Therefore it makes sense to differentiate between two types of text entry methods. Those that show the menu explicitly and those where the user is under the illusion that the computer recognizes more freely formatted input. To give the conceptual framework even more practical power, a class for language models is added. Language models are employed in conjunction with either selection or recognition based interaction techniques to improve the system. Recognition systems use language models to improve their accuracy in guessing what the user is writing. Selection based systems typically try to utilize the redundancy in language to reduce the number of selections that need to be made through disambiguation and word completion. These main building blocks of text entry systems are shown in Figure 2.1.

The following description of the known text entry systems aims to be comprehensive regarding the different types of systems. While being comprehensive regarding individual systems would be an even worthier goal, it turns out to be very difficult. For example there are hundreds, if not thousands of publications on handwriting recognizers that appear rather similar to the user, but work internally in different ways. Describing all these systems is an effort that does not serve a purpose in this thesis. Instead, the reader is directed to surveys that specifically address the issue [Tappert *et al.*, 1990, Steinherz *et al.*, 1999, Plamondon and Srihari, 2000, Vinciarelli, 2002].

Another goal of this chapter is to serve as an introduction to some of the

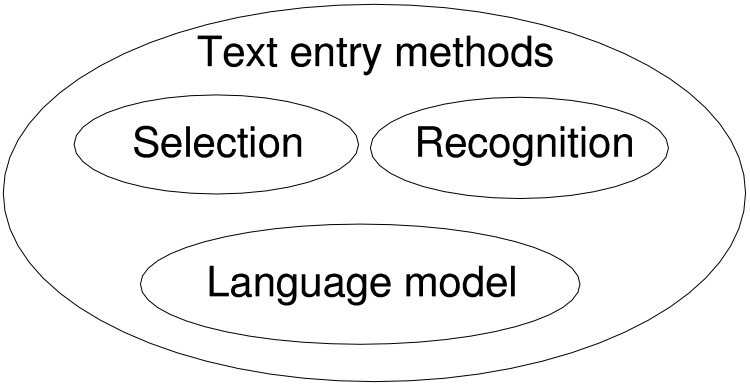

Figure 2.1: Main building blocks of text entry methods.

problems that are addressed later in the book. I list the best practices of handling the various text entry methods theoretically when modeling user performance. This information serves as an introduction to the work reported in the three papers (IV,V, and VI) that deal with user modeling and is given at the end of the discussion on each class of systems. Modeling is an important tool in HCI design and research in general, but particularly so in text entry. Text entry skills are practiced often and for extended periods of time. This is why experts can develop skills that are well beyond what can be measured in an experiment with users that in the case of a new text entry method are necessarily beginners. Thus, it is of great value to be able to model expert user performance as accurately as possible to find those text entry methods that are worth teaching to users.

In addition, this overview discusses two aspects of text entry methods. The first of these aspects is the modularity of composite methods. Composite methods that consist of separable components are good for architectures like the one presented in paper VII because the components need to be implemented only once and then used in many methods. The opposite of modular composites are composites where the parts are so intertwined that clear and reusable interfaces between them are more time consuming to implement than a purpose-built complete re-write of the whole method. The second emphasized aspect is the multi-device compatibility of text entry methods that is a central theme in papers I and III. The reasoning behind multi-device methods is that if the same text entry method can be used on many devices, some learning is saved because the user only needs to adapt to the new device instead of learning a whole text entry system.

2.1 Keyboards

Keyboards are pure selection interfaces. The user is presented with a matrix of keys and he or she is to select them sequentially to produce text. There are two kinds of keyboards: *hardware keyboards* and *soft keyboards*. The terms virtual

keyboards, soft(ware) keyboards, and on-screen keyboards are synonymous in this thesis. I prefer the term soft keyboard because it emphasizes the fact that the keyboard is software rendered in contrast to hardware keyboards that are physical objects. An important difference between physical keyboards and soft keyboards is that they offer different approaches to user interface design. Physical keyboards are largely immutable. The keys are where they are and function the way they were constructed to function. The user interface designer can do very little to change these things. A new keyboard can be designed, but as there usually is no more than one keyboard in each device, the design must be a compromise that serves all applications and users. The shape and size of soft keyboards on the other hand is entirely software controlled as is the visual appearance of the keys. Soft keyboards can change according to the application, user, or even depending on the usage context.

2.1.1 Physical Keyboards

Buttons and keys exist in many shapes, sizes and arrangements. However, a collection of keys is a keyboard worth mentioning in the context of text entry only if it is used for entering text. This rules out light switches and other isolated buttons and switches connected to non-digital devices. However, many household appliances such as alarm clocks, TV-sets, microwave ovens etc. nowadays contain a small computer that every now and then needs textual input. Mostly this happens infrequently such as setting the time on an alarm clock after changing the batteries, but nevertheless the activity concerns a set of keys and a string of text (in this case numbers) that need to be entered. While delving into the intricacies of these user interfaces might be interesting, I will, to stay in par with the other sections, limit the following discussion to keyboards that are used for more extensive text entry tasks such as taking notes or writing an email message.

Desktop Keyboards

The design for desktop keyboards is inherited from the typewriter era. The QWERTY character layout and its language-specific adaptations dominate the market. It has been observed that the QWERTY layout is not optimal for typing the languages that it is used for. Difficult finger movements are needed more often than is necessary. Also, long stretches of text are often written using only one hand. Presumably a layout that relies mostly on the keys on the home row and distributes consequent characters to left and right hands more equally could be better.

One of the somewhat successful attempts at developing a better layout for the English language is the Dvorak layout [Potosnak, 1988]. The reason for the limited usage of Dvorak and other non-QWERTY layouts is the fact that despite its shortcomings, the QWERTY layout actually makes pretty good use of the human hands. While one finger is pressing a key, others can prepare for their work by moving over the following keys. This kind of typing skill takes a while to develop, but once learned, it is fast enough for most practical

purposes. Indeed, attempts to demonstrate the benefits of Dvorak layout have shown only slim success in improving text entry rate [Potosnak, 1988]. Speed is not the only important criterion. Increased user comfort and lessened risk for stress injuries with the Dvorak layout have also been c laimed [Brooks, 2000].

Besides key arrangement, other aspects of the design space have been explored. Laptop computers often have slightly smaller keyboards that are sometimes curved to reduce wrist angles. Many of the currently available desktop keyboards are of the split design. The keyboard is divided at the middle to allow straighter wrist posture. Keys have also been painted on flat surfaces that can sense one or many points of contact allowing simultaneous keying and gesturing [Potosnak, 1988, FingerWorks, 2003].

Desktop keyboards without the actual keyboard have also been constructed. They operate by sensing the finger movements by some other means such as cameras [Roeber *et al.*, 2003] or pressure sensors [Goldstein *et al.*, 1999]. The keyboard can be projected on the desktop [Roeber *et al.*, 2003] or typing can occur without any visual guide [Senseboard, 2003, Goldstein *et al.*, 1999].

Telephone keypad and disambiguation

Because each key in a telephone keypad is associated with several characters, a software layer that transforms the keypress stream into text is needed. Because the software turns ambiguous keypresses to unambiguous characters, the process is known as disambiguation. The paper by Rau and Skiena [Rau and Skiena, 1994] is a good source for information on the state of the art in telephone keypad disambiguation preceding the mobile phone era. What is now considered the traditional disambiguation algorithm associates the first consecutive press on a key with the first character on the key, the second with the second and so on. When two characters on the same key need to be entered consecutively, the user needs to wait for a pre-determined amount of time (usually 1.5 seconds), or press a special timeout-cut key. This algorithm is known as the multi-press disambiguation algorithm.

The multi-press system can be configured in many ways. The standard way is the configuration A in figure 2.2. The alphabetical order presumably facilitates novice performance if novices are familiar with the alphabetical order. The problem with the alphabetical layout is that the frequent characters may end up in the end of the list requiring more keypresses than some less frequently needed characters. Overall this increases the number of needed keypresses and unnecessarily slows down expert text entry rates. A natural reaction is to suggest re-arranging the characters within each key according to their frequency. Pavlovych and Stuerzlinger [2003] have done exactly this and labeled their technique Less-Tap. The Less-Tap character layout is shown under B in Figure 2.2. A more comprehensive re-organization can be done disregarding the alphabetical order alltogether. Layout C in Figure 2.2 is an adaptation of the JustType keyboard (C) as reported by MacKenzie and Soukoreff [1999].[1] Layout D is the result of my own experimentation in the area.

[1]The eight-key layout reported by King *et al.* [1995] is different.

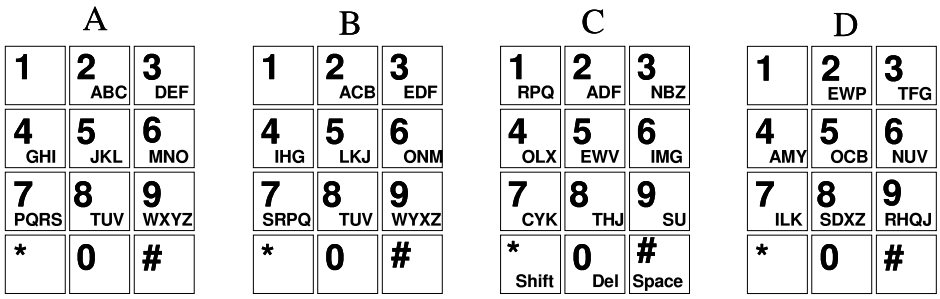

Figure 2.2: Four ways to map the alphabet in a telephone keypad.

The JustType layout was optimized for a specific word level disambiguation algorithm (See below for discussion on disambiguation). Layout D was constructed by starting from the most frequent character in English and assigning a character for each key (except 1) until all keys had three characters in decreasing frequency order. The remaining characters were assigned to the keys with the lowest overall usage frequency. Re-arranging is fairly effective. According to Pavlovych and Stuerzlinger most of the advantage can be gained by within-key re-arrangement. The average number of keypresses per character for writing English with the standard multi-tap arrangement is 2.03. The Less-Tap arrangement manages 1.52. My own computations for layout D indicated 1.47[2]. Although the optimization goal for the JustType keyboard may appear different, it turns out to be pretty much the same. For word level disambiguation characters need to be distributed so that the maximum number of different keys are pressed for each word. The end result is that each key must have roughly the same sum of frequencies of assigned characters. Thus, the number of keypresses needed per character in multi-tap use of layout C would not differ significantly from layouts B and D.

Despite the keypress efficiency of these optimized key arrangements no implementations are widely available. Now that the covers (including key covers) of mobile phones are user changeable, it would not be impossible to have multiple multi-press systems with the correct printing on the key caps so that even novices could quickly pick up the more efficient systems. However, the choice of device manufacturers remains to be the support of visual personalization instead of functional. Given that some of the publications (for example the Less-Tap paper) are relatively new, such devices may be in the development pipeline. Whether that is the case remains to be seen.

The most popular improvement over the multi-press disambiguation is the T9 disambiguation [AOL, 2003]. A T9 user presses each key only once thus saving some keypresses. The T9 algorithm uses a word frequency dictionary to determine the most likely interpretation of the word. If, at the end of a

[2]these computations are done using different English language corpora, so the numbers are not necessarily comparable down to the last decimal place.

word, T9 guesses wrong, the user must press the "next" key to scroll through a list of the less frequent words that match the entered key sequence.

In addition to T9 other algorithms with similar properties have been proposed and used in phones. The simplest of these competitors is the system known as LetterWise [MacKenzie *et al.*, 2001]. It uses a n-gram (a sequence of n characters) frequency table instead of a word frequency table.[3] The user is required to monitor the entered text and press a "next" key if LetterWise guesses wrong.[4] More complicated approaches such as EzType and EzText [Zi Corporation, 2003] iTap [Motorola, 2003] add word prediction to the system allowing text entry with even smaller number of keypresses, but with the added cost of monitoring the system output and reacting to it while writing.

Disambiguation algorithms generalize to all situations where the number of available input actions is smaller than the number of different tokens that need to be entered. The input actions do not need to be keypresses. For example, the octave text input system that was marketed by a French company e-acute used a word-level disambiguation algorithm with an eight-armed star on which one moved a stylus. One arm of the star was selected for each character and when the stylus was lifted, the system computed its best guess for the word.

In addition to language models, explicit user input can be used for disambiguating the keypresses. With the traditional layout one needs four "shift" keys to disambiguate the input. Three shifts suffice if more than one are pressed at the same time [Wigdor and Balakrishnan, 2004]. Another alternative is to install an accelerometer into the device and tilt it while pressing the keys [Partridge *et al.*, 2002, Wigdor and Balakrishnan, 2003].

Other keyboards for mobile use

Keyboards can be seen as a continuum of the number of keys [MacKenzie, 2002b]. At one end the keyboard consists of one key and at the other the number of keys is unlimited. The number of keys is in inverse relationship to the number of keypresses needed for entering one character. Consequently one-key text input is necessarily awkward and time consuming. Useful systems have been constructed for the use of the disabled who cannot conveniently operate more than one button. The approach is usually to use scanning. Scanning means that the possible selections are highlighted sequentially and the user is to press the button when the desired selection is highlighted. Another classic one-button compatible technique is the use of morse code which is based on sequences of carefully timed key presses and pauses.

Two keys can be used in many ways. For example so that one key moves the selection and the other confirms the selection. Starting from three buttons the variety of approaches increases. All the techniques that work with fewer

[3]My impression based on personal communications with eatoni representatives is that trigram frequencies are good enough and actually used in their products. However, the approach is not limited to three character sequences. Therefore, the n-gram expression.

[4]In contrast the output of T9 is often not correct before the word is finished and tends to change as the entry proceeds, it is actually beneficial not to look at the entered text until at the end of the word.

keys are of course available. In addition multiple selection schemes can be envisioned. The design space has been explored at least by MacKenzie [2002c] and Sandnes *et al.* [2003]. Work on techniques for four keys includes the Bin-Scroll [Lehikoinen and Salminen, 2002], four key adaptation of our MDITIM work (Paper I) and other direction based systems. Using five keys adds the ability to select in addition to moving along two orthogonal axes. An example of a movement and selection interface with five keys is explored by Bellman and MacKenzie [1998]. Five keys is also a natural number for chord keyboards [Gopher and Raij, 1988] because it allows allocating one button for each finger. Because of the widespread use of mobile phones for text messaging, the telephone keypad is a major milestone in the continuum between five and 27 keys. 27 is an important number because 27 keys have often been used in simplified models and experiments pertaining to "full" keyboards that have a key for each character. Keyboards with more than 27 keys belong in this sense to the same class that generally tends to aim for one keypress per character operation with minor deviations such as the production of upper case characters. Below we will concentrate on chord keyboards and full keyboards.

Originally mobile phones inherited their keyboard layout from desktop telephones. Only recently mobile phones with keypads other than the 3 by 4 key matrix have become available[5]. Devices that do not have such historical baggage have used other keyboard designs. A popular solution is a very small keyboard with QWERTY layout. Small QWERTY keyboards have appeared many devices including PDAs, two-way pagers and even mobile phones.

Although the QWERTY layout remains the most popular full miniature keyboard design, other designs have been proposed. For example, the Fastap design where alphabetically arranged keyboard is combined with the telephone keypad as shown in Figure 2.3 [Digit Wireless, 2003]. The round telephone keys are not real keys. They are just indentations in the keyboard base plate. The smaller angular alphabet keys are real keys that can be pressed. They are clearly higher than the base plate. When a user tries to press his or her finger into one of the indentations, several of the alphabet keys surrounding the indentation are pressed. The keyboard interpets this as a press of the telephone key. The alphabet keys can be pressed individually. The inventors claim that the key arrangement allows packing more keys per unit of base plate area without making the keys too small to press even with large fingers.

Cockburn and Siresena [2002] tested a Fastap prototype device against multi-tap with a traditional mobile phone keyboard and T9 with another traditional phone model. The experiment consisted of a initial test for determining walk-up usability, six 10-minute practice sessions on different days, and a final test to determine expert[6] performance. Walk up performance with Fastap was found to be superior to both multi-tap and T9. Experts were faster with T9 except when entering abbreviations. Unfortunately the test did not include a QWERTY keyboard with the same physical dimensions as the Fastap pro-

[5]for example Nokia 3650, 5510, 6800, 6910, and 7600.

