PREDICTING AND REDUCING THE IMPACT OF ERRORS IN CHARACTER-BASED TEXT ENTRY

AHMED SABBIR ARIF

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 2014

Abstract

This dissertation focuses on the effect of errors in character-based text entry techniques. The effect of errors is targeted from theoretical, behavioral, and practical standpoints. This document starts with a review of the existing literature. It then presents results of a user study that investigated the effect of different error correction conditions on popular text entry performance metrics. Results showed that the way errors are handled has a significant effect on all frequently used error metrics. The outcomes also provided an understanding of how users notice and correct errors. Building on this, the dissertation then presents a new high-level and method-agnostic model for predicting the cost of error correction with a given text entry technique. Unlike the existing models, it accounts for both human and system factors and is general enough to be used with most character-based techniques. A user study verified the model through measuring the effects of a faulty keyboard on text entry performance. Subsequently, the work then explores the potential user adaptation to a gesture recognizer's misrecognitions in two user studies. Results revealed that users gradually adapt to misrecognition errors by replacing the erroneous gestures with alternative ones, if available. Also, users adapt to a frequently misrecognized gesture faster if it occurs more frequently than the other error-prone gestures. Finally, this work presents a new hybrid approach to simulate pressure detection on standard touchscreens. The new approach combines the existing touch-point- and time-based methods. Results of two user studies showed that it can simulate pressure detection more reliably for at least two pressure levels: *regular* (~1 N) and *extra* (~3 N). Then, a new pressure-based text entry technique is presented that does not require tapping outside the virtual keyboard to reject an incorrect or unwanted prediction. Instead, the technique requires users to apply extra pressure for the tap on the next target key. The performance of the new technique was compared with the conventional technique in a user study. Results showed that for inputting short English phrases with 10% non-dictionary words, the new technique increases entry speed by 9% and decreases error rates by 25%. Also, most users (83%) favor the new technique over the conventional one. Together, the research presented in this dissertation gives more insight into on how errors affect text entry and also presents improved text entry methods.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Wolfgang Stuerzlinger, for his guidance, support, and funding through his GRAND-NCE and NSERC grants.

I would also like to thank the members of my dissertation committee, Dr. Parke Godfrey, Dr. Jimmy Huang, Dr. I. Scott MacKenzie, Dr. Daniel J. Wigdor, and Dr. Melody Wiseheart, for taking time out of their busy schedules to serve on my dissertation examination committee. Their valuable feedback has made this work substantially better.

My sincere thanks go to the staff in the Department of Computer Science and Engineering, especially Seela Balkissoon, Susan Cameron, Ouma Jaipaul-Gill, and Ulya Yigit, for guiding me through York University's administrative process. In addition, I would like to thank the current and former members of our research lab, especially Michelle Brown, Steven J. Castellucci, Euclides José de Mendonça Filho, Andriy Pavlovych, Dmitri Shuralyov, Robert J. Teather, Hussain Tinwala, and my closest friends, Kazi Tanzin Ahmed, Marcel Birkner, Geri Grolinger, M. Arafat Hossain, M. Selim Hossain, Nelson Moniz, Olga Mushtaler, Nassim Nasser, Tomasz R. Nykiel, Shibly Sadiq Shoeb, Humaira Siddiqa, Tareq Ahmed Siraj, and Jaisingh Solanki for being there for me.

Special thanks go to NSERC, the Government of Ontario, York University, and CUPE 3093 for financial support through various funds and scholarships.

Last but not least, I would like to thank my parents, M. A. Mannan and Shireen Jahan, my younger brother, Ahmed Sazzid Arif, and my loving wife, Nadia Mustafa, for their unconditional love, encouragement, and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The earliest identifiable record on any text entry technique is a typewriter. In 1714, Queen Anne granted a British patent to Henry Mill for a *Mechanical Transcribing Machine*. Since then typewriters evolved further, with numerous researchers and designers working on improvements in different parts of the world (Cooper, 1983). Typewriters gained worldwide acceptance in the mid-1880s, during the era of global industrialization. During that time several typewriters were commercially distributed, including the first commercially successful one—the E. Remington and Sons' Sholes-Glidden Type Writer. Most of these typewriters were very similar to the modern version and made an immediate impact on businesses, organizations, governments, and even on the social structure.

Traditionally, typewriters were used only for documenting and transcribing texts. But the development of personal computers in the 1970s added a new dimension to the task of text entry (Randell et al., 2003). Computers enabled users not only to input and transcribe texts but also to edit, format, and store them in electronic formats. Such electronic documents can be accessed anytime and be changed freely with much less effort compared to the earlier methods for changing content on paper. In addition, some current text editors permit even real-time collaborative text editing over the Internet.

The telephone was invented in the 1800s, around the same time as modern typewriters. Early telephones used manual switchboards and rotary dials. The standard 12-key phone keypad was introduced in the 1960s (Deininger, 1960) and provided for encoding of letters. Mobile phones appeared a few decades after that. At the beginning, mobile phones were limited to making calls, similar to land phones. The Short Message Service (SMS) was introduced in GSM phones in the early 1990s (Silfverberg, 2007). This changed the world of text entry once again, as it enabled users to exchange short text messages between mobile phones. In 2006, the GSM Association[1] estimated a worldwide total of one trillion text messages during the year of 2005, which exceeded the total number of voice calls.

[1] http://www.gsmworld.com

The concept of smartphones became popular in the 1990s. Smartphones are mobile phones built on mobile operating systems with more advanced computing capability and connectivity than standard "feature" phones. The first smartphone was developed by IBM, which they demonstrated in 1992 at the Computer Dealers' Exhibition (COMDEX). In 1994, BellSouth Cellular released a refined version of that prototype under the name *Simon Personal Communicator* (Sager, 2012). Currently, about half of all U.S. mobile phone users own a smartphone and about two thirds of the new buyers are opting for one (Nielsen, 2012). The introduction of smartphones expanded the horizon of mobile text entry by enabling users to not only exchange short text messages but also to take notes, send emails, and even to author full-length documents.

Nowadays, text entry is not limited to typewriters, computers, and mobile devices. We input text with our game controller or television remote control, and on a navigation system or automatic teller machine. We input text when we are at the office, at school, at home, and even when we are on the move. We input text to do our work, to obtain up-to-date information, for recreational purposes, and to keep up with our social life. Although the most popular text entry techniques are full-length and reduced-size physical and virtual keyboards, there are also gesture- and voice-based techniques, as well as techniques based on handwriting.

The task of text entry is complex, as it demands both cognitive and motor skills (Salthouse, 1986). It becomes even more complex with recent text entry techniques. Most handheld devices, for instance, use either reduced-size or virtual keyboards. The smaller key sizes of these keyboards make text entry more difficult and error prone compared to a standard (full-length) Qwerty keyboard (Drury and Hoffmann, 1992). On such keyboards, the whole fingertip often covers a key completely during text entry and may even extend well beyond it. This makes it harder to visually find and physically press the intended key, even when users are familiar with the keyboard layout. In a physical mobile keyboard, users can feel the keys under their fingers and experience an opposite force when pressing the keys. This feedback often helps experienced users to perform better. The absence of this feedback in virtual keyboards makes text entry with such keyboards even more challenging (Sears, 1991; Sears et al., 1993). Unsurprisingly, studies showed that *substitution errors*, where wrong keys are pressed in place of the intended ones, are the most common type of error committed with both physical and virtual mobile keyboards (Sad and Poirier, 2009; Sears et al., 1993).

Although handwriting and gesturing are considered as relatively natural and fluid modes of interaction, text entry with such techniques is slower and more error prone compared to standard and mobile Qwerty keyboards. This is mainly due to the lack of reliability in handwriting and gesture recognition technologies (Mankoff and Abowd, 1999; Zhai and Kristensson, 2003). Most handwriting and gesture recognizers

cannot differentiate reliably between similar characters or gesture pairs and also have difficulties with interpreting *illegible* handwriting and gestures, as do humans. The *segmentation* problem; i.e., the decision of which strokes should be grouped together, poses additional challenges, especially for characters or gestures that can be drawn in different ways. Several techniques attempt to avoid these issues by limiting the total number of possible gestures or by using a simplified set of gestures (Tappert and Cha, 2007). Even with these approaches, reliability in handwriting and gesture-based text entry techniques remains an issue (Mankoff and Abowd, 1999).

As text entry is becoming an integral part of our everyday life, users are naturally drawn to techniques that are not only fast but also accurate. Researchers are working on the development of new techniques and the optimization of the existing ones to address this need. As text entry requires a close cooperation between the users and the techniques, it is essential that new techniques be developed with due consideration of both human and system factors. System factors include the input hardware, good algorithms to interpret human input, how the system displays feedback, prediction methods and their accuracy, etc. Human factors include user comfort, limits on required motor and cognitive skills, training, human error rates, as well as any potential adaptation to the system itself. Even the psychology of text entry must be considered. All these factors need to be contemplated and integrated to provide users with a better text entry experience. While many of these factors have been studied extensively in the past, one noteworthy omission is the consideration of both human error behaviors as well as and system errors. The effect of such errors on text entry performance has not been very well investigated. This dissertation attempts to overcome this shortcoming through *experimental studies*, *modeling of relevant phenomena*, and the *development of new solutions*.

This work focuses only on character-based text entry techniques. Character-based techniques are those that involve text entry character by character, not by word or phrase. While the majority of text entry techniques are character-based, some techniques, especially techniques built on handwriting or speech recognition, are either word- or phrase-based. In general, these techniques do not permit text entry character by character or make it fairly laborious. However, many character-based techniques augment text entry with (prefix-based) word predictions and auto-correction. These techniques suggest the most probable word(s) based on what users are inputting and automatically correct *probably* misspelled words. Although this feature is most popular in mobile keyboards, many desktop applications, such as Microsoft Office and Apache OpenOffice, also use basic prefix-based word prediction and auto-correction. When a prediction is accepted, these techniques automatically input a chunk of text to complete the last unfinished word, enabling users to input more than a character with a single action such as a keystroke or a gesture. Such techniques are considered

to be character-based in this dissertation, as the functionality of these techniques is not dependent on the prediction and auto-correction features. Instead users are permitted to either ignore or disable said prediction without hindering the natural flow of text entry. Henceforth, the term *text entry* will refer to character-based text entry, unless stated otherwise.

1.1 Motivation

Almost all text entry user studies are conducted with one of three error correction conditions: *none*, *recommended*, or *forced*. In the *none* condition, participants are not allowed to correct any errors; in the *recommended* condition, correction of errors is recommended if and as participants identify them; and in the *forced* condition, participants are required to correct each and every error, which is usually enforced by the system. Section 2.4.1 elaborates on these conditions. This raises the question, if these different error correction conditions have a noticeable effect on popular text entry performance metrics. The answer to this question is vital for two reasons. First, it indicates if it is reasonable to compare results from different user studies that use different metrics and/or different error correction conditions. Second, it helps researchers to better understand current text entry metrics and also the relationships between them. Answers to these questions constitute a step towards making comparisons between different text entry user studies easier.

Most users use both character- and word-level correction strategies to correct text entry errors. In character-level correction an erroneous character is corrected right away, while in word-level correction an error is corrected after several other characters were inputted following the incorrect one(s). This latter strategy is used when experienced users chunk their input or when they do not verify their input right away. Section 2.6.5 explains these strategies. Almost all text entry techniques permit users to correct errors with both strategies (Grudin, 1983[a]). Thus, it is important to account for the effect of both strategies when developing models for predicting text entry error correction performance. However, currently there is no precise characterization of how frequently these two strategies occur in text entry.

Many models and modeling techniques have been proposed to predict the performance of text entry techniques. Several qualitative and quantitative models have also been proposed by psychologists to analyze the complex behavior of transcription typing. Section 2.6.8 provides a brief overview of these models. Yet error behavior in text entry is not very well understood. All existing models for predicting text entry performance account for errors in an indirect way. They either fail to account for both human- and system-specific factors or are not general enough to be used with different text entry techniques. Also, most of these models are relatively difficult to use, as they require significant amount of expertise, time, and

effort to be used in any concrete situation. Therefore, a straightforward and high-level model can be beneficial to researchers and practitioners as it can assist them to quickly pre-evaluate and fine-tune a technique before conducting empirical studies.

A phenomenon observed in many human-related domains, including user interfaces, is that most users adapt to a non-fatal system error if it remains in the system for long enough. Once users get accustomed to a system error, they either actively avoid replicating the sequence of actions that causes that error or start treating it as a feature[2]. This behavior can be indirectly explained through theories of learning. Some of these theories assume that learning is a process of replacement, where incorrect response tendencies are replaced with correct ones (Newell and Rosenbloom, 1981). Alternative theories describe learning as a process of accumulation, where incorrect response tendencies remain constant and correct response tendencies increase with practice (Mazur and Hastie, 1974). There are also theories involving the process of committing and correcting errors. Section 2.6 provides an overview of some of these theories. Regardless of the exact explanation, all learning theories suggest that it is vital to avoid both human and system mistakes in order to learn the correct responses. Human errors are well studied and explained in the field of text entry, error research, and cognitive psychology. However, *how* users deal with system errors has not been studied in depth. Based on the existing literature, one can hypothesize that the users' learning rate for system errors depends on how error prone a particular system is. Also, users should get used to avoiding an erroneous action faster, if it occurs more frequently than others. Unfortunately, no empirical studies have been conducted to investigate these likelihoods.

Almost all recent mobile touchscreen keyboards augment text entry with prefix-based word prediction and auto-correction. These methods suggest the most probable word(s) based on what users are inputting and automatically correct *likely* misspelled words. Most of these methods require users to tap on an area outside the virtual keyboard to *reject* or *bypass* an unintended suggestion. This requires additional mental preparation, visual scan time, as well as a finger movement to the target. Due to the small target sizes used, users may need several attempts to reject a prediction. This raises the possibility of accidentally selecting

---

[2] Long existing system errors are often jokingly referred to as undocumented features.

The Original Hacker's Dictionary: http://www.dourish.com/goodies/jargon.html

Gleick, J. (2002) Chasing Bugs in the Electronic Village. In What Just Happened: A Chronicle from the Information Frontier. Pantheon Books, New York, NY, USA, 15-26.

the wrong word. All these factors increase the overall cost of error correction with such techniques and increase also the total number of word-level error correction episodes. Section 2.2.4.3 elaborates on this issue. Eliminating the need for tapping on an area outside the keyboard should theoretically resolve at least some of these issues. However, no previous work has explored this possibility.

Most current touchscreen-based mobile devices do not provide hardware support for pressure detection. Several software solutions are available that simulate pressure detection on touchscreens (see Section 2.2.9). However, none of these solutions are broadly applicable, as they either increase the time to perform tasks that involve additional pressure or as they are user specific, mainly due to different finger sizes and touch behaviors. Thus, a new approach, which does not suffer from these shortcomings, could be beneficial as an alternative modality in text entry or other user interfaces.

1.2 Contributions

First, an empirical study was conducted to investigate the effect of different error correction conditions on popular text entry performance metrics. The study also explored the main strategies with which users correct their errors in short phrases. Results showed that the way errors are handled has a significant effect on all frequently used error metrics. Results also showed that about 50% of all errors are corrected at the character level; i.e., immediately, while the remaining 50% errors are corrected at the word level; i.e., after inputting one or more characters after the erroneous one(s). About 96% of all erroneous characters are noticed and fixed between the first and fourth character following the erroneous one.

Then, a new high-level and method-agnostic model was developed for predicting the cost of error correction with a given text entry technique. Towards this, users' error correction behaviors and strategies were first analyzed using data collected from the literature. The emphasis was on how users input text and correct errors. More specifically the focus was on the most frequently used error correction operations and the probability of making errors during the correction process. Then, based on the findings, a new model was developed that can predict the average *extra* time it requires per character to fix errors with a given character-based technique, regardless if a mistake was made on that character or not. The model encompasses not only the entry speed and error rate, but also any applicable error correction efforts. Thus, it provides designers with a better insight into a technique's performance and usability. Furthermore, it can also be used to pre-evaluate newly developed text entry and error prevention techniques by comparing those with the existing ones without performing user studies. The model was validated against quantities

derived from the literature and with a user study. Results of the user study showed that the predicted and observed costs of error correction correspond well.

To verify the hypotheses that users' gradually adapt to the *misrecognitions* of a faulty gesture recognition technique; i.e., a recognizer that frequently misrecognizes gestures, and that this adaptation rate is dependent on how frequently they occur, two separate user studies were conducted. In text entry, a *misrecognition* occurs when users input a gesture accurately, but the system fails to recognize it correctly and thus outputs a different letter. During the studies, a custom gesture recognition technique was used that was intentionally made more error prone. That is, it produced (synthetic) *misrecognitions* with controlled frequency. The intention was to observe if users start to use an alternative method to input the letters that are frequently misrecognized by the system. Assuming that there is an alternative gesture set, results of the studies confirmed that users gradually adapt to *misrecognition* errors by replacing error prone gestures with alternative ones. Also, users adapt to an error prone gesture faster if it occurs more frequently than others.

Then, a new hybrid approach was developed to simulate pressure detection on standard touchscreens. The new technique combines two existing touch-point- and time-based approaches. It uses the average time and touch-point movement for a specific task as baselines. Then, it simulates the detection of extra pressure when users take more time *and/or* their touch-point moves a larger distance compared to the baseline while performing that task. A user study investigating two pressure levels, *regular* and *extra*, showed that the hybrid technique simulates pressure detection more reliably. Results also indicated that users interpret the terms *regular* and *extra* pressure in a reasonably consistent manner. A separate user study investigated how much force is really applied for these two pressure levels. Results showed that *regular* pressure involves on average 1 N and *extra* pressure on average 3 N force on the surface.

Finally and to counteract some shortcomings of recent touchscreen virtual keyboards, a new pressure-based text entry technique was developed. The new technique does not require tapping outside the virtual keyboard to reject an incorrect or unwanted prediction. Instead, it only requires users to apply more pressure for the tap on the *next* target key, which may be any key. The performance of the new technique was compared with the conventional approach in a user study. Results showed that for inputting short English phrases with 10% non-dictionary words, text entry speed increased by 9% with the new technique and error rates decreases by 25%. Also, most users (83%) favored the new technique over the conventional one.

1.3 Brief Outline

This dissertation starts with an overall discussion on related work in Chapter 2. The primary focus is on the most important character-based text entry techniques, human and system errors, as well as error correction behaviors, performance metrics, and prediction models. Then, Chapter 3 investigates the effects of different error correction conditions on the most popular text entry performance metrics, outlines the approaches used by humans to correct errors, and identifies necessary parameters for the development of a high-level method-agnostic model for the cost of error correction. Based on these findings, Chapter 4 develops and validates a new model that can predict the costs of error correction for most character-based text entry techniques. Subsequently, Chapter 5 sheds light on how users adapt to (synthetic) misrecognitions in an error prone unistroke gesture recognizer. Chapter 6 presents and validates a new hybrid technique for simulating pressure detection on standard touchscreens. It also introduces and evaluates a new pressure-based predictive technique that enhances text entry performance by eliminating the need for tapping on an area outside the virtual keyboard. Finally, Chapter 7 concludes this dissertation and speculates on future research opportunities.

Chapter 2

Related Work

This chapter starts with a survey of the most popular commercial and academic character-based text entry techniques, errors and error correction behaviors, performance metrics, and prediction models. This review also identifies human- and system-specific factors related to text entry and error correction. Then, the chapter summarizes existing text entry performance data from user studies for the most important techniques to facilitate comparisons. This comparison provides a better understanding of where popular text entry techniques stand in terms of performance and offers a reference point for new work in the area.

2.1 Text Entry vs. Transcription Typing

Text entry research usually compares one text entry technique against another. It also provides guidelines, suggestions, and tools for the improvement of the existing or the development of a new technique(s). At first glance, it may seem more appropriate to permit participants to freely input whatever they desire during a study, as this replicates natural usage and improves the external validity of the study procedure. A few recent works attempted to collect data from natural usage (Evans and Wobbrock, 2012; Henze et al., 2012). The drawback of this approach is the absence of a source text to compare the transcribed text with to determine errors. Also, such data may be contaminated with spurious behavior, such as taking a break or performing a secondary task (MacKenzie and Soukoreff, 2003). Thus, the most common procedure is to present participants with predetermined short English phrases from a set, such as the one proposed by MacKenzie and Soukoreff (2003), one at a time to transcribe. These phrases are composed of on average 28.61 characters and do not contain numeric and special characters. The corpus also has a high correlation with the character frequency in the English language. The phrase set used in experiments is often a subset of the whole set. To imitate natural text input behavior as closely as possible, researchers usually instruct participants to first read, understand, and memorize the phrase, and then to input it. This approach greatly facilitates the measurement of error rates.

Although the text entry procedure discussed above is somewhat similar to transcription of short English phrases, transcription typing research attempts to analyze and understand the complex interaction of the perceptual, cognitive, and motoric processes involved in transcription typing. The intention is to contribute to the knowledge of the nature of skilled performance in a wide range of cognitive activities (Salthouse,

1986). In transcription typing research, participants are usually asked to transcribe a long piece of prose using a Qwerty typewriter or keyboard (Salthouse, 1986, 1987). They are not required to read, understand, or memorize the to-be-transcribed text prior to typing. Instead, they usually encode the text and translate that into a sequence of corresponding manual keystrokes concurrently (Rayner, 1998). Section 2.5 reviews transcription typing.

2.2 Text Entry Techniques

This section provides a brief overview of the most important text entry techniques, their effectiveness, and limitations. It also presents text entry performance data from the literature for those techniques.

2.2.1 Standard Qwerty Keyboard

The Qwerty layout has been dominant for both typewriters and computers since the late 1890s (Yamada, 1980). Currently, it is becoming the dominant layout for handheld devices as well (Arif, 2012). The name "Qwerty" comes from the sequence of the leftmost six keys in the layout's top row. It was designed by Christopher Sholes in the 1870s to overcome early typewriters' mechanical limitations. The problem with prior typewriters was that successively actuated levers would easily get jammed with one another. The Qwerty layout was arranged in such a way so that frequent bigrams in the English language were located far away from each other (Yamada, 1980). After the layout's invention almost all typewriter manufacturers adopted it. Later, the layout was slightly modified by swapping a few characters. It is unknown when exactly it took the form of today's standard Qwerty layout, but most reviews speculate that this must have happened well before the 1890s (Silfverberg, 2007; Yamada, 1980). When the layout was embraced in the first generation computer terminals, some extra keys were added to produce various computer-related operations, such as *escape,* or modifiers, such as *control*. The *function* and the *cursor arrow* keys were added later. In 1971, American Standards adopted the resulting layout as a standard (Noyes, 1983; Silfverberg, 2007). There are also several variations of the layout that provide the support for different languages. As Qwerty was designed for the English language, which does not have diacritical marks, almost all text entry systems were extended to enable users to input characters with accents, typically through changing modes. Text entry performance with Qwerty varies vastly based on user expertise. An earlier work claimed that first time Qwerty users could achieve up to 20 WPM after 12 hours of proper training (Noyes, 1983). Most professional typists can input text at 50-80 WPM, while some even achieve above 120 WPM (Ayres and Martinás, 2005). See Section 2.4.3, especially Table 1, for more data.

As the Qwerty layout was not designed with a deep consideration of human capabilities or limitations, it is often argued that it cannot be the best possible solution for desktop text entry (Silfverberg, 2007). Thus, many attempted to design more optimal layouts by rearranging the keys to optimize human hand or finger movements. Arguably, the most well-known redesign is the Dvorak Simplified Keyboard (DSK), designed by Dvorak and Dealey in 1936 (Dvorak et al., 1936). Yet user studies that compared the performance of Qwerty and Dvorak report contradictory results. In some studies, DSK yielded significantly better performance than Qwerty, while in many others no such significance was observed (Liebowitz and Margolis, 1996). However, most agree that with proper training it is possible to gain comparable entry speeds with these keyboards (Strong, 1956; Miller and Thomas Jr., 1977). DSK never achieved commercial success. Many attribute this to the theory of path dependence (Liebowitz and Margolis, 1990), which claims that the first product that attracts consumers will tend to have an advantage, even over superior alternatives that come along later. Others contribute this to a keyboard layout's steep learning curve (Carroll and Rosson, 1987), by referring to the theory of production paradox that states, *"The end users are willing to learn a new technology if and only if it is useful and lets them get their work done."*

2.2.2 Mini-Qwerty Keyboards

Mini-Qwerty keyboards, also known as Thumb and Two-thumb keyboards, are miniature versions of the standard Qwerty keyboard. They are typically employed on wireless handheld devices, such as smartphones. Mini-Qwerty keyboards were popularized by the RIM BlackBerry smartphone, where a mini-Qwerty keyboard was added in 1997 (CBC, 2013). Today, a large number of commercial handheld devices use mini-Qwerty keyboards (Arif, 2012). While most of these devices use a reduced-sized Qwerty keyboard with some augmentation, a few explore more creative approaches. The Nokia 6800, for instance, has a front face that can be flipped open to expose a split mini-Qwerty keyboard, with the screen at the center. Figure 1 showcases three commercially successful mini-Qwerty keyboards.

Figure 1. Three commercially successful Mini-Qwerty keyboards: (a) T-Mobile Sidekick 2 (b) Nokia 6800, and (c) BlackBerry Bold.

Previous work has established that keyboard size has a significant effect on entry speed for both novice and expert users (Sears et al., 1993). Thus, it is not surprising that mini-Qwerty keyboards yield comparatively lower entry speed than the standard Qwerty keyboard. A theoretical model predicted a peak expert text entry rate of 61 WPM with these keyboards (MacKenzie and Soukoreff, 2002[a]). Clarkson et al. (2005) verified this in a longitudinal user study, where expert users achieved on average 60 WPM entry speed and 6% error rate by the end of the twentieth session. Section 2.4.3 and particularly Table 1, provides more data on this.

2.2.3 Projection Qwerty Keyboard

A projection keyboard is a virtual keyboard that projects the image of a Qwerty keyboard (or any other user selected layout) on flat surfaces and permits users to input text by tapping on the projected keys. Most of these devices use sensor modules to pick up the finger movements over the virtual keys and translate those into standard keyboard input data (Roeber et al., 2003; Tomasi et al., 2003). The concept originated from IBM in 1995 (Korth, 1995). Their intention was to replace physical input devices by virtual ones so that the device can be optimized for the current application and the user's physiology, while maintaining the speed, simplicity, and unambiguity of manual data input. Several companies, such as Canesta, Celluons, Developer VKB, Elcom, and Virtual Devices, manufacture projection keyboards.

After the introduction of the first projection keyboard, many expected it to be quickly adopted by major handheld device manufacturers (Hesseldahl, 2002). Yet the technology has failed to achieve the anticipated traction in the market. This is may be due to business decisions, as projection keyboard take up valuable space in devices and add to production and maintenance expenditures, or usability issues. In a user study, a projection keyboard yielded on average 46.6 WPM entry speed and 3.7% error rate, which is substantially slower and more error prone compared to the results of its physical counterpart (Roeber et al., 2003). The most likely cause for this is the lack of tactile feedback and smaller key sizes (Lewis et al., 1997). Most projection keyboards use relatively smaller keys to permit users to use it even on small surfaces. Section 2.2.4.1 elaborates on the effects of tactile feedback on text entry.

2.2.4 Virtual (Qwerty) Keyboard

A virtual keyboard is a software component displayed on screen that enables users to input text on desktop and handheld devices, usually with a Qwerty layout. Although a virtual keyboard can be used in desktop environments with a mouse or similar devices, it is mostly used with touchscreens. Touchscreen Qwerty keyboards are becoming the dominant method for text entry on mobile devices (Arif, 2012; Nielsen, 2012).