[6]In comparison to many other studies an hour of practice does not seem like much time to become an expert. The definition of an expert has not solidified in text entry research. In the existing literature it is used for pretty much anything except for absolute novices.

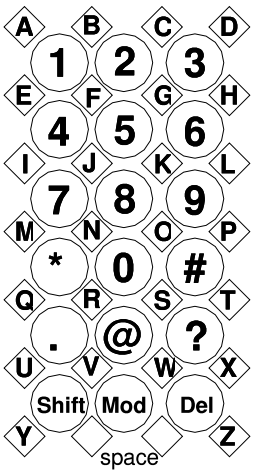

Figure 2.3: The Fastap keyboard design [Digit Wireless, 2003]

.

totype. Including this comparison would have allowed evaluating the claims that Fastap improves the text entry user interface over previous miniature full keyboard designs.

A miniature QWERTY keyboard has many buttons which means that the buttons tend to be rather small. Approaches with fewer and larger keys include the various chording keyboards. Chording means pressing more than one key simultaneously to enter a character. Early work on chord keyboards was done in the context of mail sorting [Noyes, 1983]. Later work has concentrated on text entry. Experiments with chord keyboards have shown that the interfaces tend to be rather easy to learn. In some cases even easier than traditional touch-typing [Gopher and Raij, 1988]. However, even well trained chord typist cannot reach QWERTY touch typing speeds because chording is more sequential whereas touch-typists can prepare for the following strokes in parallel with the execution of the preceding ones. However, chord keyboards can have a very large character set. Chord stenography machines that allow entering more than one character per chord can be operated very rapidly. Also, it should be noted that learning to be a fully trained QWERTY touch-typist takes years of practice. Most people never reach speeds over 100 words per minute. In fact in my experiments typical QWERTY typing rates have been in the order of 40 wpm. At these speeds chording would be competitive if people were to find it otherwise appealing. This does not seem to be the case. The need to memorize the chords seems to deter most potential users. Some chord keyboard manufacturers do manage to survive in this niche market. Currently available chord keyboards include Twiddler2 [Handykey Corporation, 2003],

Bat [Infogrip Inc., 2003] and CyKey [Bellaire Electronics, 2003][7]

Skill transfer from a system known to the users can aid users to learn the use of a new device. The success of mini-QWERTY keyboards and the failure of chord keyboards to enter the market is just one example of this. The Half-QWERTY system is an interesting design that aims to utilize the user's familiarity with the desktop QWERTY keyboard. The Half-QWERTY keyboard is a half of the QWERTY keyboard. The characters of the missing half are located mirrored on the existing half. The space key is used for shifting the active half. Matias et al. tested the design and found that people can transfer some of their two-handed touch-typing skill to half-QWERTY use [Matias *et al.*, 1993, Matias *et al.*, 1996].

Physical keyboard theory

User populations exhibit very large spread of keyboarding skills. Some users can barely type while others are proficient touch-typists reaching speeds up to 100 words per minute (wpm)[8].

Thus modeling the performance of the general user population is necessarily a guesswork. One might assume that it takes on average 500 milliseconds to type one character or that it takes 250 milliseconds and both guesses could be correct. For the same reason detailed psycho-motor models of typing performance cannot be of much value if the user population is not well known. If the user population is known, the best way to estimate user performance is to take a sample of the population and measure the performance. In short, research over the last 20 years has not added much to the performance figures listed by Card *et al.* [1983].

Despite the difficulties, models for typing with full-sized desktop keyboarding can be constructed. Such work has been summarized at least by Barber [1997] and Potosnak [1988]. The models can explain some aspects of the typing activity and produce estimates for the efficiency of different keyboard layouts. While important for understanding the activity, such models have little value in keyboard design. The reason for this is that when both hands and all fingers are used for typing, performance differences for well-trained users are small. Consequently keyboard redesign has been comparatively dormant area of research in recent years.

Because mobile telephones tend to be so small, only a few fingers can be used for entering text using the telephone keypad. Models for expert performance with one finger and two thumbs have been developed [Silfverberg *et al.*, 2000, MacKenzie and Soukoreff, 2002a]. These models are based on the work on soft keyboarding models that are discussed below.

By large, the recent work on physical keyboards has been dominated by the effort of minimizing the number of keypresses in the context of limited

[7] a descendant of the MicroWriter often mentioned in earlier chord keyboard reviews

[8] Words per minute remains the dominant unit for reporting text entry speed despite its shortcomings. Word lengths vary and therefore, instead of words, five character chunks are counted. So, one word per minute is equal to five characters (including spaces, punctuation, and other non alphabet characters) per minute. The more standard and intuitively clear unit of characters per second is emerging, but has not been favored by reviewers until recently.

keyboards. There are at least two reasons for this. Firstly the number of keypresses is a concrete measure that is easy to understand and handle in optimization computations. This makes it very attractive to researchers aiming for academic publication or hoping to attract capital in order to set up a company. Secondly, there has been an opportunity to make real improvements especially in the case of the telephone keypad that has been an important platform due to the explosive growth of SMS messaging that took most device manufacturers by surprise.

The emphasis on keystrokes per character (KSPC) [MacKenzie, 2002b] has left other aspects of text entry activity with much less attention. Different text entry systems can demand different cognitive and perceptual behavior from the user. Sometimes these issues can be even more important than KSPC in judging the suitability of a particular method for a particular use.

One attempt at describing the differences between disambiguation algorithms was made by Kober *et al.* [2001] in an unpublished paper. Their main concern was the effect that errors have in dictionary-based disambiguation. When a word contains one wrong button press, the whole word or a substantial part of it is disambiguated in a wrong way . Kober *et al.* call this phenomenon *error amplification*. Multi-press disambiguation does not suffer from error amplification because errors made on one character do not affect other characters in the word. The main result in the paper is that under certain assumptions the throughput of a dictionary based disambiguation algorithm like T9 will degrade below the level of multi tap when keypresss error rate exceeds 8%. In addition Kober *et al.* modeled their own disambiguation algorithm known as WordWise. WordWise uses a shift key to explicitly disambiguate eight characters thus making 45% of English input unambiguous on a telephone keypad. Because unambiguous characters are encountered often within a word, WordWise is not as sensitive to key press errors as T9.

The work of Kober *et al.* could be expanded to include other disambiguation methods. While error amplification is not as big a problem with many other text entry methods, the cost of correcting an error may vary making modeling the effect of errors on text entry rate a valuable exercise. The work of Kober *et al.* is the only example of this kind of error modeling with disambiguation algorithms, but including errors in performance models in general has been done before. For example Barber [1997] reviews work on using Markov models and task-network models for computing performance of systems like speech recognizers under different error rates. These models could be adapted to describe manual text entry activity just as well.

Other attempts at including the cognitive and perceptual aspects of text entry systems includes the application of the Keystroke Level Model (KSL) by Card *et al.* [1983] to the use of word completion systems. The results of this work are discussed in more detail in section 2.4.2.

2.1.2 Soft Keyboards

Unlike in 1988 when Potosnak [1988] concluded that virtual keyboards would not be covered in the Handbook of Human-Computer interaction due to lack

of research in the area, we now have a wealth of information. Soft keyboards are an attractive way to enter text on touch-screens and stylus operated computers. Reasons for the attractiveness include the simplicity of the needed software, the self-revealing nature of the user interface, and skill transfer from physical keyboards. Experiments have shown that in addition to all these good properties, soft keyboards are very fast and error free in comparison to many other stylus-based text entry methods.

In practice the most popular soft keyboard design is the QWERTY layout and its language-specific adaptations. Practically all pen-operated computing devices are equipped with a QWERTY soft keyboard. In addition they may have other text entry methods, but a soft keyboard is always available as the last resort.

Various alternative layouts have been proposed over the years [Textware Solutons, 2003, MacKenzie and Zhang, 1999, Zhai *et al.*, 2002a] but none of these have gained much popularity. The main reason for this is that although the software-rendered layout is easy to alter, it takes a significant amount of effort to learn to use the new layout. This, together with the relatively small amount of text being entered with soft keyboards, makes users rather conservative in adopting new layouts.

Unlike with physical keyboards, with soft keyboards the key layout has a large effect on the text entry performance. This is because typing is strictly sequential. To type a character one has to move the stylus from a key to the next one and during this time no preparation for the following key can happen. Thus, minimizing the distance to be traveled can greatly enhance text entry speed. This can be done more or less through intuition as in the Fitaly keyboard [Textware Solutons, 2003], and the result can be verified with a detailed model of pointing performance as with the OPTI [MacKenzie and Zhang, 1999] and OPTI II [Zhang, 1998, MacKenzie and Soukoreff, 2002b] layouts. Alternatively a suitable algorithm can be used to do the optimization work using the same efficiency metrics that are used for evaluation [Zhai *et al.*, 2002a].

Soft Keyboard Theory

Soft keyboard modeling is one of the areas of text entry research that have received the largest amount of attention in recent years. There are at least two reasons for this. First, soft keyboards are widely used making research on them well justified. Second, the task lends itself well to modeling because of the limited and predictable role that the user has.

Work on soft keyboards has been reviewed in considerable detail in three papers in the recent special issue of the Human-Computer Interaction journal [MacKenzie and Soukoreff, 2002b, Zhai *et al.*, 2002a, Hughes *et al.*, 2002]. I will not duplicate this effort. Instead, I give only a short overview with some emphasis on issues that are most relevant regarding the work presented later in this thesis.

The basic idea in the dominant soft keyboard models is that because the user is typing with only one finger (or a stylus), the typing activity is actually

a series of discrete pointing tasks. A pointing task can be modeled using Fitts' law [Fitts, 1954, Card *et al.*, 1983, Soukoreff and MacKenzie, 1995]. The models describe the kind of behavior where the motor act of pointing and tapping on the keys is the bottleneck that is limiting the text entry speed. This kind of behavior occurs when people have a lot of experience in the task and no simultaneous cognitive tasks that slow down their performance. In practice this kind of behavior can usually be observed only in bursts between slower passages where the writers thoughts are occupied by something else than the act of typing. However, if the Fitts' law parameters are measured from real usage situation, the model can produce realistic estimates for user performance even when some cognitive deays are present in addition to the motor performance. In this case, however, the modeling assumptions are being stretched. The consequence is that the results are estimates based on the motor performance and an implicit correction for time spent on other activity. Both issues should be considered when comparing such models.

The original model by Soukoreff and MacKenzie [1995] included a component for modeling novice performance with soft keyboards. A person new to a particular soft keyboard needs to scan the keyboard visually and look for the key to press. Soukoreff and MacKenzie used Hick-Hyman law [9] to describe the visual scanning time. Sears *et al.* [2001] have argued that Hick-Hyman law is not suitable for describing visual scanning time because it describes choice reaction time. They also re-defined novices in a more practical manner that does not require them to be completely new to the keyboard layout under question. With this definition it is clear that previous experience is another factor that needs to be included in the model. Unfortunately, no workable model has ensued leaving the modeling of novice soft keyboarding performance a gray area. Luckily novice performance does not need to be modeled because it can be measured. Expert performance, on the other hand, is expensive to measure because training users in the use of a new soft keyboard can take years of time. The Fitts' Law based upper bound component of the model by Soukoreff and MacKenzie remains the best tool for finding an estimate for expert performance. The alternative method by Hughes *et al.* [2002] requires extensive data collection and is therefore somewhat more laborious at least if the quality of the data needs to be good enough to exceed the accuracy of results attainable by the Fitts' law model.

2.2 Menus and Menu Hierarchies

There is no essential difference between a stationary menu and a soft keyboard. Both are selection-based interfaces. However, both are a well known user interface components that are usually conceptualized separately for historical reasons. This is why I discuss keyboards and menus separately.

There are two kinds of menus in user interfaces: stationary and pop-up menus. These are usually managed so that some space on a display is used for

[9] The Hick-Hyman law states that the time from a stimulus to selection of one of N known targets is equal to $c + dlog_2(N)$ where c and d are constants.

a small stationary menu that pops up larger pop-up menus. Context-sensitive pop-up menus that contain options pertaining to the object that was clicked to launch the menu are another commonly used technique. All these approaches can be used in text entry. Menu items can be individual characters, prefixes or suffixes, words or entire phrases.

A large vocabulary can be arranged into a tree form and displayed as hierarchical menu system. A menu system like this can be navigated using a very constrained input device. In the extreme only one switch is needed. Menu items are then highlighted automatically in sequence and selections are done by activating the switch while the desired item is highlighted.

Systems like this are used for text entry especially by people with disabilities that prevent the use of other input devices. The menu systems can be context sensitive so that the tree is pruned of branches that cannot fit the phrase being written.

Hierarchical menus have also been proposed for stylus-based text entry for able-bodied users. The T-Cube system [Venolia and Neiberg, 1994] used a two-level circular menu structure. The first level menu had eight items in a doughnut arrangement around a central ninth item. Landing the stylus on any of these nine items popped up a further eight-item menu. Characters were selected in the second level menu by moving the stylus to the direction of the desired item and lifting it.

The difference between menus and interfaces sometimes labeled "selection-based" or "gesture-based" is not entirely clear. The selection-based techniques such as Cirrin [Mankoff and Abowd, 1998], Quikwriting [Perlin, 1998], EdgeWrite, [Wobbrock *et al.*, 2003] and Weegie [Coleman, 2001] all have an input area that is divided into zones that are selected in specific sequences. Whether we call these sequences selections, menu selections, or gestures does not make that much difference. Herein all these systems are considered menu selection techniques. Systems that claim to be gesture recognizers or character recognizers but work using a similar zone-based algorithm should be considered recognizers. The difference is, as stated above, whether the user is supposed to be aware of the selection nature of the system or not.

2.2.1 Menu Theory

The theory to apply to menu-based text entry interfaces depends on the nature of the interface. If the user does not know the menu system or if the menu system is dynamic and therefore requires the user to observe the display and make decisions, the best way to go is an appropriate adaptation of the Goals, Operators, Methods, and Selection rules (GOMS) [John and Kieras, 1996] methodology taking into account the lessons learned in the use of menus outside text entry [Norman, 1991, Aaltonen *et al.*, 1998, Byrne *et al.*, 1999, Shen *et al.*, 2002, Kurtenbach and Buxton, 1993]. If the system is to be learned so that using it requires only limited cognitive involvement and feedback processing, models of motor performance such as Fitts' law [Fitts, 1954, MacKenzie, 1992] or Steering law [Accot and Zhai, 1997] should be used instead of time constants for the motor parts of the usage. A sim-

ple model for a text entry method involving pointing and menu selection is developed in Paper IV.

2.3 Text Recognition

Initially teaching computers to read the same text representations that are intended for human use may seem like a good idea. From the human perspective it indeed is a good idea. However, from the perspective of computing it is a horrible idea. Text on paper whether it is machine or hand written is not a suitable way to present information for computers. Decades of research has been invested on developing text recognition algorithms and the results are still far from perfect. The capabilities of the currently available systems are impressive for anyone who has ever tried to construct such a system, but for a layman user they are still too error prone. At least if the user is expecting perfection, which is reasonable if the attitude is that computers should not make mistakes. According to studies [Frankish *et al.*, 1995, LaLomia, 1994] users may be expecting perfection, but do not absolutely require it. The required recognition accuracy depends on task and application [Frankish *et al.*, 1995] but 97% accuracy is a good rule of thumb [LaLomia, 1994].

Given the nature of the recognition task, 97% recognition rate is a tall order. The difficulties stem from the fact that when looked at a low level, text on paper is ambiguous. The same shape may mean different things in different places. A circle may be ".", "o", "O", "0", or even the dot on "i", "ä", "ö" or more likely on "å". In handwriting the text is not precisely formatted and different shapes may mean the same thing. People make use of the semantic, and other redundancies in the text to fill in the blanks and resolve the ambiguities. In order for computers to do the same, they would need roughly the same level of language skills that humans have. Despite the ongoing work on language technology and artificial intelligence, this is unlikely to be reality in the foreseeable future.