Until recently, styli and digital pens had been the primary mode for touchscreen interactions. But today's preferred mode for interacting with touchscreens is the finger(s), as most manufactures avoided the inclusion of styli with a view towards user convenience and simplicity (Tu et al., 2012). Moreover, the majority of users also prefer using finger(s) over styli (Arif and Sylla, 2013). Therefore, this section emphasizes mostly finger- or touch-based interactions.

Virtual Qwerty keyboards yield better text entry performance than most other character-based layouts, most probably due to skill transfer from familiarity with the standard Qwerty keyboard (MacKenzie et al., 1999). The average entry speed and error rate reported for virtual Qwerty on touchscreen mobile phones are respectively 23.55 WPM and 12.13% for experienced users (Arif et al., 2011). See Section 2.4.3 and Table 1, for more data.

Yet, text entry with virtual keyboards is more difficult (Barrett, 1994; MacKenzie et al., 1999; Sears, 1991) and more error prone (Lewis et al., 1997), compared to physical keyboards. This is likely due to smaller key sizes and the absence of tactile feedback (Sears, 1991; Sears et al., 1993), similar to the decreased performance observed for projection keyboards. Prior studies on typewriters (Lessenberry, 1928) and Qwerty keyboards (Grudin, 1983[a]) revealed that substitution errors, where wrong characters are inputted in place of the correct ones, are the most frequent mistakes made by the users. These are usually caused by erroneous or unintentional keystrokes (Gong et al., 2005). Substitution errors are even more frequent in virtual Qwerty keyboards, where the key sizes are relatively smaller (Sad and Poirier, 2009). On such keyboards the whole fingertip often covers a key completely during text entry and may even extend well beyond it. This makes it harder to find and press the right key, even when the user is familiar with the layout. In a physical keyboard of the same size users can feel the keys under their fingers and experience an opposite force when pressing the keys. This feedback helps experienced users to perform better. Due to the absence of such tactile feedback in virtual keyboards, erroneous keystrokes are likely more frequent. The subsequent section provides further details regarding this.

Several academic and commercial alternatives are available to enable text entry on mobile devices. Most of these techniques utilize speech, handwriting, or gesture recognition and are not character-based. There are also a few hybrid techniques that combine Qwerty with gestures. Section 2.2.8.1 discusses a number of such techniques.

2.2.4.1 Keyboard and Key Sizes and Synthetic Feedbacks

Sears et al. (1993) conducted two user studies to investigate the effect of keyboard size on entry speed and error rate for touchscreen keyboards. They investigated four keyboard sizes from 6.8 to 24.6 cm wide. Results showed that entry speed ranged from 10 to 20 WPM for novice and 21 to 33 WPM for experienced users from the smallest to the biggest keyboards. This was found to be statistically significant. A significant effect of keyboard size on corrected error rate was also observed for novice users. However, no such effect was identified for experienced users.

Colle and Hiszem (2004) investigated the effect of different key sizes on entry speed and accuracy. They explored four key sizes: 10, 15, 20, and 25 mm square. Results showed that entry times were longer and error rates were higher for smaller keys. However, no significant difference was found between 20 and 25 mm keys. Recently, Parhi et al. (2006) conducted two user studies to investigate one thumb target pointing performance on mobile touchscreen devices. They first examined five key sizes from 3.8 to 11.5 mm and then examined five key sizes from 5.8 to 13.4 mm. Results showed that while speed generally improved with increasing key sizes, there were no significant differences in error rate between key sizes $\geq$ 9.6 mm in discrete tasks and targets $\geq$ 7.7 mm in serial tasks. Interestingly, similar investigations with styli have drawn different conclusions. A study that compared two virtual Qwerty keyboards, one with 6.4 mm and another with 10 mm wide keys, found no significant difference in text entry speed (MacKenzie and Zhang, 2001). A subsequent study verified and extended this work for keys from 2.6 to 4.4 mm (Sears and Zha, 2003). Then again, a study that compared key sizes from 2 to 5 mm square found that speed and error rate improved with increasing key sizes (Mizobuchi et al., 2002).

Many researchers also explored whether providing users with auditory and/or synthetic tactile feedback can improve touchscreen interaction performance. Lee and Zhai (2009) compared different touchscreen virtual keypads with physical ones and showed that for smaller tasks, such as dialing a phone number, virtual keypads augmented with synthetic tactile feedback (vibration) can offer a level of performance similar to physical keyboards. Hoggan et al. (2008) also used vibration with virtual keyboard to replicate tactile feedback. That addition raised text entry performance almost to the level achievable with physical keyboards. Kaaresoja et al. (2006) tested a touchscreen device augmented with synthetic tactile feedback in four tasks: numeric character input, text selection, scrolling, and drag and drop. Based on their observations, they speculated that the addition of tactile feedback has the potential to improve both usability and the user experience of such devices. Arif et al. (2010) compared text entry with a virtual Qwerty with and without synthetic tactile feedback (vibration). Results showed that augmenting synthetic tactile feedback improves

both entry speed and accuracy for novice users. Kaaresoja and Linjama (2005) conducted an interesting study on the perception of temporal characteristics of vibration of a mobile device by placing a mobile device at different body sites, such as hand, trouser front pocket, and belt case. They argued that the duration of the vibration control signal should be between 50 and 200 ms. Following up on this, Koskinen et al. (2008) conducted a series of studies with a piezo actuator and a vibration motor to find a tactile click that is the most pleasant to use with a finger. Their results were consistent with earlier findings that tactile feedback is superior to a non-tactile condition. They also found that the perceived pleasantness depends on the characteristics of the tactile feedback parameters that define the wave shape of the stimuli. Their results showed that a 46 mA drive current for the piezo actuator and a 16 ms drive time for the vibration motor created the most pleasant tactile feedback. In addition, Bender (1999) found that auditory feedback improves speed and accuracy in some cases. In a different study Brewster (2002) investigated data entry with different key sizes with a range of different types of auditory feedback. Results showed that key click sounds significantly improved usability for all key sizes.

2.2.4.2 Error Prevention Techniques

To reduce the effect of substitution errors several error prevention techniques have been developed. The default iOS virtual keyboard uses an error prevention technique, called key-target resizing (Pogue, 2007). In this approach and instead of the visual representation of the keys, the (invisible) underlying target areas are dynamically resized based on the occurrence probability associated with each character. Similar algorithms exist in the research literature (Gunawardana et al., 2010). Clawson et al. (2008) proposed a different approach. They developed an error prevention technique for mini-Qwerty keyboards where they used a trigram frequency table along with the proximity information of the keys and the time between the previous and the current keystroke to predict unlikely characters. When the user inputs such an unlikely character, they then replace it with a more likely alternative. Their approach makes inputting characters flagged as “unlikely” almost impossible. Arif et al. (2010), in contrast, proposed a timeout- and a pressure-based error prevention technique. Using a bigram character frequency table, this approach predicts improbable next characters based on the previously entered ones, and then make those characters harder to input. In order to input an improbable character, one then has to either tap-hold the corresponding key for a predetermined period with for the timeout approach, or has to tap on that key with extra pressure with the pressure-based approach. Results of a pilot study indicated that the new techniques reduce errors significantly for novice users when augmented with synthetic tactile feedback (vibration). However, a subsequent study failed to observer such effects for expert users (Arif and Stuerzlinger, 2013).

2.2.4.3 Predictive Techniques

Nowadays, almost all virtual keyboards augment text entry with prefix-based word prediction and auto-correction. These approaches suggest the most probable complete word(s) based on what users are typing and even automatically correct a *likely* misspelled word. Figure 2 (a) shows word prediction on the iPhone keyboard, where the most probable word completion “education” is suggested based on the input (or prefix) in a prediction bubble. When a word is suggested, one can perform any of the following operations.

1. *Accept* the prediction by tapping on the Space key. This will replace the partially inputted word with the suggested one, followed by a Space character.
2. *Reject* or bypass prediction for that word by tapping on the prediction bubble. This will remove the prediction bubble along with the predicted word.
3. *Ignore* the prediction and continue typing. Here, the system will keep updating the suggestion based on the prefix. For instance, if the user has input “edu” and continues typing even though a prediction is shown and reaches “educab”, the system will update the suggestion to “educable”, which is the most probable word that starts with that prefix. When the system fails to find a match based on the prefix, it often assumes that a spelling mistake has been made. It then suggests the *closest* most likely word. For instance, if one inputs “educc”, the system will assume that the user made a spelling mistake and thus will continue suggesting the word “education”.

Some virtual keyboards suggest more than one word. The default Android keyboard, for example, suggests the two most probable words in a prediction panel placed above the keyboard. With this approach users can again perform any of the above-mentioned actions. See Figure 2 (b). Here, the system highlights the word “education” to signify that this word will be used for auto-completion when the Space key is pressed. To reject or bypass this suggestion, one has to tap either on the typed text (in the left of the panel) or the second most probable word (elsewhere in the panel). Thus, both Apple and Android platforms require users to tap in an area away from the virtual keyboard to reject or bypass an incorrect prediction. Once a prediction is rejected, both keyboards will suggest completions again only after the user inputs a Space character or tap on the Return or Backspace keys.

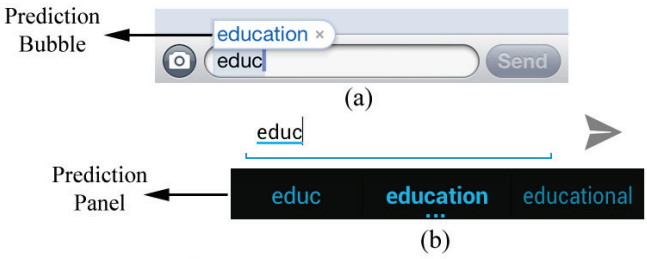

Figure 2. Default word prediction systems on: (a) Apple iOS and (b) Android OS.

Although word prediction is almost exclusive to mobile touchscreen devices, several desktop applications provide limited prefix-based word prediction and auto-correction. Microsoft Office and Apache OpenOffice, for example, suggest the most common nouns (month and day names) and autocorrect common substitution errors such as “teh” to “the”. Figure 3 illustrates word prediction on these two applications.

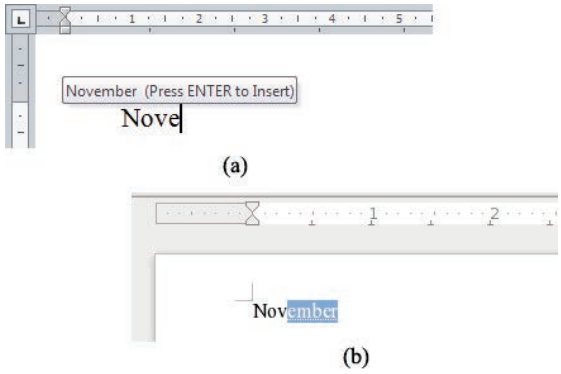

**Figure 3. Default word prediction on two desktop applications:
(a) Microsoft Office and (b) Apache OpenOffice.**

Many schemes have been proposed for accurate word prediction, such as bigram, trigram, or n-gram letter and word frequencies, grammar rules, adaptive dictionaries, geometric pattern matching, word classification, and language models. Garay-Vitoria and Abascal (2006) offer an inclusive survey of the major schemes used in text prediction. Prediction schemes are however outside the scope of this work as the primary focus of this work is on error behaviors and not on prediction algorithms.

2.2.5 Standard 12-Key Mobile Keypad

Although gradually becoming obsolete in more developed countries (Arif, 2012; Nielsen, 2012), text entry in handheld devices worldwide is still largely dependent on the standard 12-key mobile keypad (Gupta et al., 2013), especially in the developing world. The standard mobile keypad consists of a 3×4 grid with ten

numeric (0-9) keys and an asterisk (*) and a hash (#) key. See Figure 4. Typically, the letters of the English language are mapped to (eight of) the numeric keys. The asterisk and hash keys are usually used for special functions. This layout was standardized in the 1990s (Silfverberg, 2007). Text entry with the standard mobile keypad is challenging due to the underlying key ambiguity, as more than one character is assigned to each key. This problem is universal as most languages have more than twelve characters. To overcome this, text entry with mobile keypads requires the use of special techniques such as Multi-tap and T9.

Multi-tap is the one of the dominant techniques used with the standard 12-key keypad (James and Reischel, 2001). To input text with Multi-tap, users have to press a key repeatedly until they get the intended character. Then, they can proceed to the next character, providing that it is on a different key. If not, they have to either wait for a timeout period for the system to accept a character on the same key or have to press a predetermined *kill* button, usually the hash (#) key. Multi-tap was preceded by a similar technique introduced by Casio Computer Company in their Databank wristwatches[3] in the 1980s. Casio Databank keypads were laid out in a 4×4 grid, where thirteen keys contain two English letters each and the remaining three are for special functions. Similar to Multi-tap, Casio Databank users had to press the keys once or several times to get the intended character.

Text entry with Multi-tap is usually slower and more error prone compared to mini- and virtual Qwerty keyboards. An empirical study reported an average 8 WPM entry speed and 28.64% error rate for Multi-tap for experienced users (Lyons et al., 2004[b]). See Section 2.4.3, especially Table 1, for more information.

Figure 4. A standard 12-key mobile keypad.

T9 is the most popular predictive technique for the standard mobile keypad. It was primarily developed by Tegic, now acquired by Nuance Communications, and later was licensed to several major mobile phone manufacturers (Dunlop and Crossan, 2000). T9 addresses the issue of key ambiguity by predicting probable

[3] http://www.casio.com/products/watches/databank

input words based on dictionary and grammar rules stored in the device. The idea is to reduce keystrokes by allowing users to press a single key per character instead of repeated keystrokes. T9 assumes that for most key sequences there is only one or a limited number of words that match the exact sequence. Thus, upon a keystroke, the technique attempts to match the key sequence with the phone's dictionary for the probable input word. When multiple words match the input sequence, it also uses statistical data on word frequency to suggest the most common word first. Thus, T9 can often suggest accurately what the user is attempting to input. However, if an intended word is not in the dictionary, users have to input the word by resorting to Multi-tap in a different dialogue. T9 is usually faster than Multi-tap. A user study reported 20 WPM entry speed and 8.4% error rate for experienced users (James and Reischel, 2001). See Section 2.4.3, especially Table 1, for more information.

Many alternative methods have been proposed to address the standard mobile keypad's key ambiguity, mostly from academic circles. TiltText, for example, attempts to resolve key ambiguity using the orientation of the phone (Wigdor and Balakrishnan, 2003). It requires users to tilt the phone in one of four possible directions to select a character on a specific key. A user study revealed that with sufficient practice this method yields 23% faster entry speed than Multi-tap (Wigdor and Balakrishnan, 2003). It was however 73% more faulty than Multi-tap. Another approach, known as Two-key, uses multiple modes for different character groups (Butts and Cockburn, 2002). To input text with this technique, users first have to select a character group by pressing a key, and then have to press 1, 2, 3, or 4 to input the intended character within that group. A study reported an average 5.5 WPM entry speed for novice users for Two-key. LetterWise is a predictive technique that predicts characters instead of words—it predicts the most probable next character based on the previous character using bigram and trigram tables (MacKenzie et al., 2001). It rearranges the character sequence on each keys dynamically based on the prediction. A user study reported that with sufficient practice this method could yield up to 36% faster entry speed compared to Multi-tap, without compromising accuracy (MacKenzie et al., 2001). Tanaka-Ishii et al. (2003) and Trnka et al. (2009) provide excellent reviews of other, less popular predictive text entry techniques.

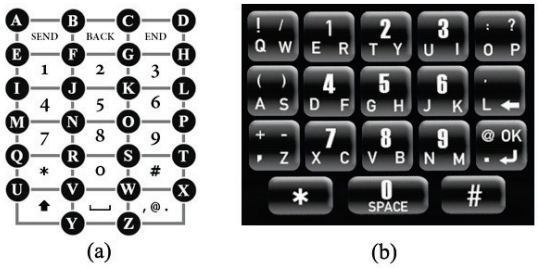

Figure 5. Two commercial alternates to the standard 12-key keypad: (a) Fastap or OneTouch and (b) A variant of reduced-Qwerty.

Many also attempted to redesign the mobile keypad layout. Pavlovych and Stuerzlinger (2003) designed a new layout called Less-tap that rearranges the characters within each button according to their frequency so that the most common characters require only a single keystroke. They conducted a user study to compare Less-Tap to Multi-tap, where the new technique exhibited 9.5% faster entry speed than Multi-tap. A commercial technique, called Fastap, formally known as OneTouch, adds two additional function keys and places all the letters of the English language in alphabetic order intertwined between the keys (Levy, 2002). See Figure 5 (a). This layout contains more than forty keys in a small space, which makes it harder to master. However, many claim that with proper training it can be faster than Multi-tap (Sirisena, 2002). Another commercial layout, called reduced-Qwerty, overlaps a Qwerty layout with the standard mobile keypad. In this layout, characters are ordered in a typical Qwerty fashion but the keys contain one or more characters. There are several variations of this layout available in the market. Figure 5 (b) illustrates one such layout.

2.2.6 Chorded Keyboards and Keyers

Chorded keyboards and keyers are designed to accommodate the idea of wearable computers. Wearable computers provide computational support when users' hands, voice, eyes, arms, and/or attention are actively engaged with the physical environment. In this environment, such technologies enable users to input text with only one hand, leaving the other hand free to do other tasks. This is usually accomplished via key combinations on smaller keysets. Therefore, to input text with a chorded technique, users have to press multiple keys simultaneously. This is called a *chord*. The difference between a chorded keyboard and a keyer is that keyboards use physical boards to arrange the keys, typically in rectangular layouts, while keyers arrange the keys in clusters adapted to the human hand. The most popular chorded keyboard is the HandyKey Twiddler (Lyons et al., 2004[a], 2006). Twiddler has twelve keys, similar to the standard mobile keypad, arranged in a grid with three columns and four rows on the front. Each row is usually operated by one of the four fingers of a hand. Disregarding the cords in which no buttons are pressed there are in total 255 possible combinations or chords (Lyons et al., 2004[a]). Twiddler fits almost completely inside one hand, which allows users to input text almost invisibly (Lyons et al., 2004[b]). Twiddler is not fully character-based as it provide shortcut chords for frequently used words, such as "or" and "the", and common word endings, such as "ing" or "ed".

Text entry with chording techniques is difficult from both a cognitive and a physiological standpoint. It also takes significant training time to master the chords. Thus, it was never widely adopted. A user study reported 46.3 WPM entry speed and 7% error rate for experienced Twiddler users (Lyons, 2004[a]). See Section 2.4.3 and Table 1, for more data.

Several additional chorded techniques have been proposed. Gopher et al. (1985, 1988) designed a two-handed chorded keyboard that connected two separate keyboards in a lateral tilt arrangement, which were mirror images of one another. Wigdor and Balakrishnan (2004) developed a chorded version of Two-tap (see Section 2.2.5), called ChordTap, by adding three additional keys on the back of a mobile phone. With this technique, users have to make a between-group selection using the standard mobile keypad with their dominant hand and then a within-group selection using the chorded keys with their non-dominant hand. Several other chording keyboards are (or were) available in the market such as the 7-key TextWriter, the 7-key Infogrip BAT, the 6-key Microwriter, the 12-key WriteHander, the 6-key GKOS, the 20-key FrogPad, the 7-8-key Chordite, the 12-key EkaPad, and the 10-key IN10DID. Amongst the keyers, the most popular one is called the Septambic Keyer[4]. It is made of three thumb and four finger keys grouped in a cluster for being handheld. It allows for in total 47 distinct combinations of keystrokes and chords.

2.2.7 Handwriting Recognition (HWR)

Handwriting is considered a relatively natural and fluid mode of text entry with a long history. Moreover, handwriting is learned in early school years (Plamondon and Srihari, 2000). Recently, there has been an increase in handwriting-based text entry on handheld devices. However, handwriting recognition of standard characters is relatively slower and more error prone than text entry with Qwerty (Zhai and Kristensson, 2003). This section briefly discusses online or real-time handwriting recognition, which is one of the dominant text entry methods on smartphones (Tappert and Cha, 2007). This section excludes offline handwriting recognition methods that deal with static information, as such methods fall into the area of Optical Character Recognition (OCR).

Online handwriting recognition techniques recognize the writing as users write the text. Unlike offline techniques, online techniques capture dynamic information during writing, which includes the *number* and the *order* of the strokes, the *direction* of the writing of each stroke, and the *speed* of writing within each stroke. This enables the recognizer not only to recognize characters more accurately but also to differentiate between similarly shaped characters and numbers. To record such dynamic information, special equipment is required to record the writing process. For instance, a tablet digitizer accurately records the *x*-*y* coordinate data of stylus movement for each point in time. Recent devices that support pen-based interactions, such as

[4] http://wearcam.org/septambic

the HP Compaq Tablet PC and the Microsoft Surface, attempt to imitate paper-like interaction by combining a digitizer and flat display. This permits use of the same surface to both record the input and to show the output, typically as printed characters, which provides immediate visual feedback. Many devices also support multi-touch interaction. Some provide the hardware support for detecting stylus pressure, such as the Bamboo Pen & Touch Graphic Tablets.

The first commercial online handwriting recognition technique was sold with the Apple Newton device in 1993 (Silfverberg, 2007). The technique was considerably advanced as it accepted whole words, permitted cursive writing, and could recognize common shapes and symbols. Besides, the system was adaptive. That is, it learned a user's writing styles. However, the recognition accuracy was not very good (Silfverberg, 2007). Although a second version was limited to recognition of printed text to improve accuracy, the method was still not reliable enough. This has likely contributed to the commercial failure of the Apple Newton as well as the competitors' solutions.

Figure 6. Different characters with the same shape.

One fundamental property of handwriting is that differences between different characters are more significant than differences between different writing of the same character (Tappert et al., 1990). Most handwriting recognition techniques attempt to make the most of this. However, and especially in English handwriting, this property holds within the subcategories of uppercase, lowercase, and numeric characters, but not across them. Figure 6 illustrates how different handwritten characters can have the same shape, where the digit "1", uppercase "I", and the lowercase "L" are drawn the same way, also the lowercase "O" and the digit "0", if size is disregarded. Also, there are different kinds of handwriting, such as hand-printed discrete characters used in boxes while filling out forms, spaced discrete characters, run-on discrete where characters can touch and overlap, pure cursive writing, and a mixture of discrete and cursive writing (Tappert et al., 1990; Tappert and Cha, 2007). Figure 7 illustrates this. Most real-time techniques are quite accurate with the first three kinds of handwriting. Yet recognition the accuracy for the latter two kinds highly depends on the writing style and the regularity and clarity of the writing (Tappert and Cha, 2007). Especially in *illegible* handwriting, it is difficult to distinguish between similar character pairs, not only for recognition systems but also at times for humans. Additionally and as many characters can be drawn with a single or multiple strokes, it is often hard for a recognizer to decide which strokes should be grouped together. This is known as the *segmentation problem*. It is also hard to separate a *character-within-character*. For example, an uppercase "B" drawn with two strokes can be mistakenly recognized as number

"13". In addition, different users can draw the same character in different stroke *numbers*, *directions*, and *orders*. The total number of possible variations for a multistroke character can be calculated using the following equation.

$$n! * 2^n \qquad \text{Equation (1)}$$

Here, *n* is the total number of strokes. Using this equation, a two-stroke character such as "T" has in total 8 writing variations: 2! = 2 for different stroke *orders* multiplied by $2^2 = 4$ for two possible stroke *directions* for each stroke. This, combined with the *segmentation* and *character-within-character* problems, poses a big challenge to online handwriting recognition.

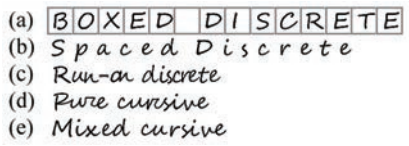

Figure 7. Different kinds of English handwriting: (a) Hand-printed discrete characters, (b) Spaced discrete characters, (c) Run-on discrete characters, (d) Pure cursive, and (e) A mixture of discrete and cursive writing.

Many approaches have been used to overcome these issues. Some techniques combine preprocessing with writer control, where users first have to draw all characters in predefined boxes to allow the recognizer to group strokes within a box. Then that information is used to detect the completion of character input (Tappert and Cha, 2007). The stroke code method, in contrast, takes a more drastic measure by recognizing each stroke immediately after the stylus is lifted (Tappert et al., 1990), making it very similar to unistroke gesture recognizers. Some techniques permit only a single way of drawing each character in an attempt to avoid stroke variations (Plamondon and Srihari, 2000). Others permit multiple variations, but attempt to train users to the recognizer by providing them constant visual feedback through a trial and error method (Plamondon and Srihari, 2000). Some recognizers disregard stroke sequences and directions by reordering the strokes and stroke directions into a normalized form such as from *left to right* and *top to bottom* (Tappert et al., 1990). Above all, most recent handwriting recognition techniques attempt to understand the context and meaning of the text by using the syntax and semantics of the language, as humans do (Tappert and Cha, 2007). Hidden Markov models (Hu et al., 1996), support vector machines (Myers, 1980), machine learning (Lee et al., 2012), parallelized machine learning (Bothe et al., 2010), neural network (Liwicki et al., 2012; Pittman, 1991), recurrent neural network (Graves et al., 2009), and many other approaches have been used with language models for this purpose. Consequently, most recent handwriting recognition techniques work at the word- or phrase-level, which is outside the scope of this work.

2.2.8 Gesture Recognition

Gesture-based text entry techniques have been widely explored and aim to increase the speed and accuracy over free-form handwriting methods (Tappert and Cha, 2007). However, gesture recognition is different from handwriting recognition in many ways. Handwriting recognition techniques attempt to support natural handwriting, while gesture recognition techniques avoid such natural usage to improve recognition accuracy and entry speed. Thus, almost all gesture-based techniques limit user behaviors by allowing only a single way of drawing each character to avoid *segmentation* and other handwriting recognition related problems (Buxton, 1995).

Many gesture-based techniques use simplified sets of characters (often called shorthand) that are drawn with a single stroke. Although recognition of traditional shorthand has also been investigated by Leedham et al. (1984), it is technically difficult. Moreover, there are not many people who have the skill of writing shorthand, and the skill is not easy to acquire (Buxton, 1995). Thus, designers have proposed alternative shorthand notations that do not suffer from these disadvantages.

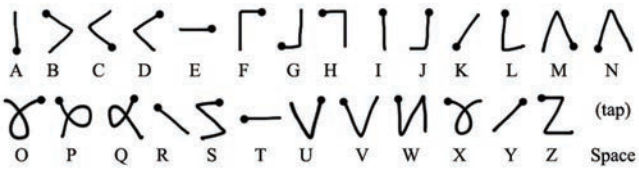

Figure 8. Unistrokes gesture alphabet. Here, a dot represents the start point of a stroke.

Goldberg and Richardson (1993) developed one such technique, Unistrokes. Their intention was to design a character set that could be entered in an eyes-free manner on portable systems with a stylus. The Unistrokes alphabet is shown in Figure 8. As the name suggests, each character is represented by a single stroke mark. Although it is necessary for the users to master the gestures, this usually takes about an hour due to the clever use of mnemonic structures (Buxton, 1995). These are illustrated in Figure 9.

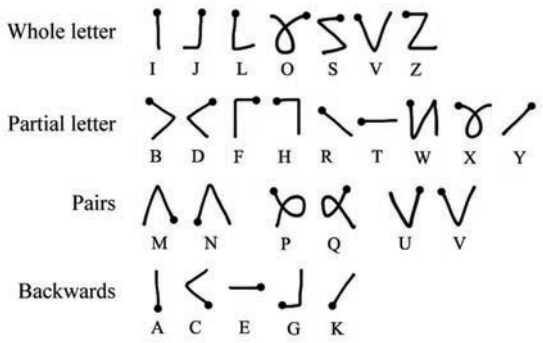

Figure 9. Unistroke mnemonics. Here, a dot represents the start point of a stroke.