Regardless of the computational challenges, many text recognition systems are in use. According to the convention in the area, I have divided these methods and systems in two main classes: off-line recognition and on-line recognition. On-line recognition is by far the more important regarding this thesis as it is the desired method in interactive text entry situations.

2.3.1 Off-line

Off-line text recognition means that text is generated first and recognized later. There are several reasons for why this is a good idea. Firstly computing power used to be very limited. When the algorithms could run as long as they needed it was possible to get better results. Another reason for using off-line recognition is that there is more information available because the whole text can be used as a context of recognizing a particular character or word. The last reason for off-line recognition is that it is specifically what is needed. For example scanning and converting texts from paper to computerized form using optical

character recognition is a task that employs off-line recognition naturally. The need for doing this emerges for example when sorting mail or processing checks automatically [Vinciarelli, 2002, Plamondon and Srihari, 2000].

2.3.2 On-line

On-line recognition means recognizing text under some sort of real-time requirement. Usually the requirements are of soft nature such as not keeping the user waiting for too long. A fundamental difference to off-line methods is that the recognition algorithm can use only past events to help its recognition. For example a character recognizer does not know whether a vertical stroke will be followed by another stroke or not. Dealing with this limitation has led to a variety of solutions.

In the context of handwriting recognition on-line recognition usually means having access to data on the dynamic characteristics of the writing. This means that the order of strokes, pen tip velocity, pen tilt, and pen tip pressure can be used to aid recognition.

In addition to the on-line/off-line continuum, text recognizers are different in the use of the context in the recognition. There is a whole range of possibilities from recognizing each character in isolation to recognizing words or phrases with and without a language model for resolving ambiguities. Language models can be simple rules derived out of usage context or more complete systems that include knowledge of grammar and other patterns typical to writing in general or in a specific domain.

Character Recognition

At one end of the range of context use are character recognizers that true to their name recognize text one character at a time. These systems need to deal with the character segmentation problem mentioned above. Solutions include time delays after each stroke in anticipation of another stroke belonging to the same character, boxed recognition where each character must be drawn in its own box, and tentative recognition where the recognizer can take back its earlier guess if new information makes it unlikely to be correct.

Off-line recognition using on-line information

Because character segmentation is difficult especially for cursive handwriting, and recognizing some characters in isolation even after perfect segmentation is sometimes impossible, it makes sense to gather longer passages of input and then recognize words or phrases instead of individual characters. This kind of approach leads to a recognizer that essentially works off-line concerning the real-time requirements, but has access to all of the information produced by a pointing device including timing of the movements. A widely available example of a recognized that utilizes this technique is included in the Microsoft TabletPC platform.

Unistrokes

Ambiguity and segmentation are two significant problems in on-line handwriting recognition. If all characters are drawn with a single stroke and the strokes are designed to be as unambiguous as possible, these problems can be eliminated. The advantage is a greatly simplified recognition algorithm with higher recognition accuracy. The downside is that people cannot use their familiar handwriting, but need to learn a new character set.

Avoiding the segmentation problem is an old trick that could not have gone unnoticed by the developers of the early handwriting recognizers. Similarly it must have been clear that designing a character set to fit a recognition algorithm is easier than designing a recognition algorithm that can recognize traditional handwriting. However, these ideas were not put forward as a goal to strive for until Goldberg and Richardson published their unistroke paper in 1993.

Unistrokes are characters that are drawn with a single stroke. This makes character segmentation trivial because each stylus lift signals the end of a character. The original unistrokes utilized four shapes that were drawn in different directions and orientations to produce all of the English alphabet. Unlike with pen and paper, the direction of stylus movement is a good way to distinguish between characters in on-line handwriting recognition.

Soon after the paper by Goldberg and Richardson, Palm computing [10] began work on their PDA platform that utilized a text input system called Graffiti [3Com, 1997]. Graffiti characters are mostly drawn with a single stroke. The exception being accented characters that are drawn with two strokes so that the base character is drawn first and the accent with the next stroke. This one stroke per character approach resembles Goldberg and Richardson's Unistrokes. The shapes of the characters, however, are usually closer to Latin hand printing than the shapes proposed by Goldberg and Richardson. Although some people find Graffiti cumbersome and dislike it from the bottom of their hearts, it has been a commercial success[11]. Palm PDAs still have a large market share and even the recognizer in the Microsoft TabletPC platform includes a mode for Graffiti-like characters.

Originally Unistrokes were argued to be faster than traditional handwriting. The claim makes sense because the strokes can be simpler thanks to the added dimension of stroke direction. This issue is discussed further in Chapter 4 where a simple model for the relationship of stroke complexity and drawing time is developed.

[10] In keeping with the dynamic years of the IT bubble Palm was soon acquired by US-Robotics that was then bought by 3com. Around this time some of the Palm veterans left the company and set up a competing company called Handspring. A few years later 3Com expunged Palm that then bought Handspring thus completing the circle.

[11] Recently Palm has abandoned their old Graffiti system and bundled a version of JOT by CIC with their PDAs. The new system is called Graffiti 2. One of the reasons for this solution is that Xerox, the owner of the Unistroke patent, won the long running dispute over whether the patent applies to Graffiti. Instead of lisencing Unistroke technology from Xerox, Palm found it better to abandon Graffiti.

2.3.3 Recognition interface theory

Research on handwriting recognition has largely focused on the recognition technology. Work on other aspects of the user interface is more rare. Notable exceptions of this trend are the character set re-design efforts discussed above. Another departure from the mainstream is work on interaction techniques for dealing with situations when recognizers cannot resolve ambiguities without help from the user. This work is summarized and extended by Mankoff *et al.* A typical technique is to present the user with a list of possible interpretations so that he or she can choose the intended one. In other words when recognition fails the user interface falls back to explicit selection. [Mankoff *et al.*, 2000]

Because the design of the characters and handwriting practices in general has been taken as a given and immutable starting point of recognition interface design, there has been little need for models and theories that aid the design of character sets and the recognition interface. One exception is the work on gesture design and gesture design tools by Long et al.[1999, 2000]. Although the original context of the work is gesture recognition rather than character recognition, the findings apply to character recognizers as well.

2.4 Composite Systems

Usually classification efforts run into trouble at some point. One of the troublesome points in previous overviews have been systems that are combinations of two or more basic technologies. Which class does a system with a soft keyboard and handwriting recognizer belong?[Zhai and Kristensson, 2003] Is it a soft keyboard or a handwriting recognizer? My solution is to call it a composite system and place it in its own class. The Composite system class is an umbrella class that covers all combinations of the basic technologies discussed above. In terms of Figure 2.1 this means introducing a category of systems that overlaps two or more of the other categories. Figure 2.4 shows a more detailed version of the category visualization including the major sub-categories described above.

The components of composite methods can be configured in different ways. Parallel and serial configurations are shown in Figure 2.5. In the soft keyboard and handwriting recognizer example above the configuration is parallel. Both components function as independent sources of text. The input is routed to one of them depending on the type of stylus activity that is taking place. The other obvious configuration is serial. In this case the output of one method is the input of another. The chain could in theory be longer than two methods, but real-world examples are difficult to find.

Typically the first method is a text entry system that can be used on its own and the second method in the chain adds some useful functionality. Word completion algorithms, abbreviation engines, and other language models are popular second layer methods.

Word completion aims to guess the word as the user writes it. If it guesses right, the user can accept the completion and move on to the next word. This technique works well if words are long and the word endings do not vary much.

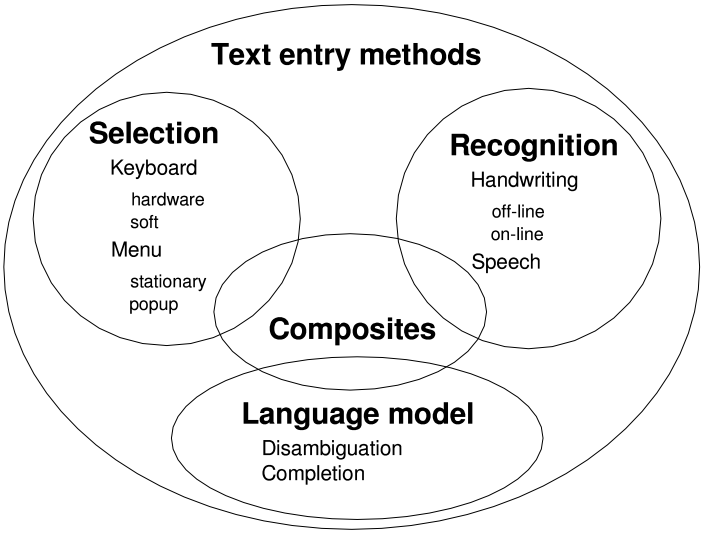

Figure 2.4: Text entry building blocks revisited.

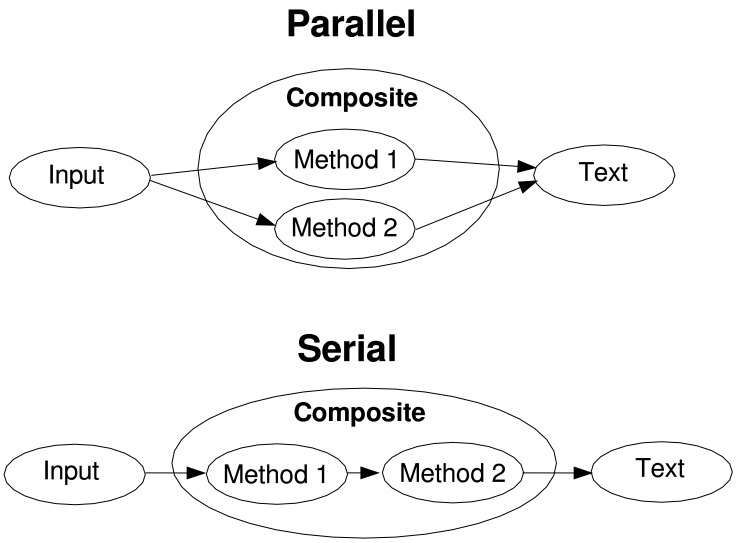

Figure 2.5: The basic composite configurations.

This is the case in some languages but not in all.

Abbreviation expansion engines have an abbreviation dictionary that they use to expand the abbreviations that the user enters. This kind of a system can be useful when a user needs to enter long phrases or words frequently.

Systems with more than one basic language model operates simultaneously are possible. For example, the EzText system by Zi Corporation combines disambiguation and word prediction for mobile phone use [Zi Corporation, 2003].

In addition to the serial-parallel dimension of organizing the components of text entry methods it is useful to think of the level of modularity of composite systems. In a clear parallel configuration the different methods do not need to communicate. Both can produce text as they see fit. In a clear-cut serial configuration the situation is similarly simple. One method produces a stream of text or other tokens and another processes that stream. In these cases the methods can be implemented quite independently of each other. This is the case for most word completion products. They are relatively independent of the underlying text entry scheme. It can be a hardware keyboard, software keyboard or handwriting recognizer. All the word completion package cares about is receiving character events to use for the prediction.

Sometimes such modularity is not equally easy to realize. For example, if we want to configure a soft keyboard to have a pop-up menu that is dynamically updated to contain the most probable characters following the last character entered (in a fashion reminiscent of [Shandbhag *et al.*, 2002] and paper IV), we cannot easily separate the menu and the keyboard into generic modules. The operating logic and the shared display area of the two systems are intertwined in a way that necessitates shared control logic. The control logic can be mediated with systems like Microsoft COM/DCOM that allow control to pass from one process to another, but this does not change the fact that function calls need to be made and somebody has to make them. Therefore, the parts cannot be truly independent. Arguments for the usefulness of independent text entry modules are included in paper VII.

To illustrate the richness of composite systems that can be generated around any given basic text input technology I will take a closer look at soft keyboard composites. Because soft keyboards have been under widespread interest during the recent years, the number of different composite methods with a soft keyboard component is rather large. A nice feature of the soft keyboard composites is that most of them make some kind of sense and could prove useful in some potential situation.

2.4.1 Soft keyboard composites

Disambiguation systems, abbreviation engines, and word and phrase completion systems can be used with any text input system including soft keyboards. However, because soft keyboards with approximately one key for each character are relatively fast, the utility of some of these techniques is questionable. However, there are other ways to use language models with soft keyboards.

Goodman *et al.* [2002] have proposed using a language model to reduce the error rate of soft keyboard text entry. This is a useful approach because

assuming that the user is writing in the language that the model knows, the model will correct errors in the background so that the user does not even notice its existence. However, as we know based on experience with word processors with automatic spelling correction, if the language of the model and the user do not match, the use of the model can actually slow down work and seriously frustrate the user in the process. The basic rules of language specific systems apply. Making one for every language is expensive. On the other hand, a few systems for the major languages go a long way.

Besides adapting the size of the keys based on language model and usage context (this is essentially what the system by Goodman *et al.* does), keyboards can adapt in other ways. These keyboards are composites consisting of the keyboard functionality and the adaptation functionality. At least one such system has been constructed [Himberg *et al.*, 2003]. This system adapted the layout of a soft-keyboard according to the pointing coordinates so that the buttons moved and changed their size to better match the user's typing motions. The keyboard that Himberg et al. experimented with was the traditional nine-key numeric keypad. It was used on a flat touch-screen with the thumb so that the other fingers were behind the screen. In the experiment the keyboard performed in a stable manner and the adaptation seemed to make sense in terms of the movement capabilities of the thumb. However, sometimes the system produced fast and large changes in the keyboard layout leading to key placement that was clearly undesirable. The adaptation algorithm needs to be improved. It is unclear if this kind of system would be useful in general.

Soft keyboards and menus have been combined in many ways. The two main goals have been to save space and increase text entry speed. Space can be saved by placing infrequently needed characters in a menu that pops up in convenient places. Shanbhag *et al.* [2002] constructed a soft keyboard and menu composite for entering the Devangari script. In this approach the 50 Devangari script primitives are arranged to groups that are accessed by selecting one of 21 keys showing "group leader" primitives. The initial selection changes the key assignments so that the surrounding keys contain the other characters in the group. Thus, characters are entered with taps and menu selections. Similar approach has been used in some soft keyboards for languages that use supersets of the Latin alphabet. For example the Fitaly soft keyboard [Textware Solutons, 2003] includes a "sliding" feature that allows entering an upper case version or an accented version of a character by doing a menu selection after landing on a key. The feature can also be customized to fit user preferences. Use of this kind of technique for speeding up text entry is examined in paper IV.

The shorthand aided rapid keyboarding (SHARK) system is an interesting composite system. It combines two kinds of language modeling with a recognition and selection based text entry methods. The soft keyboard component is the ATOMIC keyboard [Zhai *et al.*, 2002a]. The key positions have been optimized to minimize key distances when entering English text. The soft keyboard can be used in the normal manner by tapping the keys. Additionally, when the user draws on the keyboard with the stylus, the trajectory is recognized using a handwriting recognizer. The recognizer knows the shapes

that connect the keys of the most frequent words in the language (the second application of language modeling). So, the user can lift the stylus between each key or drag it from a key to the other and both behaviors result in the entry of the same word. Additionally, the recognizer does not mind if the size of the stroke changes. It is still recognized correctly. The shape of the stroke can change within limits giving the user some freedom to cut corners in order to achieve faster strokes. The goal is to let the user use the recognition part for rapid entry of the frequent words and the tapping part for sequences that he or she does not know well enough to draw. Zhai and Kristensson conducted an experiment and showed that the trajectories can be taught to both the handwriting recognizer and the users. Final conclusions on the usefulness of the system are yet to be made because long term trials with the system have not been conducted to measure the user and recognizer performance. [Zhai and Kristensson, 2003]

While methods for text enty in non-european languges in general are outside the scope of this thesis, I will mention one system as an example of more complicated composite systems. The Predictive cOmposition Based On eXample (POBox) system [Masui, 1998a, Masui, 1998b] is mainly intended for input of east asian languages such as Japanese and Chinese that have very large number of characters. It can also be used for European languages, but the advantages of using it are more limited. POBox contains a soft keyboard, a handwriting recognizer, an abbreviation expansion engine, a word completion system, a stationary (but dynamically updated) menu, and a popup menu. For a detailed description of the system, we refer the reader to Masui's articles [1998a, 1998b, 1999] on it. Here it suffices to say that the components have both parallel and serial relationships. Because of the large character set of the Japanese language POBox is an efficient way to enter Japanese into pen-based computers despite the congnitive and perceptual demands it places upon the user. Consequently, it is more widely used than the other systems discussed in this section. Many implementations are available for download in the Internet. Additionaly adaptations of POBox have been used by Sony in mobile phones in the Japanese market[12].

2.4.2 Composite system theory

The notion of composite systems emerges from the classification effort. It is an useful notion for understanding the structure of text entry systems and identifying the proper context of the different features of the components, but it is not especially useful in modeling user performance. Additionally, most composite systems are marginal in the real world. Thus it is not surprising that no general theory or tools for modeling user behavior with composite methods exists. The way that user modeling is done in these cases is to use ad-hoc composite models that combine the models of the component methods. While I am not aware of any examples, it would be relatively easy to combine for example the Keystroke Level model (KSL) [Card *et al.*, 1983] as employed

[12]According to personal communications with Toshiyuki Masui and on press releases by Sony and Sony Ericsson.

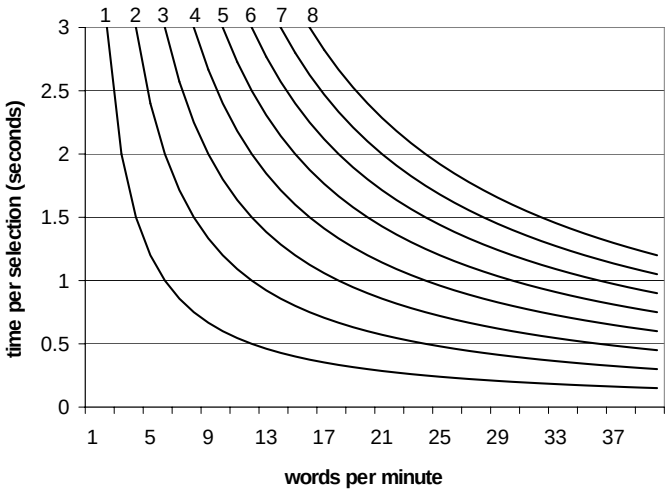

Figure 2.6: Limits for the usefulness of word completion with different number of characters saved per completion. In the area below each line some time can be saved by using word completion. The lowest curve is for one saved character and the highest for eight saved characters per completion.

by Dunlop and Grossan [2000] and the Fitts' digram model by Soukoreff and MacKenzie [1995] into a new model that could be used for modeling soft keyboard composites that do require significant cognitive effort.

While it may be difficult to construct accurate models for user performance with complex composite systems, it is sometimes easy to find limits within which the user performance must be.

For example we can estimate whether word completion will be helpful if the speed of the underlying text entry method and the time needed for selecting or accepting the completion are known. Figure 2.6 shows an adaptation of Figure 1 in by Zagler [2002] [13]. The curves show the borders where use of word completion starts to pay off. Below and to the left of any given curve word completion can save some time. Above and to the right using word completion is slower than not using it. The factors accounted for in Figure 2.6 are the speed of the text entry technique being used (horizontal axis), the time needed for each word completion (vertical axis), and the number of characters entered through a completion (curves from 1 to 8).

If we have a text entry system that can produce about 40 words per minute and selecting a word completion takes on average one second, we can see that each completion has to save us from entering seven or more characters in order to speed up text entry. Such a system is very difficult to construct because the average word length in English is less than seven characters. On the other hand if selecting a completion still takes the same amount of time but we are using

[13] That may have been inspired by Koester and Levine [1994].

a very slow text entry system such as a gaze-operated keyboard (10 wpm), we can see that if the system saves more than two characters per completion, it can be helpful.

2.5 Multi-device methods

Some text entry methods are designed for use with a specific device. This makes sense because many devices have unique capabilities that can be exploited to make the method faster and more pleasant to use. However, having a different method for each device creates the need to learn many methods. The idea of multi-device text input methods is to design methods that can be used with as many input devices as possible. This idea is in use for example when the QWERTY layout is used on a soft keyboard. The designer of the soft keyboard has decided to utilize skill transfer from physical desktop keyboards instead of requiring the user to learn a new keyboard layout.

Designing a good multi-device text input system is a difficult task because maximal device independence tends to produce systems that use only those features that all of the compatible devices share. This set of features tends to be very small. Some devices are not used optimally making it difficult to match the performance of device specific methods.

Dasher by Ward *et al.* [2000] is an example of a multi-device method. It can be used with any input device that allows reasonably good two-dimensional pointing. This includes mice, styli, joysticks and eye trackers.

Dasher is used by pointing characters that appear from the right edge of the display. Each character has its own area within which the pointer has to be in order for the character to be selected. The character areas close to the pointer grow in size until they fill the whole display. the following characters then grow within the area of the preceding ones. The growing speed of a character area is controlled with the pointer. The closer the pointer is to the right edge of the display, the faster it grows. This dynamic animation is orchestrated by a variation of a data compression algorithm so that the most probable following characters are given the largest initial sizes. Entering "typical" text can be done very fast because the typical strings are present most prominently and therefore they are easy to see and select.

In empirical tests by Ward [2001] Dasher proved to be competitive in both speed and error rate against traditional stylus based text entry techniques such as handwriting and soft keyboard tapping, but not as fast as touch typing on a desktop keyboard. In eye tracker use Ward claims the highest text entry rates ever recorded on an eye tracker (up to 20 wpm).

The disadvantages of Dasher include the relatively large display area that it requires and potentially stressful operation as the user needs to control the cursor continuously. Taking a break requires a conscious decision to withdraw the cursor on the central area so that the animation stops.

Multi-device text entry methods are discussed in more detail in Chapter 3 where two experiments on such systems are reported.

2.5.1 Multi-device Theory

Like with hybrid systems, the use of multi-device systems is problematic to model accurately. The reasons for the difficulty are somewhat different. Hybrid systems themselves can be complicated and therefore their interactions with the user very varied necessitating complicated models. The interactions with multi-device systems tend to be simple because they utilize only a limited set of input primitives that are common to all compatible input devices. The existence of multiple input devices is the factor that complicates the situation. Because the input devices can be different, one model most likely cannot handle them all accurately in a generic model. Device-specific modeling can be more fruitful. The appropriate methodology depends on the implementation on the particular device. Suitable models can be found in the earlier sections.

2.6 Performance of the different methods

Comparing the performance of different text entry methods is almost impossible to do accurately. Because of the long learning path of many methods, users exhibit a wide variety of skills. Even empirical pairwise within-subject comparisons are influenced by the earlier experience that the users have. Despite these difficulties there are good reasons for doing performance comparisons. Improved performance is by large the only objective reason for choosing one text entry method over another.

Using information throughput measures commonly used in engineering for modeling human performance has been a long running undercurrent in HCI. Examples of such work include some uses of Fitts' law that are based on the analogy between Fitts' equation and Shannon's theorem for information transfer over a noisy channel [Ward, 2001]. Shannon's concept of information is sometimes directly applicable to user interfaces. For example in a communications system for disabled people the user is often actually selecting one object among many. This is the very task that Shannon used for his definition of information [Shannon and Weaver, 1949]. Consequently in this area there have been calls for using bits per second as the measure of user interface efficiency [Wolpaw *et al.*, 2002].

There is nothing wrong with these endeavors. Shannon's theory is sound and consequently does describe information transfer between a computer and a user. However, just like in engineering, the theory only sets the limits under which the systems operate. Exact information transmission rates depend, within these limits, on the practical implementations of the apparatus that are used for the information transmission. Conclusions such as those drawn by Ward [2001] should, therefore, be taken with the caution that while this or that transmission rate is theoretically possible, it may not be so in practice.

The performance figures given below are based on experimental results available in the literature. In some cases modeling results have been used to fill in blanks in the experimental work. Overall the numbers have been selected to reflect the best available knowledge and to give a coherent view of the state of the art without going into the intricacies of each system, experiment and

model. Consequently the given numbers are unlikely to be strictly accurate. The purpose is to give an overview, not to replace more detailed comparisons.

Keyboards

Full keyboards are by far the fastest text entry methods in common use. World records of over 200 wpm over short periods of time have been set. Highly proficient typists can maintain speeds of over 100 wpm for several minutes. Typical non-professional typists reach speeds between 40 and 80 wpm.

Often comparison between a desktop keyboard and a text entry method intended for mobile use is not really fair because some mobile devices are used with just one hand. I have been unable to find reports on one-handed typing. Therefore I, in the context of work reported in paper III, measured one-handed text entry rates. The results indicate a rate of about 20-25 wpm with a desktop QWERTY keyboard. This corresponds to about 70% of the two-handed performance of the same participants who were not particularly fast averaging only 36 wpm. With faster typists the difference may be created even if, unlike my participants, they take some time to train their one-handed skill. Whether to compare the performance of other text entry methods to one or two-handed typing does not depend only on the one- or two-handedness of the method being compared, but also on whether two handed keyboarding is a realistic alternative. In mobile use this is rarely the case.

Because miniature QWERTY keyboards are too small to fully allow the parallelism that makes desktop-sized keyboards so fast, they are somewhat slower. The results of the Dom Perignon speed contest organized by Textware Solutions [2002] give an indication of the kind of performance that is possible with highly trained users and limited text passages. The highest rate measured in the third contest was 84 wpm. Due to extreme training with the short text passage used in the context, this result exceeds the upper limit estimate of 60.74 wpm produced by the model for two-thumb text entry [MacKenzie and Soukoreff, 2002a]. Typical expert text entry rates with full miniature keyboards are likely to be in the order of 20-40 wpm.

Text entry rates with the telephone keypad have been measured in experiments and estimated with models. Novice performance with multi-tap disambiguation is typically around 7 wpm. The longest experiment with multi-tap was in the LetterWise study by MacKenzie *et al.*. By the 20th 25-minute session the participants reached the average rate of 15.5 wpm. Models predict that the human motor system allows speeds up to 27 wpm [MacKenzie *et al.*, 2001]. Disambiguating language models reduce the number of necessary key presses. MacKenzie *et al.* measured average text entry rate of 21 wpm with the LetterWise algorithm. Theoretically 38 wpm should be possible [MacKenzie *et al.*, 2001].

Chord keyboards seem to be relatively fast. Speeds up to 36 wpm with one-handed chording and up to 42 wpm with two-handed chording have been reported after 35 hours of training [Gopher and Raij, 1988]. These rates would undoubtedly increase with further training. Due to the scarcity of chord keyboard users information on highly trained users is not available. However, we

can safely assume that chording cannot be as fast as touch typing on regular keyboards. This is because chording is more serial than ten-finger typing. The whole hand is committed to the entry of one character and no preparation for the following ones can happen. Two-handed keyboards allow parallel operation of two input streams, but this is still far from what can be achieved with ten somewhat independent fingers. A reasonable estimate for the range of expert text entry rates possible with chord keyboards is in the order of 40-70 wpm.

The crucial difference between physical keyboards and soft keyboards is that soft keyboards usually allow only one point of contact. This makes the motor activity in text entry strictly serial. Consequently soft keyboards are not quite as fast as physical miniature keyboards. Dom Perignon III Speed Contest recorded the highest soft keyboard rate of 78 wpm. This rate was recorded with the the Fitaly keyboard that has been modeled to be capable of about 42 wpm [MacKenzie, 2002a]. Again, the modeling result attempts to reflect the average performance of a well trained population of normally talented users, whereas the record rate has been set by an apparently exceptional individual. In an experiment with the OPTI layout that has modeled performance roughly equal to the Fitaly layout, the participants achieved an average text entry rate of 45 wpm. Overall, text entry rates with soft keyboards are in the order of 15-50 wpm depending on the key organization and user skill level.

Menus and Menu hierarchies

Scanning menu systems intended for communications aids for the disabled tend to be slow. Text entry rates are at best in the order of 10 wpm. With more expressive input devices scanning can be replaced with direct selection. yielding higher rates. The next obstacle is overcoming the need to use the visual feedback loop for guiding the selection. This can happen if the user's learn the menu layout so that they do not need to see and comprehend it in order to use it. This can happen for example in T-Cube where the second level menus can be learned. Only the initial selection in the first menu needs to be visually guided. The second selection can happen instantly after it in one fluid motion. A longitudinal pilot experiment with T-Cube yielded text entry rates between 12 and 21 wpm [Venolia and Neiberg, 1994]. At the end text entry rates were still growing suggesting that with practice the rate would improve. It is likely that efficient menu systems yield text entry rates only slightly lower than soft keyboards. That is, experienced users can enter text at rates between 20 and 40 wpm.

Handwriting Recognition

Text entry rate with handwriting recognition is slightly lower than traditional handwriting speed. The fastest shorthand systems are mostly unsuitable for text entry because they rely heavily on abbreviations effectively increasing the number of strokes to be recognized enough to make constructing a reliable recognizer nearly impossible. Even regular handwriting tends to deteriorate as speed increases. Realistically we can expect fluent recognition to happen

with text that is written somewhat slowly and taking into account some of the special needs of the particular recognizer being used. Alternatively one can write fast and spend time on correcting errors. The end result is that effective text entry rate is less than 25 wpm [Ward, 2001, Chang and MacKenzie, 1994, MacKenzie *et al.*, 1994].

Composite methods

As explained above in the context of word completion systems, composite methods that employ a language model to reduce the number of input actions in selection based text entry methods can increase text entry speed. Due to the need of visual feedback and cognitive effort related to processing the suggestions made by the language model these techniques are effective only if the underlying text entry method is slow enough. Other composite methods such as the SHARK [Zhai and Kristensson, 2003] system and my work in paper IV claim to offer speed advantages, but have so far not demonstrated significant improvements. Consequently composite systems tend to perform no faster than the fastest one of the component methods.

Multi Device methods

The known multi-device methods achieve their input device compatibility through using some form of two dimensional pointing that degrades gracefully when the performance of the pointing device diminishes. For example MDITIM uses a touchpad or a mouse but extracts only four movement directions and a button press from the input. These can just as well be entered with five keys. Similarly the nine tokens used for Quikwriting input can be entered by pointing or with nine keys. Due to its extreme simplification and unfamiliar character shapes MDITIM is slow: only 7.5 wpm after five hours of practice. Quikwriting and Dasher on the other hand are somewhat competitive in comparison to other systems that can be used with the same input devices. With eye trackers Dasher is the fastest known system allowing expert text entry rates of over 25 wpm [Ward, 2001]. Highest joystick-based text entry rate of 13 wpm is reported for Quikwriting (Paper III). Although there are no empirical results available, Dasher is likely to be faster in joystick use. Overall multi-device methods are likely to be slower than the fastest device-specific methods with each device.

Chapter 3

Experiments

This and the following two chapters introduce and discuss the papers that contain the main contributions of this thesis. For each subject matter in the papers there are two subsections: introduction and discussion. It makes sense to read the introduction before the relevant paper and the discussion after the paper.

The breadth and depth of the treatment in these chapters varies depending on the amount of relevant work that has been left out of the papers due to space constraints involved in conference publication. In some cases new material is introduced based on feedback received after the publication.

The work is divided between this chapter and Chapter 4 on models based on the content. Some papers contain both experiments and models. Here the experiment is the main focus. Discussion on the central modeling part of Paper IV is left for Chapter 4.

3.1 MDITIM

3.1.1 Introduction

One of the main themes in this thesis is coping with the variety of input devices that are available. Paper I presents the idea of designing text input methods that can be used with many devices in the context of a text entry method designed for the purpose. Paper II examines the suitability of clock metaphor previously applied with pen input for touchpad-based entry of numbers. Paper III extends the use of another text entry method originally designed for pen-based computers to joysticks and keyboards. Finally, Paper VII discusses another approach to the issue. Namely, an architecture that automatically selects suitable text entry system based on user preferences and the available input devices.