To reduce the learning effort, Palm Computing introduced the Graffiti character set in 1997 (Isokoski, 1999). Similar to Unistrokes, Graffiti characters are entered with single strokes. Yet Graffiti is relatively easier to learn, as the strokes are closer to their printed counterparts. Graffiti is illustrated in Figure 10 (a). Due to a legal issue, Palm later replaced the original Graffiti characters with Graffiti 2. It is claimed that the latter is an improved version as it offers more intuitive entry of accents and umlauts and also more consistent entry of special characters (Költringer and Grechenig, 2004). This version requires some characters such as "I", "K", "T", and "X" to be drawn with two strokes, illustrated in Figure 10 (b).

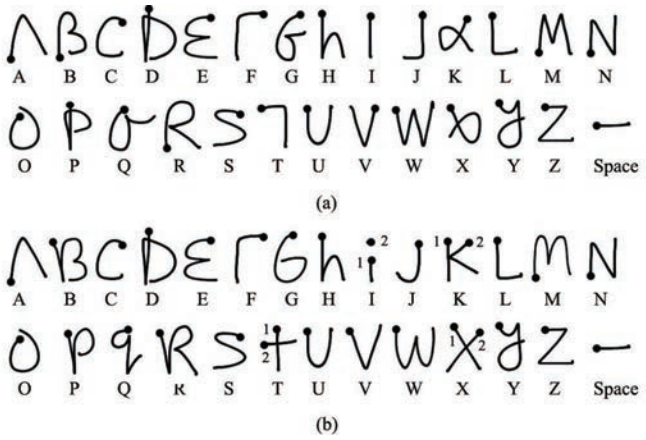

Figure 10. (a) Graffiti and (b) Graffiti 2 unistroke gesture alphabet.
Here, a dot represents the start point of a stroke and the numbers represent stroke sequences.

Castellucci and MacKenzie (2008) conducted a longitudinal user study to compare Unistrokes and Graffiti. Interestingly, they did not find any significant difference between these techniques' entry speed, correction rate, and preparation time. Average entry speed and accuracy with Unistrokes were 15.8 WPM and 16%, while with Graffiti were 11.4 WPM and 26%. Unistrokes were executed faster than Graffiti due to their

short and simple strokes, which complements a prior study conducted by Cao and Zhai (2007). See Section 2.4.3 and Table 1, for more information.

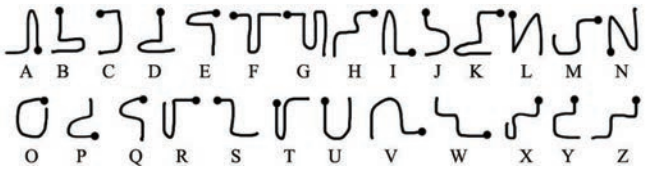

Figure 11. The Minimal Device-independent Text Input Method (MDTIM) unistroke gesture alphabet. Here, a dot represents the start point of a stroke.

Isokoski (1999) observed that four directional gestures, namely up, down, left, and right, are easy to make with most pointing devices. Based on this observation, he developed a new unistroke gesture set, called the Minimal Device-independent Text Input Method (MDTIM). The set optimizes the mapping between the gestures and the letters of the English language. Thus, more frequent characters have shorter strokes. Figure 11 illustrates this set. Isokoski (1999) evaluated MDTIM with a number of pointing devices and reported 7.5 WPM entry speed for novice users with touchpads. This method shares a drawback with Unistrokes, in that gestures are different from their printed counterparts. Therefore, it requires practice to learn the gestures and to achieve fast entry speed (MacKenzie and Soukoreff, 2002[b]).

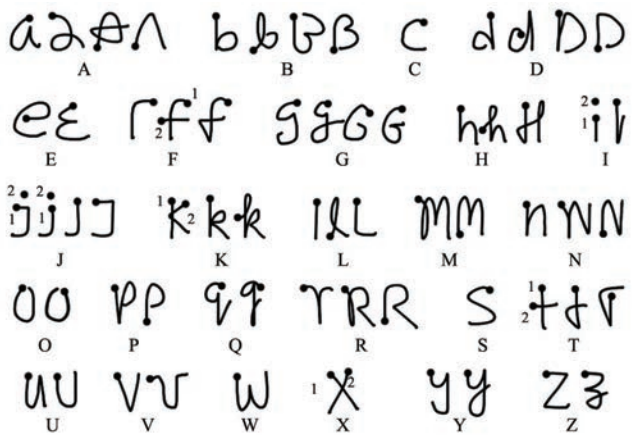

Figure 12. Jot gesture alphabet. Here, a dot represents the start point of a stroke and the numbers represent stroke sequences.

Jot System, a relatively recent technique, was developed by Communication Intelligence Corporation and licensed by Microsoft in 1998. This technique is very similar to Graffiti 2 as it includes almost all Graffiti 2 gestures. Besides, it includes several variants of the gestures to accommodate handwriting-like drawing.

See Figure 12. Jot also allows users to indicate drawing preferences for some characters. That is, it permits users to select alternative methods for drawing some characters (MacKenzie and Soukoreff, 2002[b]).

Based on Kurtenbach and Buxton's (1993) marking menu, Venolia and Neiberg (1994) developed a new gesture-based text entry technique, called T-Cube. T-Cube is similar to a two-tier pie menu system. The main level contains nine starting points and the second level contains eight pie menus, each representing a particular character. To input a given character, users first have to select an entry in the main menu to activate the second level menu that contains the intended character. Then, they have to flick the stylus into the direction where the intended character is situated in said second level menu. See Figure 13. T-Cube displays the menus only when users hesitate, in an attempt to reduce visual scan time. In a pilot longitudinal study, the technique yielded a maximum entry speed of 21.2 WPM. Interestingly, a linear increase in entry speed over time was observed during the pilot, which showed no indication of leveling off. Based on this Venolia and Neiberg (1994) speculated that although difficult to learn, reasonably fast entry speeds can be achieved with T-Cube with sufficient practice.

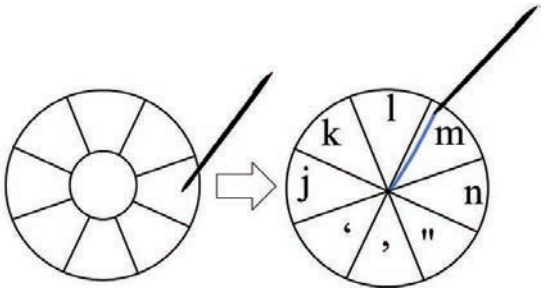

Figure 13. T-Cube pie menu structure. Here, first the user selects an entry from the main level menu. In the second level menu he/she then flicks the stylus into the direction of the intended character, here "m".

Wobbrock et al. (2003) designed a unistroke technique called EdgeWrite for users with motor impairments. The EdgeWrite alphabet was designed to maximize users' ability to guess, illustrated in Figure 14. Unlike other gesture-based techniques, it requires users to input characters by traversing the edges and diagonals of a square hole overlaid over the character drawing area of a PDA. Figure 15 illustrates the character and digit drawing areas of a PDA. Then a gesture is recognized not through patterns, but based on the sequence of corners that are hit. This technique has been explored widely with different devices such PDA, touchpad, displacement and isometric joysticks, trackball, and a 4-key keypad (Wobbrock and Myers, 2005). A study established the technique to be more accessible to users with motor impairments. Individuals who cannot input text with Graffiti or similar techniques can do the same with EdgeWrite. Also, the stylus version has been shown to be significantly more accurate than Graffiti, for both able-bodied and motor-impaired users (Wobbrock et al., 2003; Wobbrock, 2006).

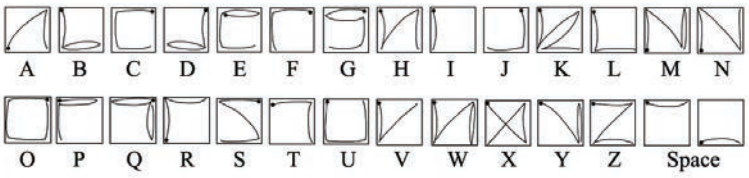

Figure 14. EdgeWrite unistroke gesture alphabet. Here, a dot represents the start point of a stroke.

MacKenzie et al. (2006) developed a technique called Unipad, which augments Unistrokes with word and suffix completion and word prediction. After two hours of practice the technique exhibited 11.6 WPM and 0.90% error rate in a user study. A recent technique, called UniGest, allows users to input text with pointing devices without a display (Castellucci and MacKenzie, 2008). Based on a web-based study, an upper-bound text entry rate of 27.9 WPM was predicted for the technique. Another technique, called Ubi-Finger, provided a wearable interface for sensory control of mobile computers with finger gestures (Tsukada and Yasumura, 2002).

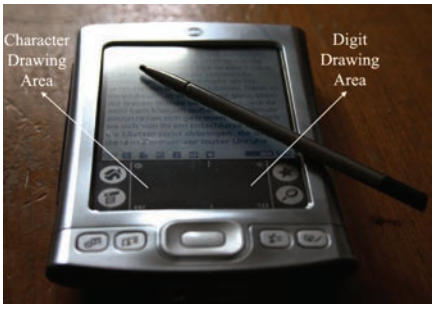

Figure 15. A Palm Tungsten E PDA that allows users to input text using Graffiti 2.

Choi et al. (2005) proposed a number of gesture-based interaction methods using a tri-axis accelerometer for handheld devices. They tested their approach with a mobile phone that achieved an average recognition rate of 97% for a set of eleven gestures. Kallio et al. (2003) developed an accelerometer-based gesture recognition system for a small wireless sensor-box. They evaluated the new technique with gestures of four degrees of complexity. Results showed that with at least ten training vectors, the accuracy rate for complex gestures could reach up to 95%. Patel et al. (2004) designed a sensor-based authentication mechanism for mobile devices, which uses simple shaking to authenticate with the public infrastructure. The technique has not been evaluated in a user study.

Gesture-based text entry received a lot of attention during the late 1990s. Nowadays, techniques such as virtual and mini-Qwerty have become the dominant method for text entry on mobile devices, also because most users are familiar with the layout (Arif, 2012; Tappert and Cha, 2007). Another potential reason for

the failure of gesture-based techniques to achieve widespread usage is their lower accuracy rate. Empirical comparisons between Graffiti 2 and a virtual Qwerty keyboard showed that text entry with Graffiti 2 is significantly slower and more error prone, even when augmented with word prediction (Költringer and Grechenig, 2004). A high accuracy rate is imperative for acceptance as a study showed that a gesture recognition technique has to be at least 97% accurate for its users to find it useful (LaLomia, 1994). Another study showed that mobile users abandon a gesture-based technique and start using an alternate interaction mode when error rates reach about 40% (Karam and schraefel, 2006).

Although gesture recognition is relatively easy compared to online handwriting recognition, most gesture recognition techniques still suffer from recognition errors (Mankoff and Abowd, 1999; Shilman et al., 2006). Yet recent advancements in pen, finger, and wand gestures with user interfaces for mobile, tablet, large display, tabletop, televisions, interaction without display, and desktop computers[5] (Cao and Balakrishnan, 2003; Guimbretière et al., 2001; Gustafson et al., 2010; Hinckley et al., 2004; Juhlin and Önnevall, 2013; Karlson et al., 2005; Wilson and Shafer, 2003) have increased the overall gesture ambiguity. Gesture recognition has been a topic of interest to experts in artificial intelligence and pattern matching (Shaffer, 1975[a]; Rubine, 1991). The original Unistrokes gesture recognizer recognized a performed gesture by comparing it with a list of ordered x-y coordinates for each gesture (Goldberg, 1997). This method is dependent on the relatively wide separation of the Unistrokes gestures to differentiate between the gestures. The mechanism of the original Graffiti gesture recognizer has not been publicly disclosed due to its proprietary nature. However, some speculate that it is very similar to the original Unistrokes recognizer (Hertzberg, 2011).

So far, at least Hidden Markov Models (Anderson et al., 2004; Cao and Balakrishnan, 2005; Sezgin and Davis, 2005), neural networks (Pittman, 1991), dynamic programming (Myers, 1980), and machine learning (Lee et al., 2012), have been tried to enhance the performance of gesture recognizers. No fundamentally superior approaches have been identified. Others have proposed easy and efficient implementation methods for gesture recognition. These techniques recognize gestures through templates based on basic geometry and/or trigonometry and usually do not require feature selection or training examples (Li, 2010[b]; Wobbrock et al., 2007). Various error prevention and error correction methods have also been proposed (Mankoff and Abowd, 1999). As the primary focus of this work is on interaction and

[5] https://www.leapmotion.com

not on algorithms, this document does not discuss gesture recognition and gesture error prevention algorithms in more detail. Nonetheless, the most frequent types of human and system errors and the most frequently used methods for handling such errors are discussed in Section 2.6.7.

2.2.8.1 Tap and Gesture Hybrids

In a 1995 patent Buxton and Kurtenbach (1995) proposed a hybrid of tapping and linear stroke gestures for a standard Qwerty virtual keyboard. With this approach users can tap on the characters, flick to input Space, Backspace, and Enter, or employ an upwards flick on a key to input uppercase letters. This patent, however, does not report on the results of empirical studies. Since then, many works have made similar proposals for pen-operated keyboards (Hashimoto and Togasi, 1995; Isokoski, 2004; Masui, 1998).

Isokoski (1995) developed a model of expert performance for gesture-augmented keyboards and assessed its performance on a realistic text entry task for several keyboard layouts in a user study. Results showed that gestures are significantly slower at first, but can match the speed of tap-based stylus text entry after twenty sessions. A commercial product, called the Hot Virtual Keyboard[6], also augments virtual Qwerty keyboards with gestures. The keyboard uses the right, left, up and down-left flick gestures for Space, Backspace, Shift, and Enter, respectively. Similarly, the default keyboards on Windows Mobile 5 and 6 utilized the right, left, and up flick gestures for Space, Backspace, and Shift, correspondingly.