3.1.2 Discussion

Statistical tests are missing from paper 1. I re-analyzed the data from the experiment and present the results here to patch paper I to the form that has been followed in the later experimental papers.

Two issues should have been tested. First, we claimed that participants can learn to use the text entry system. A repeated Measures ANOVA confirmed what is obvious based on Figures 4 and 5. The session (i.e. practice) has a significant effect on the text entry rate ($F_{9,36} = 22.698, p < 0.000000001$).

The second issue was the existence of skill transfer from touchpad to the other devices. This claim seems valid based on Figure 7, but its statistical justification is more difficult based on the collected data. The weakness is that we did not do the same measurements with all devices. The missing piece of information is user performance with other devices than the touchpad before the 5-hour touchpad training. We can assume that the performance would not have been any better than with the touchpad, but we do not really know based on the collected data. Additionally, in the Discussion section we claim to have found differences in text entry rate with the different devices.

The best we could do to examine these issues was to run 15 paired samples t-tests to compare the six productive text entry rates (first touchpad session, last touchpad session, trackball, mouse, joystick, and keyboard). The only significant (at $p < 0.05$ level) differences were between the first touchpad session and everything else. Thus, we have no statistical grounds for claiming that the devices exhibited different text entry rates in the end test. However, such differences are likely to exist. The power of our test was low due to the small number of participants. More powerful experiment would be likely to reveal statistically significant differences.

The data does seem to support the belief that there was significant skill transfer. The first touchpad session was slower than all other tested conditions. The differences were highly significant ($p < 0.005$). Differences between the first touchpad session and the final text entry rate with touchpad and joystick were significant enough to withstand the bonferroni adjustment for 5% family-wise type I error probability for 15 comparisons ($\alpha = 0.003$). In other words it seems that the participants were faster with all devices after the practice with the touchpad than they were with the touchpad without practice. This suggests skill transfer, but does not show its existence because we did not measure user performance with the other devices before the practice with the touchpad. A sensible interpretation is that most of the learning that occurred during the experiment was in learning the character shapes. This knowledge can be transferred between devices. Thus, skill transfer is likely to have happened.

All in all, our original conclusions seem correct, but not entirely justified.

3.2 Touchpad-based Number Entry

3.2.1 Introduction

Having recently finished work on MDITIM, I was listening a presentation by Professor MacKenzie on the work that he and his colleagues had done on the PiePad system [McQueen *et al.*, 1994, McQueen *et al.*, 1995]. PiePad used the clock metaphor for easy remembering of the menu locations of numbers. The

main problem with it was that the error rate tended to be high. This is understandable because the menu slices were only 30 degrees wide. The two-segment characters in MDITIM were easy to draw and recognized robustly. These two pieces of information were combined in what is referred to as the hybrid design in paper II.

3.2.2 Discussion

The publication of paper II was met with two kinds of commentary. First, it was observed that the results depend on human capabilities and the capabilities of the algorithms used for recognizing the strokes.[1] We do not claim otherwise. Based on our data we cannot conclude that the better performance is due only to the fact that the hybrid strokes are better for the user. They are also better for the recognizer. The other part of this argument is that the pure stroke recognizer could possibly be improved so that there would be no difference between the two systems. This is possible, but that does not mean that it was futile to test the un-improved pure recognizer. Now that we know that it performs poorly we have the motivation for attempting the improvements.

The second class of feedback consisted of suggestions for improving the user interface. This includes ideas like printing or engraving tactile guides on the touchpad and using some adaptive[2] or intelligent recognition algorithms. These are all good suggestions, but similarly to the first point we consider them ideas for further work rather than shortcomings of paper II.

3.3 Quikwriting on Multiple Devices

3.3.1 Introduction

The motivation for undertaking an evaluation of Quikwriting [Perlin, 1998] arose from a number of sources. Firstly, the literary record on the subject needed to be corrected. The statement in the original publication on Quikwriting being typically three times faster than Graffiti [3] has been frowned upon over the years. For example MacKenzie uses it as an example of inflated claims that are not based on quantitative measurements and should therefore not be made [MacKenzie and Soukoreff, 2002b]. MacKenzie has good grounds for stating that Perlin's claim is not based on properly gathered quantitative evidence. However, to credibly refute the claim one needs to measure the performance of Quikwriting. Inflated claims are commonplace enough not to justify arduous experimental work on their own. Our main motivation for experimenting with Quikwriting was that it is well suited for adaptations for different input devices. It, like MDITIM works on all two-dimensional pointing

[1] This view was sharply presented by Guo Jin from Motorola Silicon Valley Human Interface Lab.

[2] Re-analysis of the collected data from the point of view of designing an adaptive recognizer was suggested by Barton A. Smith from IBM Almaden Research Center in a posting at CHIPlace (www.chiplace.org).

[3] See last paragraph on page 2 in [Perlin, 1998].

devices and keyboards with four or more keys. Because of this, Quikwriting was a good tool for testing some of the issues in the text entry architecture (described in Paper VII) that I was developing at that time.

3.3.2 Discussion

We did not compare Quikwriting and Graffiti head-to-head. Therefore, strictly speaking, Perlin's claim still remains to be refuted. However, unlike before, we now have a measured learning curve for the early part of Quikwriting use. More importantly we found Quikwriting well suited for multi-device use and it appeared to perform better than MDITIM.

3.4 Menu-augmented soft keyboards

3.4.1 Introduction

Paper IV is different from the earlier text entry experiment papers because it does not address the issue of multi-device compatibility. The connection to the main subject matter of this thesis is through the modeling section discussed in the next chapter. The model shows that combining a soft keyboard and a marking menu makes text entry significantly faster on some soft keyboard layouts. The experiments were done to clarify the conditions under which this might happen.

3.4.2 Discussion

The results presented in paper IV have been met with many kinds of critique and questions. What was the purpose of the first experiment? What would have happened if the longitudinal experiment had continued? Would it not be easier to learn a optimized soft keyboard layout? Can using the menu ever be worth learning? What is the nature of the cognitive burden measured in the second experiment? Most of these questions concern issues that I do not have data on. This makes giving conclusive answers impossible. However, some aspects of these issues can be discussed in more detail than the space constraints in paper IV allowed.

The purpose of the first experiment was simply to see if tapping and selecting indeed is as efficient as it seems intuitively. For those readers who trust their intuition this may seem an unnecessary step. I considered it worth taking to make sure that the basic notions in the modeling of the motor efficiency and the whole concept are not fatally flawed.

The longitudinal experiment was preceded by a pilot experiment that took several weeks. I used the same 15+15 -minute protocol that was used in experiment 2 up to session 92. The crossover in the text entry rates happened around session 50. Another person did the same up to session 27. He reached the menuless text entry rate but did not show speed advantage with the menu-augmented system. We also did short experiments with different learning protocols that introduced the menu items gradually instead of suggesting to

learn them all at once. To no avail, we observed no benefits under this protocol. Based on these experiences it was clear that we could not demonstrate speed advantage with the menu-augmented system in a 20-session experiment.

However, it was equally clear that the performance of the pilot participants was potentially tainted by intimate knowledge of the workings of the system and possible motivation to show that the menu is a valuable idea. So, an experiment with more independent participants was needed to see if these initial experiences were accurate in the sense that the menu usage can be learned and that the text entry rate does indeed increase as rapidly as it seemed to do. The results seemed to confirm our initial observations. Unfortunately the participants did slightly better than we expected almost reaching the menuless text entry rate by session 20. This makes it seem as if the experiment ended on a very critical moment. However, producing a statistically significant difference in favor of the menu-augmented system would have taken at least until session 30. Running the experiment this long was impossible due to practical scheduling reasons.

It does not seem reasonable to assume that the development of the text entry rate with the menu-augmented system would suddenly stop at the menuless rate. Other experiments have not shown evidence of some common general barrier for text entry rate with different systems even when used with the same input devices [MacKenzie and Zhang, 1999, McQueen *et al.*, 1995, MacKenzie *et al.*, 2001]. Therefore it is reasonable to believe that at least in short term, the power curves are accurate estimates of future performance.

A different question is whether the speed advantage that expert users might have is significant in practice. Using the menu seems to be cognitively more demanding that using a plain soft keyboard. Even if the cognitive performance can be trained to a level where the motor performance begins to limit text entry rate (as suggested by the model), it could still be demanding enough to hurt the user's multi-tasking capability while entering text. If this is the case, using the menu might not be wise even if it was faster.

The critical advantage of the menu-augmentation is that traditional and menu-augmented usage of a soft keyboard can coexist. Traditional use of a soft keyboard is not disturbed. However, in the context of soft keyboards this advantage is especially slim. Soft keyboard layout can be changed easily depending on user preferences. Thus, it might indeed make more sense to learn a new optimized sot keyboard layout. The participants in a study that compared QWERTY and an optimized soft keyboard layout reached their QWERTY performance in about 200 minutes [MacKenzie and Zhang, 1999]. With the menu-augmented system in Paper IV it took about 300 minutes. Due to differences in the experimental procedure these figures may not be directly comparable. However, they suggest that learning a soft keyboard layout is easier than learning the on-line planning skill that is needed for efficient utilization of the vowel menu.

One aspect of the user interface that was not tested or discussed in Paper IV, is the physical strain while using the systems. Rapid text entry with the menu-augmented system seems much more peaceful and relaxed than entering the same text at the same rate without the menu. This is because 30% of

the characters seem to appear for free. The input activity in these cases is piggybacked on the tap on the previous key. Through reducing the need to move the stylus the menu use also reduces the hand movements. It could be that this reduces the stress on the hand potentially reducing the risk of stress injuries. Without objective data on the actual strain on the hand this conjecture is, of course, unfounded. However, it is a factor potentially worth investigating in future work.

3.5 Future Work

Detailed ideas for further work with each individual system can be found in the papers. On a more general level the experimental work presented above has revolved around the notion of device independent text input methods. Despite the effort, I have failed to find a system that would be compatible with a wide range of input devices and competitive in speed and error rate with the best systems for each device. In the future the notion of device independent text entry methods should be kept in mind and if suitable candidates emerge, they should be investigated. When a good device independent text entry method is paired with the kind of system described in chapter 5, the concept may suddenly have practical value. At this point, however, device independent text entry is unrealizable due to lack of suitable text entry methods and architectural support.

Chapter 4

Models

Experimental work produces isolated pieces of information that sometimes suggest the existence of general rules that govern the phenomena under investigation. Models condense this information into useful constructs that can be used to describe and predict events in similar situations. In short, my approach to modeling is utilitarian in the spirit described by MacKenzie [2003]. The simpler the models are the better as long as they are useful.

Below I describe models that address three issues in text entry. First, models for learning, second, a model for unistroke writing time, and finally a model for text entry rate with a menu-augmented soft-keyboards.

4.1 Models for Text Entry Rate Development

4.1.1 Introduction

Text entry involves extensive learning. A short-term test, say five minutes of writing, does not tell much about the text entry system. What it tells about is how a particular user (or a group of users) performs with a text entry system given the learning preceding the test. If this is all we want to know a short test is OK. If, however, we want to know what would happen if the tested text entry system were to be used for extended periods of time, we need to account for learning. Historically commitments to text entry systems tend to be rather long. This is why we need to understand the effects of learning on the user performance with any system proposed for general use.

Learning has very different effect on error rate and text entry rate.[1] Error rate is a product of the speed-accuracy trade-off that the users make. Typically in a longitudinal experiment with a new text entry system error rate is initially high but quickly falls to a level that the users are willing to tolerate. If the error tolerance of the users does not change, error rate tends to stay on this same level until the end of the experiment. Text entry rate on the other hand improves following the power law of learning. This law can be used to describe the time needed for an individual action such as entering one character, word

[1]This is by no means an original observation. McQueen et al. [1994] give Bailey [1989] as a source for this typical speed-accuracy trade-off behavior.

or phrase [Jong, 1957, Card *et al.*, 1978]:

$$t_n = \frac{t_1}{n^x} \tag{4.1}$$

where t_n is the average time for operation n, t_1 is the time for the first operation, and x is estimated from the data. Values for x must be between 0 and 1. Typical values for x are around 0.32 [Jong, 1957]. The law can be written to describe the rate of doing these individual operations in the form [McQueen *et al.*, 1995]:

$$r_n = r_1 n^x \tag{4.2}$$

where r_n is the rate (operations per unit of time) at which the work proceeds during repetition n, r_1 is the rate during the first repetition and x is again estimated from measured data. Both curves are linear in two dimensional *log-log* space making the use of linear regression easy for estimating x.

Measured performance is known to initially follow the power law. In fact the law usually holds long enough to make it seem to hold forever in the light of experimental results. Clearly, this cannot be the case. Because the Fitts' law based modeling techniques can be used to calculate an estimate for the upper limit of text entry speed, we attempted to combine power law and the upper limit in a model that would more accurately reflect user behavior in real-world situations. This work is presented in paper VI.

4.1.2 Discussion

After presenting paper VI at CHI 2003, we were informed[2] that similar work has been done before. The paper in question appears to be the one published by De Jong in 1957 [1957]. Because of the similarities it is worthwhile to discuss the differences between our approach and that by De Jong.

De Jong is mainly concerned with the duration of repetitive tasks in industrial settings where it has economical consequences. For example if workers are paid bonuses based on above-normal performance, it is important to know what is normal. Because workers' skill increases over time, the incentive programs must be structured to take this into account. On a higher level planning of production needs to take into account the increasing rate at which the work happens so that different batches of products can be scheduled reliably to avoid costly idle hands in the factories.

De Jong cites earlier work as a source for the basic power law that is presented in the form:

$$T_s = \frac{T_1}{s^m} \tag{4.3}$$

where T_1 is the time required for the first cycle of the repeating task, T_s is the time for the cycle number s, and m is the "exponent of the reduction".

[2]By Stuart K. Card

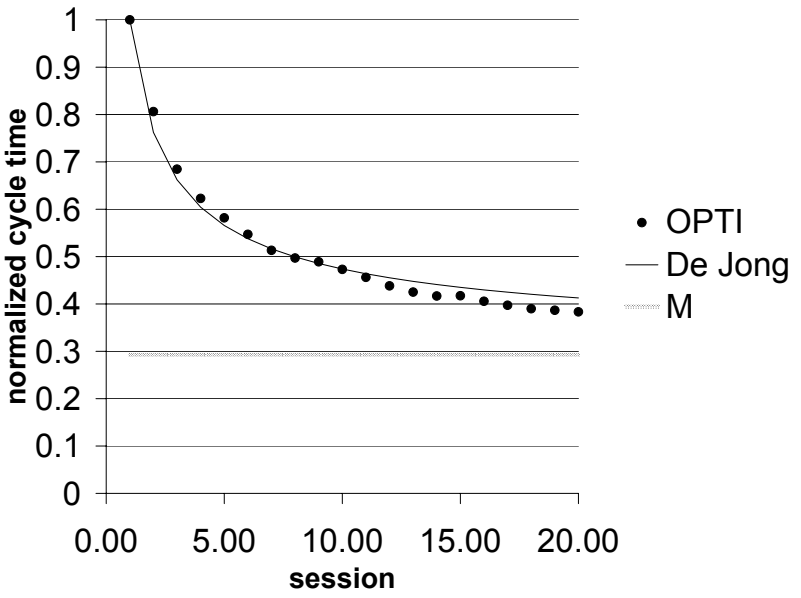

Figure 4.1: Per session average cycle times of MacKenzie and Zhang [1999] and a model after De Jong's equation 3.

De Jong introduces the concept of "factor of incompressibility" denoted by M. And gives an example where M is used to describe the fall of cycle times using the formula:

$$T_s = T_1(M + \frac{1-M}{s^m}) \tag{4.4}$$

De Jong notes that this equation explains the situation where the fall of the cycle time is limited by a hard lower limit. He does not claim the account to be perfect. Instead he describes it "satisfactory". Indeed, as Figure 4.1 reveals, equation 4.4 suffers from the same phenomenon as model 1 in paper VI. It does not fit the data very well. The curve is too tight in the early part and too straight in the later parts. Furthermore, the exponent m is not naturally produced in the process. It needs to be estimated separately. Note that I am not using repetition cycle count as the unit on the horizontal axis. Instead the units in figure 4.1 are sessions. In this case the change does not matter. The same relationship between De Jong's equation and the OPTI data can be observed in a plot with the cycle count on horizontal axis.

However, De Jong's factor of incompressibility suggest another approach that can be combined with Model 1 in Paper VI to produce an improved version of model 1. First M is calculated. This can be done by finding the upper limit of text entry speed R_{max} and calculating the time needed per character T_{min}. M is then $\frac{T_{min}}{T_1}$. Then the time spent per character is normalized so that $T_1 = 1$ and then M is subtracted from these normalized values. At this point the data looks like figure 4.1 except that the points have been shifted down by M (with the OPTI data $M = 0.29$). Now the best fitting power law curve is found

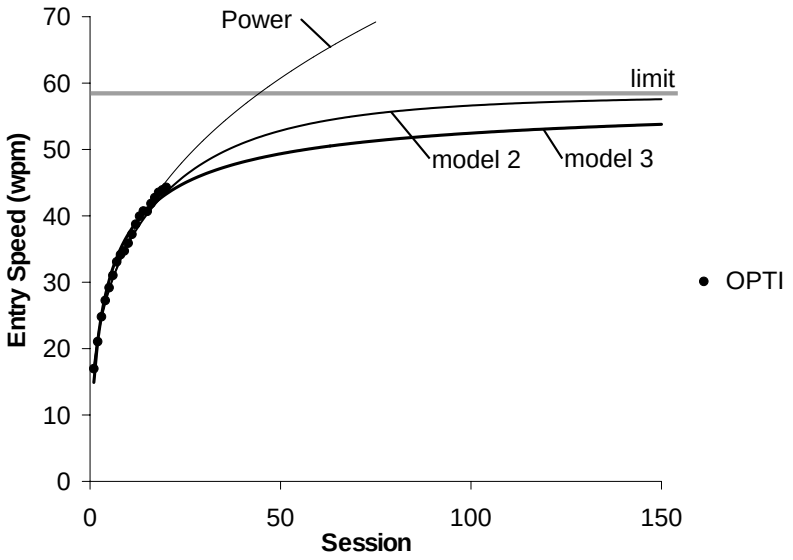

Figure 4.2: Comparison of mid-range predictions of three models on the OPTI data by MacKenzie and Zhang [1999]. Model 3 is the new model, model 2 is from paper VI, and Power is the traditional power law prediction.

through log log linear regression. In the case of the OPTI data the equation is $T_s - 0.29 = 0.8475s^{-0.712}$. The cycle times can be approximated and thus the text entry rates calculated for any positive s. The approximations are limited from above by R_{max} which was the point of the whole exercise.

In Figure 4.2 the resulting curve is compared to the traditional power law and model 2 from paper VI up to session 150. The new model (model 3) curves still slightly too much in the early part and too little in the later part. The advantage of this new procedure over model 1 is that it improves the model fit in terms of R^2. With our example data the R^2 for model 1 was 0.92. With the new model it is 0.99. With De Jong's equation 3 the correlation is about the same, but there is the extra trouble of estimating m. In comparison to model 2 in paper VI, both De Jong's equation 3 and the new model produce lower medium range predictions. It is not known which of the medium range trends is more accurate. In the early part of the medium range predictions the tendency of all of the models to under-estimate the last measured points suggests that model 2 may be more accurate.

On the whole, the purpose of these models is to maximize the use of the expensively acquired experimental data by allowing reliable extrapolations beyond the end of the experiment. The other facet of this issue is that if reliable models can be developed, we can run shorter experiments. If, for example we are interested in user performance after ten hours of practice, we could compute R_{max} , measure a couple of hours of performance and then model the performance at 10 hours saving eight hours per participant or making it possible to get a more representative sample of the user population by running

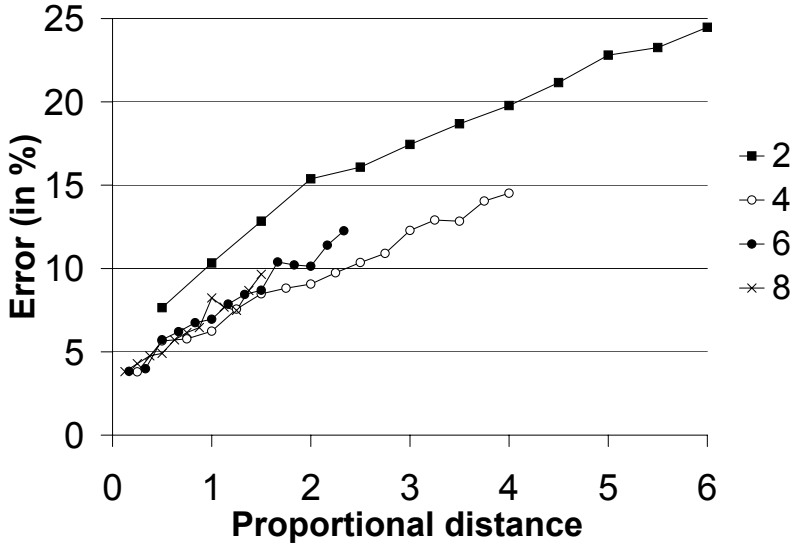

Figure 4.3: The average error in predicting text entry rate development with the power law after using 2, 4, 6, and 8 first sessions for building the model.

five times the number of participants in the same amount of time.

For this kind of use, we need to know how accurate the models are. An estimate can be found by examining published data on longitudinal text entry experiments. I did this for 15 data sets from 8 different papers [Gopher and Raij, 1988, Matias *et al.*, 1996, McQueen *et al.*, 1995, Isokoski and Käki, 2002, MacKenzie *et al.*, 2001, MacKenzie and Zhang, 1999, Isokoski and Raisamo, 2003b, Isokoski, 2004]. The data sets were chosen based on their length (minimum of 20 sessions) and suitability of the text entry rate data for naive power law modeling (the one-handed chord keyboard data by gopher and Raij was rejected because it has too steep a slope between sessions 1 and 2). All data were modeled by using 2, 4, 6, and 8 first points for double log linear regression to determine the power law coefficients. The remaining points were then re-created using the model, and the difference (in %) between the model and the measured value calculated. The results are shown in Figure 4.3. The horizontal axis is proportional to the number of points used so that at 1 the two-point model is predicting point 4, four-point model is predicting point 8, and so on. We see that the two point model is somewhat weaker than the others. The 4, 6, and 8 point models can predict roughly at 7% error rate as far into the future as the length of data that they were built on. The error exceeds 10% at around two times the length of data used for building the models.

Except for the two point model the results seem encouraging. It appears that if we are willing to accept a $\pm 10\%$ error, we can save two thirds of the sessions in a given experiment. Unfortunately the truth is not as nice. The 10% error is the average. The actual errors may be larger. In the examined data

there were several examples of learning curves that seemed to jump up or down after 1-4 sessions. Such jumps may be the result of change in the participants' motivation or strategy in completing the text enty task or a feature of the learning process such as overcoming some initial difficulty. Regardless of the reasons of these anomalies in the curves, the consequence is that the very early performance cannot be relied to develop consistently in the long run.

The effect that the combined models discussed above would have on the results in Figure 4.3 is a small increase in the error. The reason for this is that the combined models tend to underestimate the text entry rate slightly. In the basic power law models used to create Figure 4.3 there were roughly equal number of cases where the models tended to overestimate, be pretty much correct, and underestimate the data. Under these conditions adding a slight bias toward underestimating increases the overall average error. In this light the combined models seem bad. However, this is not what they are made for. The goal in their development was to remove the gross over-estimation that unbounded power curves have in the long run.

4.2 Model for unistroke writing time

4.2.1 Introduction

The design of handwriting systems has been a surprisingly popular hobby. Especially in the era preceding computers, many people who wrote a lot had their own variations of a mixture of short hand and regular handwriting. The critical difference that computers have brought to the situation is that the writing no longer needs to be legible on paper. It is enough that computer can translate it to text.

In order to design efficient character sets for computer input, we need to know what factors govern the efficiency. It seems intuitively clear that the more strokes and corners a character consists of the more time it takes to draw it. The accuracy of this simple model is explored in paper V.

4.2.2 Discussion

While the accuracy of the model in describing or predicting the time consumption per individual instance of character is poor due to natural variation, a strong linear relationship between the character complexity and writing time emerges when writing time is averaged over several instances of the character. Averaging over users further strengthens the relationship. Finally if all characters are pooled according to their complexity, a picture like that shown in Figure 4.4 emerges.

Each point in Figure 4.4 represents the average writing time of all characters of a given character set that belong to the same complexity class. The correlations between the complexity and writing time are surprisingly high. MDITIM received the highest correlation ($r^2 = 0.992$). This is partially explained by the nature of the characters that consist of straight lines connected by 90 and 180 degree corners. Additionally MDITIM has only 3 different complexity

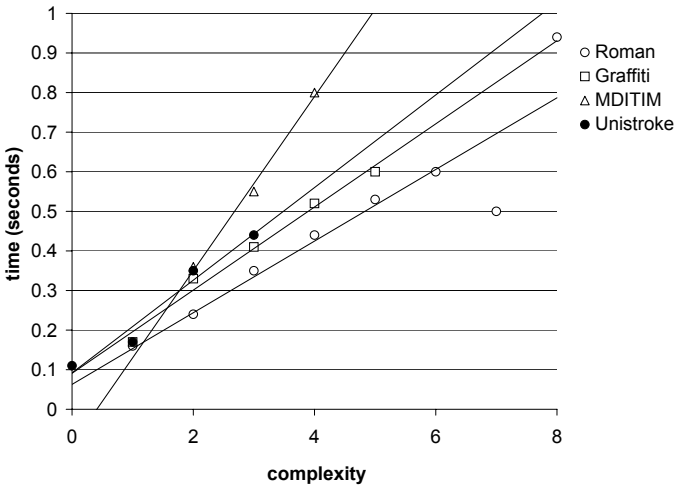

Figure 4.4: Average writing times vs. complexity class for all four tested character sets.

classes making a high correlation likely. Unistrokes ($r^2 = 0.969$) has only four complexity classes. Graffiti ($r^2 = 0.989$) has five and the Roman hand printing characters ($r^2 = 0.851$) have eight. The relatively low correlation for the Roman characters is due to the poorly fitting points for complexities 7 and 8. These points represent only one character written by only one participant each. Removing them increases the r^2 value to 0.997.

Another feature of the data shown in Figure 4.4 is that the slopes of the regression lines vary. MDITIM has the steepest slope (0.22 seconds per complexity unit), Unistrokes are next (0.117), followed closely by Graffiti (0.105) and Roman characters (0.091). This order is the same as the order of familiarity that the participants had with the character sets. Writing the Roman characters is close to pure motor activity and the other character sets require more cognitive involvement that slows the performance down. The earlier work that the model is partly based on presented a rule of thumb that states that we have roughly 5 Herz hands. That means that we can perform a controlled movement about 10 times a second[3]. Our data with the more familiar character sets suggests that the model successfully extracts these movements from the character shapes.

[3]Approximating sine wave for the frequency measurement requires a movement to one direction and a movement back. Thus, 5 Hz=10 movements per second. Other examples of this can be found in the key repeat time measurements by Soukoreff and MacKenzie [2002] and Silfverberg et al. [2000]. Similar figures are cited by Card *et al.* [1983]

4.3 Modeling Menu-augmented soft-keyboards

Paper IV includes two parts: modeling and experimental work. Some aspects of the modeling work are discussed below.

4.3.1 Introduction

The traditional approach to the modeling of expert soft keyboard tapping has been to use spreadsheets with matrixes for key distances and digram frequencies [Soukoreff and MacKenzie, 1995]. This approach works well and is not very labor intensive for plain soft keyboards. However, if the layout is dynamic or if the user interface contains other components that combine in a multiplicative manner with the keys, the distance tables tend to grow. The threshold where the complexity becomes unbearable depends on the researcher performing the modeling. At some point however, alternative techniques become attractive.

One way to circumvent the complex spreadsheet calculations is to write a program that simulates the user's stylus or finger movements. The computational complexity of this approach is in linear relationship to the size of the text corpus that is used for the simulations. A more sophisticated approach could condense the corpus to, for example n-gram frequencies (with n suitable to the simulated text entry technique), simulate each n-gram once, and weight the results according to the frequency. Such approach has roughly the same computational complexity as the spreadsheet approach (essentially constant time operation regardless of the size of the corpus once the n-gram frequencies are known).

In paper IV I was faced with the task of simulating a mixture of soft keyboard tapping and menu selection activity. I used the naive approach of simulating the whole corpus. With corpora of moderate size (less than a million characters) the simulations do not take very long to run on modern computers. I used a very small corpus of only about 15000 characters.

4.3.2 Discussion

The validity of the modeling results remains unverified. However, the validity is presumably as good as it is with the spreadsheet approach or any other means of calculating the same numbers. Overall, no hard upper limit for text entry speed exists. Because we do not know precisely the level of expertise that we are modeling, the produced estimates of expert performance are likely to be somewhat inaccurate. Thus, the modeled upper limits should not be interpreted too strictly. Another interpretation of the modeling results is to compare the results of different text entry systems. This was the approach taken in Paper IV. I ran the simulations for different soft keyboard layouts. Regardless of whether the magnitude of the simulated text entry rates is correct, we can expect the relative differences between the layouts to be accurately reflected.

4.4 Future Work

The modeling of handwriting characters could be conveniently explored with a suitable software package. The work in paper V was done partially as an early feasibility study in order to find out whether there is room to exceed the accuracy of human intuition with suitable tools. This seems to be the case. Human ability to figure out the time consumption of a character using only paper and pencil seems limited. The construction of the software has not been finished. It might be worth doing.

The work on the learning curve models should be continued as well. The work reported above has concentrated on data fitting only. A more theoretical approach could produce refined models that could in addition to being theoretically sound be tunable depending on the parameters of the task and measured performance. A model that could produce upper limit prediction for text entry rate based on data recorded over a number of sessions would be especially nifty.

Modeling expert performance with soft keyboards using Fitts' law based models is beginning to be a routine procedure. However, as detailed in paper IV, there are a number of issues on which a widespread consensus does not exist. For example my choice of using the Fitts' law intercept for modeling repeating taps on a key is seems to be supported by some [Zhai *et al.*, 2002b], while others find it ridiculous [Soukoreff and MacKenzie, 2002]. Such controversies should be solved and an unified methodology developed to increase the inter-study comparability of modeling results. Setting up a open source software package with capabilities for both digraph table and simulation based modeling would allow easy comparison between a baseline model and any new developments that may happen in the future.

Chapter 5

Systems

Constructive research produces knowledge and systems. The papers in chapters 3 and 4 describe the knowledge gained through experiments where the produced systems have been used. In the paper discussed in this chapter the system has the main role.

5.1 Text Input Architecture

5.1.1 Introduction

Paper VII presents a text input architecture that supports personalization of text entry methods. The personalization happens through user-specific configurations that the user provides for the system when he or she begins to use it. Text entry methods are implemented as modules that are loaded over the Internet when needed.

This architecture is the result of evolution that has lasted for five years. At first I began by writing separate pieces of software for each computation and experimental prototype. Work in Paper I was done with this approach. Soon it became apparent that this style of work was unnecessarily laborious. The next step was to combine the common parts of the software into a framework that could be easily extended with new text entry methods. This framework was implemented in C++ and ran under Linux and X. Papers II and V report work done with this framework.

Finally it became apparent that operating system dependencies should be minimized. I chose Java as the platform for the next framework. Operating system dependent code was separated from the core of the framework and implemented separately for Linux and Microsoft Windows. Papers III and IV report work done with this latest generation of the framework.