Zhai and Kristensson (2003) developed a tap and gesture hybrid, where unistroke gestures are assigned for the most frequent words based on users' finger movement pattern on a keyboard. With this method, users effectively draw gestures for known words, and tap on the keys for the unfamiliar ones. However, in a later version they removed the necessity to alternate between gestures and taps by improving the system to handle a significantly larger set of words (Kristensson and Zhai, 2004). A similar method, called Swype[7], also permits users enter words as gestures. It uses shape recognition to identify the words, as the resulting stroke forms a shape that is very often unique to the intended word. In case the shape matches multiple words, users can select the desired word from a short list. These techniques are word-based as a whole word is input at once. When there is no match in the words list user can still resort to tapping (Zhai and Kristensson, 2012). Both of these techniques are somewhat similar to an earlier word-based technique

[6] http://hot-virtual-keyboard.com

[7] http://www.swypeinc.com

called Cirrin. The Cirrin interface arranges all characters inside the perimeter of an annulus. To input each character, users have to move the stylus into and out of the appropriate sector of that annulus (Mankoff and Abowd, 1998).

Li et al. (2011) developed a reduced virtual keyboard that combined the three rows of the Qwerty layout into a single line with eight keys. It uses word frequencies to disambiguate between similar key sequences. To select a less probable word from the prediction list, one has to flick up on the keyboard. Similarly, to input Space, Backspace, or Enter, one has to tap on the bezel, or stroke to the left or to the right on the keyboard, respectively. Recently, Arif et al. (2014) developed a virtual Qwerty keyboard that replaced the Space, Backspace, Shift, and Enter keys with strokes—straight-line gestures swiped to the right, left, up, and diagonally down-left, respectively.

2.2.9 Pressure in Text Entry

Previous work has investigated pressure-based user interfaces and widgets and a few attempts even focus on pressure-based text entry. Most of this work has targeted tabletops or large displays, not handheld devices. The main reason for this is technological, as most current handheld devices do not provide hardware support for measuring pressure. Nonetheless, recent work (Graham-Rowe, 2010; Nurmi, 2009) indicates that future handheld devices may include pressure-sensitive touchscreens as an alternative interaction modality. A recent opaque touchpad already provided support for detecting pressure levels[8].

Herot and Weinzapfel (1978) were the first to investigate the ability of humans to apply pressure and torque on a computer screen. Buxton et al. (1985) also explored touch-sensitive tablet input. They concluded that although pressure control can be difficult in the absence of button clicks or similar tactile feedback, it is a promising research area. Srinivasan and Chen (1993) asked users to control the force applied to a sensor under several different conditions as well as different forms of feedback. Their results suggest that pressure interfaces need to have a force resolution of at least 0.01 N to make full use of human capabilities. Mizobuchi et al. (2005) examined the properties of force-based input on a mobile device by asking participants to apply force in ten predetermined target levels, ranging from 0 N to 4.0 N, with and without visual feedback. They suggested that pressure levels from 0 N to 3.0 N are comfortable and controllable for

[8] http://www.synaptics.com/solutions/products/forcepad

users. Ramos et al. (2004) investigated users' ability to perform discrete target selection tasks by varying a stylus's pressure, with full or partial visual feedback. Based on their results they proposed a number of pressure widgets for tasks such as zooming and selection. Similar to Mizobuchi et al. (2005), they concluded that users could control 6 ± 1 pressure levels without major difficulties. In a different avenue of work, Zeleznik et al. (2001) proposed an alternative to binary button switches on mice. With their technique one had to press a button lightly to activate its first state and harder to activate its second state. Likewise, Cechanowicz et al. (2007) permitted users to apply different pressure levels on a mouse to simultaneously control cursor position and multiple levels of discrete selection modes in desktop tasks. They, however, did not evaluate their techniques.

2.2.9.1 Pressure-Based Text Entry Techniques

In contrast to the above-mentioned findings, text entry studies showed that pressure does not work well in this domain (McCallum et al., 2009; Wang et al., 2009), as techniques with more than two pressure levels suffer from relatively higher error rates. McCallum et al. (2009) introduced a pressure-based text entry technique for the standard 12-key mobile keypad that utilizes three pressure levels. Their technique yielded a higher expert text entry rate compared to Multi-tap, but at the expense of an 8.7% error rate (compared to a baseline of 2.8%). Tang et al. (2001) developed a three-key chorded keyboard with three pressure levels, which also suffered from high error rates, ~18% after three trials. Hoffmann et al. (2009) modified a standard Qwerty keyboard to use key resistance to prevent errors. The keyboard used dictionary, grammar, and context tests to identify probably erroneous characters, and then made those keys harder to press by increasing the resistance. This reduced erroneous keystrokes by 87% and correction attempts by 46%, on average. Similarly, Dietz et al. (2009) developed a pressure sensitive physical keyboard that used different pressure levels to enable users to delete one character or a word using the Backspace key. However, they did not evaluate their work. Jong et al. (2010) presented a tactile input method for pressure sensitive keyboards based on the detection and classification of pressing movements on already pressed-down keys. Yet they too did not compare their techniques with conventional ones. Brewster et al. (2009) presented several pressure-based techniques to switch between uppercase and lowercase letters on a virtual Qwerty keyboard. Some of their techniques were more accurate and faster than the standard Shift key. Arif et al. (2010) proposed a pressure-based error prevention technique that used bigram character frequencies to predict improbable characters based on the previous one and made those characters harder to input. In order to input an improbable character, users had to press the corresponding key with extra pressure. They conducted a pilot that showed that the technique might reduce error significantly for novice users, when

augmented with synthetic tactile feedback (i.e., vibration). Yet a later investigation indicated that it is relatively more difficult for expert users to adapt to this text entry technique (Arif and Stuerzlinger, 2013).

2.2.9.2 Pressure Detection Simulation

As most current touchscreen devices do not provide the hardware support for measuring pressure, two software-level solutions are widely used to simulate pressure detection: time-based and contact-area-based. The time-based approach simulates pressure detection based on the assumption that it takes more time to perform a task when extra pressure is applied (Cechanowicz et al., 2007; Ramos et al., 2004). It records the average time it takes to perform a task and uses that as a baseline. When users take more time than the baseline, the system deduces that extra pressure is applied. Several mobile applications such as Doodle Buddy[9] and TypeDrawing[10] use this to simulate pressure detection. The limitation of this approach is that it forces users to take additional time to perform all tasks that require extra pressure. Yet Raisamo (1999) showed that many tasks take almost the same time regardless of the level of pressure applied. Thus, a time threshold for pressure may unnecessarily slow users down.

The contact-area-based approach relies on the fact that human fingertips spread wider over the point of contact when additional pressure is applied (Buxton, 2013). It simulates pressure detection by mapping changes in a finger's contact area to changes in pressure. More specifically, this approach maps different finger areas to different pressure levels, and simulates pressure detection based on that. Forlines and Shen first implemented this approach (2005), although they did not elaborate on their implementation. Benko et al. (2006) provided a detailed explanation of this method. They also demonstrated that this technique does not require per-user training and discussed how it could be used in touchscreen user interfaces. Boring et al. (2012) investigated pressure detection simulation using the thumb's contact area. The fundamental limitation of this approach is that finger contact areas depend not only on finger sizes and the amount of pressure applied, but also on different touch types such as vertical and oblique (Wang et al., 2009). Thus, this approach cannot be used with all users or with styli. Besides, most current touchscreens provide touch coordinates, not contact area information.

A few studies identified that touch-points or coordinates move with extra pressure (Wang et al., 2009;

[9] http://blog.pinger.com/tag/doodle-buddy
[10] http://www.storyabout.net/typedrawing

Ramos et al., 2004; Boring et al., 2012). Wang et al. (2009) argued that most touch interactions are oblique due to common practices in handing physical objects and to accommodate the long fingernails of some users. They observed that touch-points shift more when users apply extra pressure. Boring et al. (2012) also reported this. A similar tendency was observed for stylus-based interactions as well (Ramos et al., 2004). However, none of these works evaluated the usability and performance of this idea. Chapter 6 explains how this approach can simulate pressure detection on touchscreens and establishes how this idea performs in practice.

Hwang and Wohn (2012) proposed an alternative pressure detection simulation technique. They monopolize a mobile device's built-in microphone to detect five different pressure levels by mapping different sound amplitude to different pressure levels. In a pilot study, their technique was found to be 94% accurate. Heo and Lee (2011) used acceleration data along the z-axis to differentiate between two pressure levels on touchscreens. In an investigation, their technique was about 90% accurate. However, it is unclear whether this technique will work in mobile settings or not. Watanabe et al. (2012) used the light transmitted by touchscreens onto fingernails to estimate the level of force applied, which changes the intensity of the transmitted light. However, this method is impractical in many situations, as it requires a light sensor attached to be attached to one's fingernail(s).

2.3 Text Entry Performance Metrics

User study data on text entry performance reported in the literature varies widely due to the use of different performance metrics and study designs (Soukoreff and MacKenzie, 2003; Wobbrock, 2007). Therefore, it is sometimes difficult to compare studies or to extract meaningful average text entry speeds and error rates from this body of work. This makes it hard for designers and researchers to use and apply these results and prevents the synthesis of a larger picture.

This section discusses the most common performance metrics employed in text entry user studies. In the literature different notations and terms are used to describe various concepts. For better understanding and to avoid confusion the metrics are here discussed using the following notations, which were formerly introduced by Soukoreff and MacKenzie (2003).

- *Presented Text (P)* is what participants had to enter, and $|P|$ is the length of P.
- *Transcribed Text (T)* is the final text entered by the participant, and $|T|$ is the length of T.
- *Input Stream (IS)* is the text that contains all keystrokes performed while entering the presented text, and $|IS|$ is the length of IS.

- *Correct (C)* is the number of correct characters in the *T*.
- *Incorrect Not Fixed (INF)* is the number of unnoticed errors, in other words, incorrect characters, in the *T*.
- *Fixes (F)* are keystrokes in the input stream that are edit functions, such as Backspace and Delete, modifier keys, such as Shift, Alt, and Ctrl, or navigation keys, such as the arrow keys, and mouse movements and clicks.
- *Incorrect Fixed (IF)* keystrokes are those in the input stream that are not editing keys *(F)*, but which do not appear in the final *T*.
- *Minimum String Distance (MSD)* is the minimum number of operations needed to transform *T* into *P*, where the operations are insertion, deletion, or substitution of a single character.

Simplifications for *INF* and *C* were also introduces by Soukoreff and MacKenzie (2001, 2003), which consider only the size of the *P* and *T*:

$$INF = MSD(P,T) \quad \text{Equation (2)}$$

$$C = MAX(|P|,|T|) - MSD(P,T) \quad \text{Equation (3)}$$

2.3.1 Entry Speed

Calculating the text entry rate for various input methods is usually straightforward. The Words per Minute (WPM) metric is the most frequently used empirical measure of entry speed (Yamada, 1980). A few other metrics exist, such as Gestures per Second (GPS), Adjusted Words per Minute (AdjWPM), and Keystrokes per Second (KSPS). But these are rarely used.

2.3.1.1 Words per Minute (WPM)

Word per Minute (WPM) measures the time it takes to produce certain number of words. WPM does not consider the number of keystrokes nor the gestures made during the text entry. It depends only on the length of the transcribed text. WPM is defined as the following.

$$WPM = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5} \quad \text{Equation (4)}$$

Here, *S* is time in seconds measured from the first key press (or a similar action such as stylus tap) to the last, including backspaces and other edit and modifier keys. The constant 60 is the number of seconds per

minute, and the factor of one fifth accounts for the length of a word as it has been common practice to regard an "average" word as five characters, including spaces, numbers, and other printable characters (Yamada, 1980). Note that *S* is measured from the entry of the very first character to the last, which means that the entry of the first character is never timed, which is the motivation for the "*–1*" in the numerator of Equation (4). While this assures accuracy, some other researchers omit this factor.

2.3.2 Error Rate

Unlike text entry rate, measuring the error rate is complex. There are many error rate metrics that are used. None of them is perfect as all of the metrics face difficulties distinguishing errors corrected during text entry and those that remain as uncorrected errors. This section discusses the five most frequently used error metrics.

2.3.2.1 Error Rate (ER)

The "raw" Error Rate (ER) is traditionally calculated as the ratio of the total number of incorrect characters in the transcribed text to the length of the transcribed text.

$$ER = \frac{INF}{|T|} \times 100$$ Equation (5)

2.3.2.2 Erroneous Keystrokes (EKS)

The Erroneous Keystrokes (EKS) error rate is only meaningful when the final transcribed text does not contain any error. This metric simply measures the ratio of the number of erroneous keystrokes (EKS) to the length of the presented text. EKS is thus almost equivalent to ER. The difference is that the first usually keeps count of erroneous keystrokes at run time, while the latter considers only the errors that remained in the transcribed text.

$$EKS = \frac{EKS}{|P|} \times 100$$ Equation (6)

ESK can also be derived using the equation: *EKS = INF + IF*.

2.3.2.3 Keystrokes per Character (KSPC)

Keystrokes per Character (KSPC) is the ratio of the length of the input stream to the length of the transcribed text.

$$KSPC = \frac{|IS|}{|T|}$$ Equation (7)

2.3.2.4 Minimum String Distance Error Rate (MSDER)

The Minimum String Distance Error Rate (MSDER) metric was introduced based on the application of the Levenshtein string distance statistic (Levenshtein, 1966) to the problem of matching (incorrect) input to the target text (Soukoreff and MacKenzie, 2001). The algorithm yields the minimum distance between two strings (MSD) defined in terms of edit operations such as *insertion*, *deletion* and *subtraction* of a single character. The idea is to find the smallest number of operations to transform the transcribed text to match the presented text, and then to calculate the ratio of that number to the larger of the lengths of the presented and transcribed texts.

$$MSDER = \frac{MSD(P,\ T)}{Max(|P|, |T|)} \times 100$$ Equation (8)

Here, $MSD\ (P,T)$ is the minimum string distance between the presented and the transcribed text. Later, an improved version of the *MSD* error rate was proposed (Soukoreff and MacKenzie, 2003), which uses the ASCII representation of the differences between the presented and transcribed text to address the disparity in lengths.