5.1.2 Discussion

The basic notion of the architecture seems alien to the present way of making and marketing computing devices and software. Very little emphasis is currently put on the user's ability to transfer his or her data and skills between devices from different manufacturers and device generations. I expect that the

time will come when the only valid marketing argument is the service that a particular device or piece of software can offer to its user. When viewed from this perspective, the ability to transport the user's preferred text entry system onto whichever device he or she is using is a basic requirement that must be satisfied. It might take a long time before we are this far. Other aspects of technology can be improved for years before there is a real need to take user interface standardization seriously enough in the area that the architecture in Paper VII addresses. It is also possible that the development takes a path that avoids the need to have user specific text entry methods. If everybody writes only English and agrees to use only one or a small number of input devices to do it, the problem that I have tried to solve disappears.

5.2 Future Work

The existing implementations of the architecture are for desktop computers. This platform is pretty much the only one with adequate text entry capabilities and a user base well trained in their use. Therefore desktop computers have the smallest need for this kind of an architecture. Implementations for the Symbian smart phone platforms or Palm or Microsoft PDA operating systems would be more useful. So far I have not done any of these since the desktop platforms are easier to work with and adequate for demonstration and research purposes. If the architecture is to be of any practical use, the implementations for other platforms need to be completed.

Chapter 6

Discussion

I have described experiments and models partially based on the results of these experiments. In this chapter I discuss some of the general limitations that apply to the work.

6.1 Experimental Methodology

The experiments done in the papers are somewhere in between a typical usability evaluation and a proper experiment in rigor. The goal was to do work with as good internal and external validity as possible given the practical limitations. Each experiment was typically preceded by a pilot phase that consisted of iterative usability testing of the experimental procedure. Changes were often made to help the participant focus on the essential parts of the task and to improve the working conditions of the experimenter.

6.1.1 Experimenter Bias

The experiments to evaluate the new text entry techniques were conducted by the inventor. It is possible that the enthusiasm of the experimenter may have influenced the participants. It is customary in other sciences such as medicine to perform evaluations with the double-blind protocol. In this protocol the treatment (for example a new drug) is compared against a placebo treatment that is known to have no medical effect or a known competing treatment. The people who interact with the participants do not know which of the treatments is placebo and which is for real. Because of this they cannot influence the participants perceptions and motivations. The difficulty of applying this protocol to user interface evaluations is that developing a placebo user interface is often very difficult. The participants can usually easily deduce the experimental setup based on their previous experience. Nevertheless, we must be aware of these issues both when designing experiments and when reading and interpreting reports on such experiments. I suspect that subjective evaluations are much more sensitive to whatever bias the experimenter may exert upon the participants. This explains the relative scarcity of subjective data in this thesis.

6.1.2 Sampling methods

Another problematic part of the experiments reported in this thesis is the representativeness of the participants. In all cases the participants were recruited from nearby offices of whatever part of the university I happened to be working in. Other than being convenient for me, this procedure required the smallest possible amount of work from the participants. The experiments were typically longitudinal consisting of 10-20 sessions. Because I did not have resources to compensate the work that the participants did for me, I deemed it unlikely to be able to recruit participants from a sample of the general public.

However, the result of this sampling protocol is that not only were the participants typically young male adults with university education, but they were also very experienced computer users, and in most cases worked as HCI researchers. If these factors affect a person's performance in experiments like mine, the conclusions drawn based on these experiments may not be representative of the general public.

6.1.3 Language issues

Language is an issue in some of the experiments. It appears that remembering and entering a phrase in a foreign language is more difficult than in the native language. I did the experiments using English phrases. This does not necessarily invalidate the results in situations where two systems are compared under the same conditions. However, cross-study comparisons with studies done on native English speakers should take the language issue into account.

6.1.4 Choice of metrics

When designing experiments one has to decide which things will be measured and how. In all of the work presented in this thesis, I have used efficiency metrics almost exclusively. In the light of the summary data by Nielsen and Levy [1994], performance measures are correlated with subjective preference. On the other hand it has been suggested that relying on one or the other is a dangerously narrow approach. For example Fokjaer et al. [2000] suggest that effectiveness, efficiency, and subjective satisfaction should all be investigated unless it has been shown that in a particular task some aspects do not matter.

These arguments have been cast in the context of usability in general. Whether text entry is a special case where performance in the form of efficiency is the dominant factor of usability and usefulness has not been shown in general. However, efficiency emphasis can be defended as a relevant approach is some areas of text entry. Namely, efficiency is always good in situations where time is money. For example, in transcription typing a slow way of typing is difficult to justify economically. Generally a user whose goal is to be efficient in his or her work will appreciate efficient user interfaces. Strangely enough, there are other uses of text entry where efficiency can actually be a bad thing. For example when people entertain themselves by writing SMS messages, they get more entertainment for a given amount of money if writing is not too efficient because only sending the messages costs.

Overall, the efficiency emphasis is a feature of the reported work. Efficiency should not be confused with the overall preferability of a given text entry method except when it is clear that the two are synonymous because of the nature of the task and needs of the users.

6.1.5 Replication

An important part of rigorous scientific work is independent replication of experimental results. Even when proper care is taken to minimize factors like experimenter bias, skewed sampling, and opportunistic choice of metrics, the fact remains that the experimenter has many interests vested in the experiment. It is possible that sometimes the observed effects are not due to the treatment that is administered. Even if no foul play from the part of the experimenters can be found, statistical conclusions contain a margin of error.

For these reasons it makes sense to replicate important experiments independently in different laboratories using different sample of the user population and experimental apparatus. If the results still hold, it is far more unlikely that it is due to chance or some unnoticed influence by the experimenters.

In HCI there is no systematic tradition of replicating experiments. In fact successful replications with no other contributions are practically impossible to publish. They are considered un-original and therefore worthless. When replication happens it is mostly because of ignorance of the original work or because another team of researchers wants to continue on the work of others and need access to data similar to what has been previously reported in order to make comparisons.

The work that I report in this thesis has not been independently replicated to verify its validity. The work does contain a small amount of internal replication since experiments were preceded by pilot experiments that were used to test the procedure. However, the power of such internal replications to reveal significant flaws in the whole setup is small. As such the work must be considered tentative until independent evidence on its validity appears.

The reported experiments themselves contain instances of partial replication on previous work. The pure clock face condition in Paper II replicates earlier work in a slightly different environment (touchpad instead of a stylus). The re-implementation of Quikwriting (Paper III) is another instance of replication as well as the stylus tapping model in Paper IV. Mostly the results confirm earlier findings. A notable exception is the case of Quikwriting where we did not observe the kind of general superiority to other writing systems that had been (informally) claimed.

Chapter 7

Conclusions

I have presented new text entry methods, results of modeling different aspects of text entry activity, and a new system for personalized text entry. While many of the results may be interesting, no salient steps forward can be pointed out. This is not surprising considering the very long history of writing. In fact, it would be highly suprising to stumble on a completely new and efficient method at this late stage in history. Consequently, the presented work consists of improvements on earlier work and new combinations of previously known systems and methods.

One of the goals listed in my original research plan was to develop guidelines for selecting an appropriate text entry method for a given task, device, or user. Despite considerable effort, the results in this respect are slim. The results of the experiments as well as some of the modeling work can be used for this purpose, but they are only small pieces in the puzzle that must be considered, not suitable for general guidelines. The only general guideline that I have found reliable is that in a short time perspective *the best text entry method is the one that the user knows.* Just about anything else requires a lengthy learning before it becomes useful and even longer before it performs any better than a system familiar to the user.

Several themes of research have been started in the course of the thesis work. Some of these deserve further investigation. One of the unfinished issues is the relationship between pointing device throughput and text entry throughput. Pointing device performance can be characterized with Fitts' law and a text stream has a certain information content. Combining these notions into one theory of information throughput has been hinted at numerous times. However, no useful conceptualization has emerged.

Another issue that continues to stimulate my curiosity is the notion of device independence. It could be possible to develop text entry methods that work well enough on all input devices to make it unattractive to learn any other methods. Unfortunately we do not know whether the fact that no such methods have emerged is due to lack of imagination or because they are impossible.

Finally, the text input architecture work is worth continuing. Device and operating system platform independent text entry methods make sense as user interface components and as a software development model in this particular

case. They may not make economical sense because they encourage standardization and free availability of text entry methods, but this can only hinder making money with the idea, not researching it.

Bibliography

[3Com, 1997] *PalmPilot Handbook*. 3Com Corporation, 1997.

[Aaltonen *et al.*, 1998] Antti Aaltonen, Aulikki Hyrskykari, and Kari-Jouko Räihä. 101 spots, or how do users read menus? In *Proceedings of CHI '98*, pages 132 – 139. ACM, 1998.

[Accot and Zhai, 1997] Johnny Accot and Shumin Zhai. Beyond fitts' law: Models for trajectory-based hci tasks. In *Proceedings of CHI '97*, pages 295 – 302. ACM, 1997.

[AOL, 2003] T9 text input, 2003. http://www.t9.com.

[Bailey, 1989] R. W. Bailey. *Human Performance Engineering*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 2 edition, 1989.

[Barber, 1997] Christopher Barber. *Beyond the Desktop: Designing and Using Interaction Devices*. Academic Press, San Diego, California, USA, 1997.

[Bellaire Electronics, 2003] CyKey, 2003. http://www.bellaire.co.uk/.

[Bellman and MacKenzie, 1998] Tom Bellman and I. Scott MacKenzie. Probabilistic character layout strategy for mobile text entry. In *Proceedings of Graphics Interface '98*, pages 168 – 176. Canadian Information Processing Society, 1998.

[Brooks, 2000] Marcus Brooks. Introducing the dvorak keyboard, 2000. http://www.mwbrooks.com/dvorak.

[Byrne *et al.*, 1999] Michael D. Byrne, John R. Anderson, Scott Douglass, and Michael Matessa. Eye tracking the visual search of click-down menus. In *Proceedings of CHI '99*, pages 402 – 409. ACM, 1999.

[Card *et al.*, 1978] Stuart K. Card, William K. English, and Betty J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 21(8):601 – 613, 1978.

[Card *et al.*, 1983] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates Inc., Hillsdate, New Jersey, USA, 1983.

[Chang and MacKenzie, 1994] Larry Chang and I. Scott MacKenzie. A comparison of two handwriting recognizers for pen-based computers. In *Proceedings of CASCON '94*, pages 364 – 371. IBM Canada, 1994.

[Cockburn and Siresena, 2002] Andy Cockburn and Amal Siresena. Evaluating mobile text entry with the fastap keypad. In *Proceedings of the 17th British HCI Group Annual Conference, volume 2*, pages 77 – 70. British HCI Group, 2002.

[Coleman, 2001] Mike Coleman. Weegie home page, 2001. http://weegie.sourceforge.net.

[Digit Wireless, 2003] Fastap, 2003. http://www.digitwireless.com.

[Dunlop and Crossan, 2000] Mark D. Dunlop and Andrew Crossan. Predictive text entry methods for mobile phones. *Personal Technologies*, 4(2):134 – 143, 2000.

[FingerWorks, 2003] Touchstream keyboard, 2003. http://www.fingerwors.com.

[Fitts, 1954] Paul M. Fitts. The information capacity of the human motor system in controlling the ampliture of movement. *Journal of Experimental Psychology*, 47(6):381 – 391, 1954.

[Frankish *et al.*, 1995] Clive Frankish, Richar Hull, and Pam Morgan. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings of CHI '94*, pages 503 – 510. ACM, 1995.

[Frøkjær *et al.*, 2000] Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk. Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated? *CHI 2000, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2(1):345 – 352, 2000.

[Gaur, 1987] Albertine Gaur. *A history of writing*. The British Library, London, UK, 2 edition, 1987.

[Goldstein *et al.*, 1999] Mikael Goldstein, Robet Brook, Gunilla Alsiö, and Silvia Tessa. Non-keyboard qwerty touch typing: A portable input interfce for the mobile user. In *Proceedings of CHI '99*, pages 32 – 39. ACM, 1999.

[Goodman *et al.*, 2002] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. Language modeling for soft keyboards. In *Proceedings of Eighteenth National Conference on Artificial Intelligence*, pages 419 – 424. American Association for Artificaial Intelligence, 2002.

[Gopher and Raij, 1988] D. Gopher and D. Raij. Typing on a two-handed chord keyboard: Will QWERTY become obsolete? *IEEE Transactions on Systems, Man, and Cybernetics*, 18:601 – 609, 1988.

[Grimberg, 1967] Grimberg. *Maailmanhistoria*. WSOY, Finland, 4 edition, 1967.

[Handykey Corporation, 2003] Twiddler2, 2003. http://www.handykey.com/site/twiddler2.html.

[Himberg *et al.*, 2003] Johan Himberg, Jonna Häkkilä, Petri Kangas, and Jani Mäntyjärvi. On-line personalization of a touch screen based keyboard. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces*, pages 77 – 84. ACM Press, 2003.

[Hughes *et al.*, 2002] Dominic Hughes, James Warren, and Orkut Buyukkokten. Empirical bi-action tables: A tool for the evaluation and optimization of text-input systems. application i: Stylus keyvoards. *Human-Computer Interaction*, 17(2&3):271 – 310, 2002.

[Infogrip Inc., 2003] Bat personal keyboard, 2003. http://www.infogrip.com/bat_kybd_details.asp.

[Isokoski and Käki, 2002] Poika Isokoski and Mika Käki. Comparison of two touchpad-based methods for numeric entry. *CHI 2002, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 4(1):25 – 32, 2002.

[Isokoski and MacKenzie, 2003] Poika Isokoski and I. Scott MacKenzie. Combined model for text entry rate development. In *CHI 2003 Extended Abstracts*, pages 752–753. ACM Press, 2003.

[Isokoski and Raisamo, 2000] Poika Isokoski and Roope Raisamo. Device independent text input: A rationale and an example. In *Proceedings of the International Working Conference on Advanced Visual Interfaces AVI2000*, pages 76 – 83. ACM, 2000.

[Isokoski and Raisamo, 2003a] Poika Isokoski and Roope Raisamo. Architecture for personal text entry methods. In Morten Borup Harning and Jean Vanderdonckt, editors, *Closing the Gaps: Software Engineering and Human-Computer Interaction*, pages 1 – 8. IFIP, 2003.

[Isokoski and Raisamo, 2003b] Poika Isokoski and Roope Raisamo. Evaluation of a multi-device extension of quikwriting. Report A-2003-5, Department of Computer Sciences, FIN-33014, University of Tampere, Finland, 2003.

[Isokoski, 2001] Poika Isokoski. Model for unistroke writing time. *CHI 2001, Human Factors in Computing Systems, CHI Letters*, 3(1):357 – 364, 2001.

[Isokoski, 2004] Poika Isokoski. Performance of menu-augmented soft keyboards. *CHI 2004, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 6(1):–, 2004.

[John and Kieras, 1996] Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320 – 351, 1996.

[Jong, 1957] J. R. De Jong. The effects of increasing skill on cycle time and its consequences for time standards. *Ergonomics*, 6:51 – 60, 1957.

[King *et al.*, 1995] Martin T. King, Cliffor A. Kushler, and Dale A. Grover. Justtype - efficient communication with eight keys. In *Proceedings of the RESNA '95*, pages 94 – 96, 1995.

[Kober *et al.*, 2001] Hedy Kober, Eugene Skepner, Terry Jones, Howard Gutowitz, and Scott MacKenzie. Linguistically optimized text entry on a mobile phone. Report, Eatoni Ergonomics Inc., 171 Madison Avenue, New York, New York 10016, USA, 2001.

[Koester and Levine, 1994] Heidi Horstmann Koester and Simon P. Levine. Modeling the speed of text entry with a word prediction interface. *IEEE Transactions on Rehabilitation Engineering*, 2(3):177 – 187, 1994.

[Kurtenbach and Buxton, 1993] Gordon Kurtenbach and William Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of INTERCHI '93*, pages 482 – 487. ACM, 1993.

[LaLomia, 1994] Mary LaLomia. User acceptance of handwritten recognition accuracy. In *Proceedings of the '94*, page 107. ACM, 1994.

[Lehikoinen and Salminen, 2002] J. Lehikoinen and I. Salminen. An empirical and theoretical evaluation of binscroll: A rapid selection technique for alphanumeric lists. *Personal an Ubiquituous Computing*, 6(2):141 – 150, 2002.

[Long *et al.*, 1999] A. Chris Long, James A. Landay, and Lawrence A. Rowe. Implication for a gesture design tool. In *Proceedings of CHI '99*, pages 40 – 47. ACM, 1999.

[Long *et al.*, 2000] A. Chris Long, James A. Landay, Lawrence A. Rowe, and Joseph Michiels. Visual similarity of pen gestures. *CHI 2000, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2(1):360 – 367, 2000.

[MacKenzie and Soukoreff, 2002a] I. Scott MacKenzie and William Soukoreff. A model for two-thumb text entry. In *Proceedings of Graphics Interface 2002*, pages 117 – 124. Canadian Information Processing Society, 2002.

[MacKenzie and Soukoreff, 2002b] I. Scott MacKenzie and William Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17(2&3):147 – 198, 2002.

[MacKenzie and Zhang, 1999] I. Scott MacKenzie and Shawn X. Zhang. Design and evaluation of a high-performance soft keyboard. In *Proceedings of CHI '99*, pages 25 – 31. ACM Press, 1999.

[MacKenzie *et al.*, 1994] I. S. MacKenzie, R. Blair Nonnecke, J. Craig McQueen, Stan Riddersma, and Malcolm Meltz. A comparison of three methods of character entry on pen-based computers. In *Proceedings of the Human Factors and Ergonomics Society 38th Annual Meeting*, pages 330 – 334. Human Factors Society, 1994.

[MacKenzie *et al.*, 1999] I. Scott MacKenzie, Shawn X. Zhang, and William Soukoreff. Text entry using soft keyboards. *Behaviour and Information Technology*, 18:235 – 244, 1999.

[MacKenzie *et al.*, 2001] I. S. MacKenzie, H. Kober, T. Jones, and E. Skepner. Letterwise: Prefix-based disambiguation for mobile text input. *UIST 2001, ACM Symposium on User Interface and Software Technology, CHI Letters*, 3(2):111 – 120, 2001.

[MacKenzie, 1992] I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91 – 139, 1992.

[MacKenzie, 2002a] I. Scott MacKenzie. Introduction to this special issue on text entry for mobile computing. *Human-Computer Interaction*, 17(2&3):141 – 145, 2002.

[MacKenzie, 2002b] I. Scott MacKenzie. Kspc (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the Fourth International Symposium on Human Computer Interaction with Mobile Devices*, pages 195 – 210. Springer Verlag, 2002.

[MacKenzie, 2002c] I. Scott MacKenzie. Mobile text entry using three keys. In *Proceedings of the Second Nordic Conference on Human-Computer Interaction*, pages 27 – 34. ACM Press, 2002.

[MacKenzie, 2003] I. Scott MacKenzie. Motor behavior models for human-computer interaction. In John M. Carroll, editor, *HCI Models, Theories, and Frameworks*, pages 27 – 54. Morgan Kaufmann, 2003.

[Mankoff and Abowd, 1998] Jennifer Mankoff and Gregory D. Abowd. Cirrin: A word-level unistroke keyboard for pen input. In *Proceedings of UIST '98*, pages 213 – 214. ACM Press, 1998.

[Mankoff *et al.*, 2000] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. *CHI 2000, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2(1):368 – 375, 2000.

[Masui, 1998a] Toshiyuki Masui. An efficient text input method for pen-based computers. In *Proceedings of CHI '98*, pages 328 – 335. ACM Press, 1998.

[Masui, 1998b] Toshiyuki Masui. Integrating pen operations for composition by example. In *Proceedings of UIST '98*, pages 211 – 212. ACM Press, 1998.

[Masui, 1999] Toshiyuki Masui. Pobox: An efficient text input method for handheld and ubiquitous computers. In *Proceedings of Handheld and Ubiquitous Computing: First International symposium, HUC '99, Lecture Notes in Computer Science 1707*, pages 289 – 300. Springer Verlag, 1999.

[Matias *et al.*, 1993] Edgar Matias, I. Scott MacKenzie, and William Buxton. Half-qwerty: A one-handed keyboard facilitating skill transfer from qwerty. In *Proceedings of the INTERCHI '93*, pages 88 – 94. ACM Press, 1993.

[Matias *et al.*, 1996] Edgar Matias, I. Scott MacKenzie, and William Buxton. One-handed typing with a qwerty keyboard. *Human-Computer Interaction*, 11:1 – 27, 1996.

[McQueen *et al.*, 1994] C. McQueen, I. S. MacKenzie, B. Nonnecke, and S. Riddersma. A comparison of four methods of numeric entry on pen-based computers. In *Proceedings of Graphics Interface '94*, pages 75 – 82. Canadian Information Proccessing Society, 1994.

[McQueen *et al.*, 1995] J. Craig McQueen, I. Scott MacKenzie, and Shawn X. Zhang. An extended study of numeric entry on pen-based computers. In *Proceedings of Graphics Interface '95*, pages 215 – 222. Canadian Information Proccessing Society, 1995.

[Motorola, 2003] iTap, 2003. http://www.motorola.com/lexicus/html/itap.html.

[Nielsen and Levy, 1994] Jakob Nielsen and Jonathan Levy. Measuring usability: Preverence vs. performance. *Communications of the ACM*, 37(4):66 – 75, 1994.

[Norman, 1991] Kent L. Norman. *Te Psychology of Menu Selection: Designing Cognitive Control at the Human-Computer Interface.* Ablex Publishing, 1991.

[Noyes, 1983] J. Noyes. Chord keyboards. *Applied Ergonomics*, 14:55 – 59, 1983.

[Partridge *et al.*, 2002] Kurt Partridge, Saureav Chatterjee, vibha Sazawal, Gaetano Boriello, and Roy Want. Tilttype: Accelerometer-supported text entry for very small devices. *UIST 2002, ACM Symposium on User Interface and Software Technology, CHI Letters*, 4(2):201 – 204, 2002.

[Pavlovych and Stuerzlinger, 2003] Andriy Pavlovych and Wolfgang Stuerzlinger. Less-tap: A fast and easy-to-learn text input technique for phones. In *Proceedings Fraphics Interface 2003*, pages 319 – 326. Canadian Information Processing Society, 2003.

[Perlin, 1998] Ken Perlin. Quikwriting: Continuous stylus-based text entry. In *Proceedings of UIST '98*, pages 215 – 216. ACM Press, 1998.

[Plamondon and Srihari, 2000] Réjean Plamondon and Sargur N. Srihari. Online and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63 – 84, 2000.

[Potosnak, 1988] K. M. Potosnak. Keys and keyboards. In M. Helander, editor, *Handbook of Human-Computer Interaction*, pages 475 – 494. North-Holland, Amsterdam, 1988.

[Rau and Skiena, 1994] Harald Rau and Stevens S. Skiena. Dialing for documents: an experiment in information theory. In *Proceedings of ACM UIST '94*, pages 147 – 155. ACM Press, 1994.

[Roeber *et al.*, 2003] Helena Roeber, John Bacus, and Carlo Tomasi. Typing in thin air: The canesta projection keyboard - a new method of interaction with electronic devices. In *CHI 2003 Extended abstracts*, pages 712 – 713. ACM Press, 2003.

[Sacher, 1998] Heiko Sacher. Interactions in chinese: Designing interfaces for asian languages. *Interactions*, 5(5):28 – 38, 1998.

[Sandnes *et al.*, 2003] Frode Eika Sandnes, Alexander Arvei, Haarvard Wiik Thorkildssen, and Johannes O. Bruverud. Trikey: Mobile text-entry techniques for three keys, 2003. http://www.iu.hio.no/f̃rodes/trikey.

[Scutliffe, 2000] Alistair Scutliffe. On the effective use and reuse of hci knowledge. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(2):197 – 221, 2000.

[Sears *et al.*, 2001] Andrew Sears, Julie A. Jacko, Josey Chu, and Francisco Moro. The role of visual search in the design of effective soft keyboards. *Behaviour & Information Technology*, 20(3):159–166, 2001.

[Senseboard, 2003] Senseboard, 2003. http://www.senseboard.com.

[Shandbhag *et al.*, 2002] Shrinath Shandbhag, Durgesh Rao, and R. K. Joshi. An intelligent multi-layered inpug scheme for phonetic scripts. In *Proceedings of the 2nd International Symposium on Smart Graphics*, pages 35 – 38. ACM Press, 2002.

[Shannon and Weaver, 1949] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communications*. Univeristy of Illinois Press, 1949.

[Shen *et al.*, 2002] Mowei Shen, Rong Tao, and Ying Liu. Visual search and selection performance of pull-down menu: An eye-tracking research. In *Proceedings of the 5th Asia-Pacific Conference on Human-Computer Interaction (APCHI2002)*, pages 131 – 140. Science Press, Beijing, China, 2002.

[Silfverberg *et al.*, 2000] Mika Silfverberg, I. Scott MacKenzie, and Panu Korhonen. Predicting text entry speeds on mobile phones. *CHI 2000, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 2(1):9 – 16, 2000.

[Soukoreff and MacKenzie, 1995] William Soukoreff and I. Scott MacKenzie. Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour & Information Technology*, 14:370 – 379, 1995.

[Soukoreff and MacKenzie, 2002] R. William Soukoreff and I. Scott MacKenzie. Using fitts' law to model key repeat time in text entry models, 2002. Poster presented at Graphics Interface 2002, available at http://www.yorku.ca/mack/gi02-poster.html.

[Steinherz *et al.*, 1999] Tal Steinherz, Ehud Rivlin, and Nathan Intrator. Offline cursive script word recognition - a survey. *International Journal on Document Analysis and Recognition*, 2:90 – 110, 1999.

[Tappert *et al.*, 1990] Charles C. Tappert, Ching Y. Sue, and Toru Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787 – 808, 1990.

[Textware Solutons, 2002] Dom perignon III speed contest, 2002. http://www.fitaly.com/domperignon/domperignono3.htm.

[Textware Solutons, 2003] The fitaly keyboard, 2003. http://www.fitaly.com/fitaly/fitaly.htm.

[Venolia and Neiberg, 1994] Dan Venolia and Forrest Neiberg. T-cube: a fast, self-disclosing pen-based alphabet. In *Proceedings of CHI '94*, pages 265 – 270. ACM Press, 1994.

[Vinciarelli, 2002] Alessandro Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35:1433 – 1446, 2002.

[Wang *et al.*, 2003] Jingtao Wang, Shumin Zhai, and Hui Su. Chinese input with keyboard and eye-tracking: an anatomical study. *CHI Letters: ACM Conference on Human Factors in Computing Systems, CHI 2001*, 3(1):349 – 356, 2003.

[Ward *et al.*, 2000] David J. Ward, Alan F. Blackwell, and David J.C.MacKay. Dasher - a data entry inferface using continuous gestures and language models. *UIST 2000, ACM Symposium on User Interface and Software Technology, CHI Letters*, 2(2):129 – 137, 2000.

[Ward, 2001] David J. Ward. *Adaptive Computer Interfaces*. PhD thesis, University of Cambridge, 2001.

[Wigdor and Balakrishnan, 2003] Daniel Wigdor and Ravin Balakrishnan. Tilttext: Using tilt for text input to mobile phones. *UIST 2003, ACM Symposium on User Interface and Software Technology, CHI Letters*, 5(2):81 – 90, 2003.

[Wigdor and Balakrishnan, 2004] Daniel Wigdor and Ravin Balakrishnan. A comparison of consecutive and concurrent input text entry techniques for mobile phones. *CHI 2004, ACM Conference on Human Factors in Computng Systems, CHI Letters*, 6(1), 2004. (in press).

[Wobbrock *et al.*, 2003] Jacob. O. Wobbrock, Brad A. Myers, and J. A. Kembel. Edgewrite: A stylus-based text entry method designed for high accuracy and stability of motion. *UIST 2003, ACM Symposium on User Interface Sotware and Technology, CHI Letters*, 5(2):61 – 70, 2003.

[Wolpaw *et al.*, 2002] Jonathan R. Wolpaw, Niels Birbaumer, Dennis J. McFarland, Gert Pfurtcheller, and Theresa M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neuropsychology*, 113:767 – 791, 2002.

[Woolley, 1963] Leonard Woolley. *The Beginnings of Civilization*. History of Mankind: Cultural and Scientific Development. UNESCO, 1963.

[Zagler, 2002] Wolfgang L. Zagler. Matching typing persons and intelligent interfaces: Introduction to the special thematic session. In K. Miesenberger, J. Klaus, and Wolfgang Zagler, editors, *Lecture Notes In Computer Science 2398: ICCHP 2002*, pages 241 – 242. Springer Verlag, 2002.

[Zhai and Kristensson, 2003] Shumin Zhai and Per-Ola Kristensson. Shorthand writing on stylus keyboard. *CHI 2003, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 5(1):97 – 104, 2003.

[Zhai *et al.*, 2002a] Shumin Zhai, Michaer Hunter, and Barton A. Smith. Performance optimization of virtual keyboards. *Human-Computer Interaction*, 17(2&3):229 – 269, 2002.

[Zhai *et al.*, 2002b] Shumin Zhai, Allison Sue, and Johnny Accot. Movement model, hits distribution and learning in virtual keyboarding. *CHI 2002, ACM Conference on Human Factors in Computing Systems, CHI Letters*, 4(1):17 – 24, 2002.

[Zhai, 2003] Shumin Zhai. Evaluation is the worst form of hci research except all those other forms that have been tried, 2003. http://www.almaden.ibm.com/u/zhai/papers/EvaluationDemocracy.htm.

[Zhang, 1998] Shawn X. Zhang. A high performance soft keyboard for mobile systems. Master's thesis, University of Guelph, Canada, 1998.

[Zi Corporation, 2003] Text input, 2003. http://www.zicorp.com/texinputhome.htm.

Appendix A

Paper I

Poika Isokoski and Roope Raisamo, Device Independent Text Input: A Rationale and an Example. *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI2000)*, ACM Press, 2000, 76-83.

Official copy available at:
`http://doi.acm.org/10.1145/345513.345262`
(requires access to the ACM digital library)

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`

Appendix B

Paper II

Poika Isokoski and Mika Käki, Comparison of Two Touchpad-based Methods for Numeric Entry. *CHI 2002, Human Factors in Computing Systems, CHI Letters*, 4(1), ACM Press, 2002, 25-32.

Official copy available at:
`http://doi.acm.org/10.1145/503376.503382`
(requires access to the ACM digital library)

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`

Appendix C

Paper III

Poika Isokoski and Roope Raisamo, *Evaluation of a Multi-Device Extension of Quikwriting.* Report A-2003-5, Department of Computer Sciences, University of Tampere, Finland, 2003.

Official copy available at:
`http://`

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`

Appendix D

Paper IV

Poika Isokoski, Performance of Menu-augmented Soft Keyboards. *CHI 2004, Human Factors in Computing Systems, CHI Letters*, 6(1), ACM Press, 2004, (in press).

Official copy available at:
`http://`
(requires access to the ACM digital library)

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`

Appendix E

Paper V

Poika Isokoski, Model for Unistroke Writing Time. *CHI 2001, Human Factors in Computing Systems, CHI Letters*, 3(1), ACM Press, 2001, 357-364.

Official copy available at:
`http://doi.acm.org/10.1145/365024.365299`
(requires access to the ACM digital library)

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`

Appendix F

Paper VI

Poika Isokoski and Scott MacKenzie, Combined Model for Text Entry Rate Development. *CHI 2003 Extended Abstracts*, ACM Press, 2003, 752 - 753.

Official copy available at:
`http://doi.acm.org/10.1145/765891.765970`
(requires access to the ACM digital library)

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`

Appendix G

Paper VII

Poika Isokoski and Roope Raisamo, Architecture for Personal Text Entry Methods. In Morten Borup Harning and Jean Vanderdonckt (editors), *Closing the Gaps: Software Engineering and Human- Computer Interaction*, IFIP, 2003, 1-8.

Official copy available at:
`http://www.se-hci.org/bridging/interact/proceedings.html`

Unofficial copy available at:
`http://www.cs.uta.fi/~poika/publications.html`