2.3.2.5 Total Error Rate (TER)

Total Error Rate (TER) is a unified method that combines the effect of accuracy during and after text entry (Soukoreff and MacKenzie, 2003). This metric measures the ratio of the total number of incorrect and corrected characters to the total number of (initially) correct, incorrect, and corrected characters. In other words, it computes the ratio between the effort of error correction and the total effort to enter the text.

$$TER = \frac{INF + IF}{C + INF + IF} \times 100$$ Equation (9)

2.3.2.6 Limitations of the Error Metrics

The two most widely used error metrics, ER and MSDER, can be considered to be almost equivalent (Soukoreff and MacKenzie, 2003). However, both do not consider the cost of error correction but only the errors still present in the transcribed text. This can make these two metrics misleading. For instance, if all the erroneous character were corrected in the transcribed text, these two metrics will yield results equivalent to having entered the text error free from the start. In other words, they do not consider the effort that was put into correcting errors. EKS is useful when the transcribed text is error free, which is usually achieved by forcing participants to correct each error. This metric does not always show an accurate picture, especially when the transcribed text was not error free, as it cannot differentiate between corrected and uncorrected errors. KSPC considers the cost of committing errors and fixing them, but does not provide any way of separating these two quantities. Nevertheless, there is an (approximately) inverse relationship between KSPC and (ER, EKS, and MSDER). Yet there is no obvious way of combining these measures into an overall error rate (Soukoreff and MacKenzie, 2003). TER, on the other hand, not only measures error rates but also takes the effect of accuracy into account. This provides more insight into the behaviors of the participants. This makes TER the most appropriate error rate metric at the present time.

2.4 Text Entry Performance

This section presents a survey of user study data collected for the most important text entry techniques. Several precautions were taken while collecting data to ensure the integrity of the final results. All articles that do not provide complete data about their user studies, use unorthodox performance metrics, or do not follow standard empirical user study procedures were ignored. If an article used unusual metrics, but provided enough data to permit a conversion into standard metrics, it was included with the converted results. Pilot studies, studies that involved languages other than English, numerical and/or special characters, and studies that were carried out with less than six participants per technique were also disregarded. This eliminated a substantial number of publications from consideration. Yet it emphasizes that one cannot perform cross-study comparison without some guarantee on the validity of the results and without solid comparison points. Most of the considered articles conducted the studies using MacKenzie and Soukoreff's short English phrases (MacKenzie and Soukoreff, 2003) as presented text. However, one study used both SMS-style and short English phrases (James and Reischel, 2001). Another study was conducted using phrases with and without numerical and special characters (Költringer and Grechenig, 2004). Only the later data points were considered in this survey. All the surveyed articles used WPM to measure entry speeds, while ER, EKS, MSDER, and TER were used to measure error rates.

Although most of the surveyed articles included the data required for additional analysis, in a few cases that had to be derived from other provided data. While experimenting on mobile keypads, James and Reischel (2001) did not measure errors with any standard metric. They provided the total number of errors in each dataset but did not elaborate on how they counted errors. For instance, assuming that "ant" was discovered as "atn" in the transcribed text, it is not clear if that was counted as a single or multiple errors. However, ER was recalculated from their paper via Equation (5). Some articles did not provide the average or individual session entry speed and error rates and/or the standard deviations in numerical form but in graphs (Kim et al., 2013; Lyons et al., 2004[a], 2004[b]). In those cases, the data were manually measured from the graphs. McDermott-Wells (2006) did not provide average error rates, but presented exhaustive data on different sessions. Recalculating average error rates from the session data was straightforward.

2.4.1 Error Correction Condition

This survey revealed that text entry user studies are conducted with one of three error correction conditions.

1) *None:* In this condition, participants are not asked or allowed to correct their mistakes. Thus, the final transcribed text contains only uncorrected errors. Usually ER or MSDER metrics are used to measure error rates.
2) *Recommended:* In this condition, participants are recommended to correct errors as they identify them. Therefore, the final transcribed text contains both corrected and uncorrected errors. TER is usually used to measure error rates.
3) *Forced:* Here, participants are forced to correct each error to keep the transcribed text error free. Thus, the final transcribed contains only corrected errors. TER is usually used to measure error rates, although some researchers keep a separate count of erroneous keystrokes to measure EKS.

2.4.2 User Expertise

The survey also indicated that most studies recruit participants at the following skill levels.

1) *Novice* users of a technique are those who never used the technique prior to the study or had a very limited exposure to it, i.e., rarely used it.
2) *Experienced* users are those who use the technique frequently, i.e., almost every day, but have not had professional training on it. In this survey, results from the final session of a longitudinal study are considered as experienced user performance.

3) *Expert* users of a technique are those who use the technique professionally. Examples are professional typists and/or those that have undergone professional or extensive training for the technique.
4) In addition, some studies recruit participants indifferent of their skill levels. Here, such participant groups are labeled as *mixed*.

2.4.3 Results

The following table presents the complete result of the survey.

Table 1. Text entry technique performance from literature.

Technique			Participant		Correction	Error	Speed	Error Rate	Ref.
Method	**Type**	**Modal**	***#***	***Expertise***	**Condition**	**Metric**	***WPM (SD)***	***% (SD)***	
Physical Qwerty	*Standard*	*Both hands*	8	Novice	*None*	ER	20 (×)	3.2 (×)	*5*
			11	Experienced	*None*	EKS	64.8 (17.3)	1.8 (0.9)	*12*
			6	Expert	*None*	ER	75.03 (10.6)	0.95 (0.54)	*5*
	Mini	*Two-thumb*	14	Novice	*Recommended*	TER	31.72 (7.0)	6.12 (3.46)	*4*
			14	Experienced	*Recommended*	TER	60.03 (8.40)	8.32 (4.13)	*4*
Virtual Qwerty	*Projection*	*Both hands*	11	Novice	*Recommended*	EKS	46.6 (9.8)	3.7 (2.4)	*12*
	Phone	*Stylus*	7	Mixed	*Recommended*	TER	21.59 (6.42)	7.34 (9.09)	*11*
	Phone	*Two-thumb*	12	Novice	*Recommended*	TER	15.92 (6.6)	10.38 (8.2)	*2*
			12	Experienced	*Recommended*	TER	23.55 (8.5)	12.13 (0.1)	*1*
	Tablet	*Both hands*	10	Experienced	*Recommended*	TER	39.52 (2.5)	10.28 (1.1)	*7*
Twiddler	*Version 1*	*One hand*	10	Novice	*Recommended*	TER	17.5 (6.2)	4.35 (1.67)	*10*
			5	Experienced	*Recommended*	TER	46.3 (11.7)	6.94 (1.73)	*9*
Gesture	*Graffiti*	*Stylus*	10	Novice	*Forced*	EKS	4.0 (1.44)	26.2 (2.6)	*3*
			10	Experienced	*Forced*	EKS	11.4 (3.6)	26.2 (2.6)	*3*
	Graffiti 2	*Stylus*	12	Novice	*None*	TER	9.24 (×)	19.35 (×)	*8*
	Unistrokes	*Stylus*	10	Novice	*Forced*	EKS	4.1 (2.18)	43.4 (16.4)	*3*
			10	Experienced	*Forced*	EKS	15.8 (4.02)	16.3 (10)	*3*
	EdgeWrite	*Stylus*	10	Novice	*None*	MSD	24.0 (2.2)	2.8 (3.4)	*13*
Phone Keypad	*Multi-tap*	*One thumb*	10	Novice	*None*	ER	7.98 (×)	16.05 (×)	*6*
			10	Experienced	*None*	ER	7.93 (×)	28.64 (×)	*6*
	T9	*One thumb*	10	Novice	*None*	ER	9.09 (×)	10.86 (×)	*6*
			10	Experienced	*None*	ER	20.36 (×)	8.4 (×)	*6*

Here, "#" means the total number of participants, "Ref." means references, and "×" means data were not provided in the literature. **References:** *1* (Arif et al., 2011), *2* (Arif et al., 2010), *3* (Castellucci and MacKenzie, 2008), *4* (Clarkson et al., 2005), *5* (Grudin, 1983[a]), *6* (James and Reischel, 2001), *7* (Kim et al., 2013), *8* (Költringer and Grechenig, 2004), *9* (Lyons et al., 2004[a]), *10* (Lyons et al., 2004[b]), *11* (McDermott-Wells, 2006), *12* (Roeber et al., 2003), and *13* (Wobbrock and Myers, 2005).

2.5 Perceptual, Cognitive, and Physical Aspects

Psychologists have often used the task of transcription typing (see Section 2.1) with the standard Qwerty keyboard for the purpose of analyzing human skilled behavior. They have been attracted to transcription, because this task has several advantages over other forms of skilled activities (Salthouse, 1986). One particularly noteworthy advantage is that the number of practitioners is very large compared to other skilled activities, making it easier for researchers to locate moderately sized samples of users at different skill levels (see Section 2.4.2). Also, transcription typing behavior can be partitioned into separate and easily measured responses, such as keystrokes, taps, stylus taps, and strokes, which makes it relatively easier to analyze. Finally, as the task involves complex interactions of perceptual, cognitive, and physical processes, a better understanding of transcription typing may contribute to the knowledge about the nature of highly skilled performance in a wide range of cognitive activities (Salthouse, 1986).

This section discusses empirically established phenomena observed in transcription typing with standard Qwerty typewriters and keyboards, referred to as *typing* in this section for brevity. These phenomena were identified by various researchers and later summarized by Gentner (1983), Salthouse (1986, 1987) and colleagues (1987), John (1988, 1993), and Wu and Liu (2004[a], 2004[b]). This dissertation categorizes these phenomena into five groups: basic behavioral (1-13), units of typing (14-19), errors (20-24), skill learning (25-32), and vision (33-36) phenomena. Although these phenomena were observed in skilled transcription typing of long English prose, a better understanding of these may assist to understand the process of other forms of text entry as well (O'Brien et al., 2008).

2.5.1 Basic Behavioral Phenomena

This section discusses thirteen basic behavioral phenomena (1-13) observed in skilled transcription typing.

1) The average interkey intervals are only a fraction of the typical choice reaction time. The median interkey interval in normal transcription typing is 177 ms, while the median interkey interval in a serial two-alternative choice reaction time task is 560 ms (Salthouse, 1984[a]).
2) Typing is slower than reading. In a study, two different user groups averaged 246 and 259 WPM when reading and 60 and 55 WPM when transcription typing, respectively (Salthouse, 1984[a]).
3) There is no relationship between typing skill and degree of comprehension of material that has been typed. In other words, these two are independent (Salthouse, 1984[a]).

4) Typing rate is independent of word order. Numerous user studies showed that typing rates are almost the same for random words and meaningful texts (Fendrick, 1937; Grudin and Larochelle, 1982; Hershman and Hillix, 1965; Larochelle, 1983; Olsen and Murray, 1976; Salthouse, 1984[a]; Shaffer, 1973, 1978; Shaffer and Hardwick, 1968; Shulansky and Herrmann, 1977; Terzuolo and Viviani, 1980; Thomas and Jones, 1970; West and Sabban, 1982). This is not unusual as in transcription typing users are not required to read, understand, or memorize the to-be-transcribed text (Rayner, 1998).
5) Typing is slower with random character order. Many user studies showed that the average interkey interval increases when the linguistic structure of the presented text degrades; i.e., becomes less structured or more random (Fendrick, 1937; Grudin and Larochelle, 1982; Hershman and Hillix, 1965; Larochelle, 1984; Olsen and Murray, 1976; Salthouse, 1984[a]; Shaffer, 1973; Shaffer and French, 1971; Shaffer and Hardwick, 1968, 1969[a]; Terzuolo and Viviani, 1980; Thomas and Jones, 1970; West and Sabban, 1982).
6) Typing is slower with restricted preview. Numerous user studies indicated that the typing rate decreases substantially when preview to the presented text is restricted (Coover, 1923; Hershman and Hillix, 1965; Salthouse, 1984[a], 1984[a], 1985; Salthouse and Saults, 1987; Shaffer, 1973; Shaffer and French, 1971; Shaffer and Hardwick, 1970).
7) Successive alternate hand keystrokes are faster than successive same hand keystrokes. Several user studies indicated that successive keystrokes from the alternate hands are usually 30 to 60 ms faster than successive same hand keystrokes (Coover, 1923; Fox and Stansfield, 1964; Gentner, 1981, 1982, 1983[a]; Grudin and Larochelle, 1982; Kinkead, 1975; Lahy, 1924; Larochelle, 1983, 1984; Oslry, 1983; Rumelhart and Norman, 1982; Salthouse, 1984[a]; Shaffer, 1978; Terzuolo and Viviani, 1980).
8) More frequent character pairs are inputted faster than the less frequent ones (Dvorak et al., 1936; Grudin and Larochelle, 1982; Salthouse, 1984[a], 1984[b]; Terzuolo and Viviani, 1980).
9) Interkey intervals are independent of word length. In other words, there is no systematic effect of word length on interkey intervals (Salthouse, 1984, 1986; Shaffer, 1978; Sternberg et al., 1978).
10) In continuous normal typing, the first keystroke in a word is usually slower than the subsequent keystrokes (Oslry, 1983; Sternberg et al., 1978; Terzuolo and Viviani, 1980). A number of studies reported that the interval before first keystroke in a word takes approximately 20% longer than the intervals between later keystrokes (Salthouse, 1984[a], 1984[b]).
11) The time for a keystroke is dependent on the specific context in which the character appears. Thus, the interkey interval for a particular character is not constant. Instead the interval depends on the characters that precede and (possibly) follow it (Salthouse, 1984[b]; Shaffer, 1973, 1978; Terzuolo

and Viviani, 1980). This phenomenon is not unexpected considering phenomena 7, 8, and 10 and Fitts' law, which states that the time required to rapidly move to a target area is a function of the distance to the target and the size of the target (MacKenzie, 1991).

12) A concurrent activity does not affect expert typing speed and accuracy. However, Shaffer (1975[b]) claimed that there are limits on the types of activities that may be performed while concurrently typing. For instance, he reported a significant drop in typing performance when users were asked to transcribe text from audio and read aloud a different visually presented text at the same time.
13) Position is an important part of the internal representation of the response. Reaction times are shorter for characters that are positioned corresponding to keyboard locations of the characters to be typed (Logan, 2003).

2.5.2 Units of Text Entry

This section discusses six phenomena (14-19) regarding the following units of transcription typing, defined by Salthouse (1986, 1987).

- *Copying Span* denotes the amount of text that can be typed accurately after a single review of the presented text.
- *Stopping Span* signifies the amount of text to which the user is irrevocably committed to typing. In other words, the amount of text typed after the user has been told to stop.
- *Eye-hand Span* indicates the amount of text intervening between the character receiving the attention of the eyes and the character whose key is currently being pressed.
- *Replacement Span* signifies how far in advance of the current keystroke the user commits to a particular character.
- *The Detection Span* is defined as how far in advance of the current keystroke the user can detect a specially designated target character.

14) The copying span ranges from 7 to 40 characters (Rothkopf, 1980; Salthouse, 1985)
15) The stopping span is between 1 and 2 characters (Logan, 1982; Salthouse and Saults 1985). Interestingly, a similar span was found for error correction as well. Several studies reported that most users notice and correct errors within 1 to 2 characters (Long, 1976; Shaffer and Hardwick, I969[b]). Also, the first keystroke after an error is much slower than other keystrokes (Salthouse, 1984; Shaffer, 1973). This indicates that error detection is often instantaneous and users pause

momentarily after noticing an error before resuming normal typing. Section 3.1.6.3 investigates this further, but for text entry tasks instead of transcription.

16) The eye-hand span ranges between 3 and 7 characters for average to expert users (Butsch, 1932; Hershman and Hillix, 1965; Logan, 1983; Salthouse, 1984[a], 1984[b], 1985; Shaffer, 1973, 1978; Shaffer and French, 1971; Shaffer and Hardwick, 1970).
17) The eye-hand span decreases for unfamiliar, meaningless, or random text in comparison to the value for meaningful text (Hershman and Hillix, 1965). Salthouse (1984[a]) reported on average 3.45 characters eye-hand span for meaningful and 1.75 characters for meaningless text.
18) The replacement span is approximately 3 characters long (Salthouse and Saults, 1985).
19) The detection span is about 8 characters long (Salthouse and Saults, 1987).

2.5.3 Error Phenomena

This section elaborates on five typing error phenomena (20-24) identified in transcription typing. Although these phenomena were reported using different error classifications in the literature, this section uses the classification proposed by Gentner et al. (1983) for easier understanding. Section 2.6.4 explains Gentner et al.'s classification approach in more detail.

20) Only a fraction of errors (40-70%) are detected without reference to the transcribed text (Long, 1976; Rabbitt, 1978; West, 1967). This suggests either that different mechanisms are responsible for producing and detecting errors or that the mechanism that detects errors is faulty (Salthouse, 1986).
21) Substitution errors mostly involve surrounding keys. Grudin (1983[a], 1983[b]) reported that between 31 to 59% substitution errors involve horizontally adjacent and 8 to 16% involved vertically adjacent keys.
22) Many misstroke and insertion errors involve "extremely" short interkey intervals (Grudin, 1983[a]). Salthouse (1986) reported that the median ratio of the interval for a particular keystroke relative to the median interkey interval across all keystrokes is considerably less for both the erroneous (0.68 ms) and the immediately following keystroke (0.871 ms).
23) Many omission errors are followed by a keystroke with an interval nearly twice the overall median (Grudin, 1983[a]; Shaffer, 1975[b]). Dvorak et al. (1936) claimed that these errors are more frequent on difficult-to-reach keys, such as "M" and "N". Thus, the keys involving the little fingers have a higher probability of being omitted than the ones involving the index fingers.

24) Most transposition errors (about 80%) are caused by successive alternate hand keystrokes (Grudin, 1982, 1983[b]; Salthouse, 1986; Shaffer, 1975[a]).

2.5.4 Skill-Learning Phenomena

This section lists eight skill-learning phenomena (25-32). They also indirectly explain how expert users achieve superior transcription typing performance over novice users.

25) Bigrams typed with alternate hands or with two different fingers of the same hand improve faster than bigrams typed with one finger (Gentner, 1983[a], 1983[b]; Salthouse, 1984[a]). This phenomenon indicates that improvements in transcription typing skill require the user to master how to overlap and coordinate the movements of successive keystrokes.
26) The repetitive tapping rate (with both same and alternate hand fingers) increases with increased skill (Salthouse, 1984[a]). This indicates that the precision and coordination of basic execution processes, such as the eye-hand span discussed above, improve with practice (Salthouse, 1984[a]).
27) Interkey interval variability decreases with increased skill. The interquartile range of interkey intervals across all keystrokes decreases about 1.5 ms for every WPM increase in entry speed (Salthouse, 1984[a]).
28) The eye-hand span increases with increased skill (Butsch, 1932; Salthouse, 1984[a], 1985). Studies reported an increase of between 0.5 and 1.2 characters with every 20 WPM increase in entry speed (Salthouse, 1984[a]; Salthouse and Saults, 1985).
29) The replacement span increases with increased skill. Salthouse and Saults (1985) reported that the replacement span increases by about one character with every 30 WPM increase in entry speed.
30) The copying span is dependent on typing skill (Salthouse, 1985[a]; Salthouse and Saults, 1985).
31) The stopping span increases with increased skill (Logan, 1983).
32) Phenomena 25, 26, and 27 differ for different keystroke sequences (Gentner, 1983).

2.5.5 Vision Phenomena

Expert transcription typists usually perform the tasks of encoding the text and then translating that into a sequence of corresponding manual keystrokes in parallel (Rayner, 1998). They use their visual system almost exclusively for the encoding of the presented text, aside from a few occasional glances at the keyboard. Many have attempted to study this to better understand how a users' visual system works in skilled typing. This section summarizes four such phenomena (33-36).

33) Gaze time per character decreases with increased preview window size (Rayner, 1998).
34) The average saccade size is approximately 4 characters (Inhoff and Wang, 1992). A saccade is a rapid intermittent eye movement that occurs when the eyes fix on one point after another in the visual field.
35) The fixation duration is about 400 ms (Inhoff and Wang, 1992). Fixation is defined as the maintaining of the visual gaze on a single location.
36) Attention to the hands disrupts skilled typewriting (Logan and Crump, 2009; Tapp and Logan, 2011). Typists usually slow their rates of typing so they can see which hand types which character.

This section discussed several well-identified phenomena in skilled transcription typing. The next section elaborates on different types of errors and how users tend to handle these errors.

2.6 Error and Error Correction

Text entry errors can be divided into two main categories, system errors and human errors, as both the text entry system and the user can commit errors.

2.6.1 System Errors

Unambiguous text entry techniques such as Qwerty have dedicated keys for each character. There, typing is usually straightforward and free of system errors. Some ambiguous text entry techniques, such as those that use a keypad with fewer keys than the number of characters, involve a software-level ambiguity. To input a character with such techniques one has to perform a predetermined procedure or a sequence of tasks for the software to identify the intended character. For instance, to input the letter “n” with the Multi-tap technique, see Section 2.2.5, one has to press the “6” key twice. Yet, because they are deterministic, these techniques are also typically free of system errors. However, recognition-based techniques involving; e.g., speech, gesture, and handwriting recognition, are more error prone due to the occurrence of system errors (Mankoff and Abowd, 1999). A system error is an instance where the user input the correct information, but the system mis-interprets the user actions and outputs no or the wrong character (or word). Interestingly, even humans can experience visual recognition errors as high as 56% when looking at handwritten word fragments without awareness of the context (Schomaker, 1994). User interaction can also change significantly depending on several uncontrollable variables, such as vocal or finger fatigue and a users’ ambient environment. Moreover, any other notable differences from the situation in which training data was acquired can also reduce recognition accuracy (Frese and Sabini, 1985). Hence, many have claimed

that it is impossible to develop a perfect speech, handwriting, and gesture recognition technique (Mankoff and Abowd, 1999). Nevertheless, researchers are persistently coming up with new methods and strategies to reduce recognition errors and to make these techniques more reliable. Yet, the question if and especially how users adapt to a text entry technique's system errors has not yet been investigated in depth.

2.6.2 Human Errors

It is evident that regardless how perfect an input technique is, humans will still make mistakes (Card et al., 1983). Such errors are an important source of insight into the organization underlying text entry performance (Grudin, 1983[a]). Thus, it is necessary to better understand human errors to improve the overall text entry performance with a given technique.

In the context of text entry, the general assumption is that human behavior is the result of a goal-oriented action (Gentner, 1983; Miller et al., 1960; Norman, 1981). Accordingly, within a goal-oriented framework, certain assumptions can be made about human errors (Brodbeck et al., 1993; Zapf et al., 1992). First, errors only appear in the pursuit of a goal. Yet goals such as to press keys at random are explicitly insensitive to errors. Second, an error implies *failure* to attain a specific or a higher order goal. Failure to attain a low-level goal occurs when, for example, someone attempts to recover deleted text that cannot be recovered. In this case, the user reaches a *dead end* where that particular goal cannot be attained. A higher order goal, in contrast, is unattainable when plans are wrongly arranged. In other words, when a set of correct specific goals is attained but the sequence of sub-goals is incorrectly set up this makes the higher order goal unattainable (Reason, 1990). For instance, when the efficiency of work performance is considered a higher order goal, taking an action detour can be seen as failure to attain that goal. Finally, an unattainable goal should potentially be avoidable, otherwise it cannot be considered as an error (Reason, 1990). For instance, it is not an error if data is lost due to a power outage, as the user cannot avoid such an incident.

2.6.3 The Mismatch Concept

A theory, known as the mismatch concept, holds both the user and the system responsible for committing an error (Brodbeck et al., 1993; Rasmussen, 1984). It suggests that errors cannot be attributed to the user or the system individually. Instead, it is the "mismatch" in the interaction between these two that causes an error. There are mainly two kinds of mismatch problems: *functionality* and *usability*.

Functionality problems are mismatches between the tasks and the system. This kind of problems occurs when the system makes a task more difficult for the users to perform than it is technically feasible (Brodbeck et al., 1993). Also, this applies when the system does not permit users to reach a goal that is required to complete a task. Functionality problems usually impose the following actions on the users (Brodbeck et al., 1993).

- *Action Blockades:* Users are forced to terminate a task-specific goal because certain task-relevant actions cannot be performed with the system.
- *Action Repetitions:* The system forces users to perform a task again because parts of their work have been lost.
- *Action Interruption:* An ongoing work is interrupted for an inappropriate amount of time.
- *Action Detour:* Users have to work around specific functional deficits of the system.

Most of these functionality problems can be avoided by making sure that the system and the tasks are well matched to each other (Brodbeck et al., 1993).

Usability problems, on the other hand, are mismatches between the user and the system. This kind of problems occurs when users do not attain their individual goals and a functionality problem cannot be assumed (Zapf et al., 1992). This implies that it is not sufficient to base user actions only on the description of their tasks, and erroneous actions also have to be observed. Despite the theoretical implication, the attribution to a system component is still important for practical reasons (Brodbeck et al., 1993). If a large number of users are making the same mistake, it might be prudent to change that specific feature of the system. This decision, however, should be taken on the basis of empirical data.

2.6.4 Error Classification

Gentner et al. (1983) compiled a glossary that provides a framework for the classification and description of text entry and transcription typing errors. Although the glossary was initially compiled for text entry and transcription tasks with a typewriter, it also applies for standard and virtual Qwerty keyboards (Wobbrock and Myers, 2006). This glossary is widely used in literature to explain the phenomena of committing errors. Therefore, it is briefly reviewed below.

- *Misstroke Errors:* Inaccurate movements of a finger cause such errors, for example, when users' finger(s) strike multiple keys simultaneously or if they contact another key in passing with sufficient force to activate it.
- *Transposition Errors:* This occurs when two consecutive characters are interchanged.
- *Interchange Errors:* This takes place when two non-consecutive characters are interchanged.
- *Migration Errors:* This occurs when a character is moved to a new position with one or more characters intervening.
- *Omission Errors:* Omission errors occur when a character of a word is left out.
- *Insertion Errors:* This occurs when an extra character is inserted. However, if the insertion of that character is due to erroneous finger movement, it can also be classified as a misstroke error.
- *Substitution Errors:* This takes place when wrong key(s) surrounding the intended one are pressed by mistake.
- *Doubling Errors:* This occurs in words contacting repeated characters, when the wrong character is repeated instead of the correct one.
- *Alternation or Transposed-Doubling Errors:* This occurs in words containing alternate characters, where a wrong alternation sequence is produced (Dvorak et al., 1936).

Table 2 presents examples of each class of human errors presented above.

Table 2. Classification of human errors while inputting text with typewriters or similar devices.

Error Type	Intended	Output
Misstroke	major	m**n**ajor
Transposition	**ma**jor	**am**jor
Interchange	**maj**or	**jam**or
Migration	ma**j**or	**j**maor
Omission	major	majr
Insertion	major	majo**u**r
Substitution	major	**n**ajor
Doubling	b**oo**k	bo**kk**
Alternation	the**s**e	th**s**e**s**

Grudin (1983[a]) conducted a study with novice and skilled users to identify error patterns while transcribing text with a standard Qwerty keyboard. The method used to categorize the errors was very similar to the above-presented classification. Eight novice and six skilled users participated in the study, where they achieved on average 20 and 75 WPM entry speed, respectively. Results showed that the most frequent error types are insertion, substitution, omission, and transposition. Novice users mostly made substitution errors, while skilled users more frequently performed insertion errors. Figure 16 illustrates this.

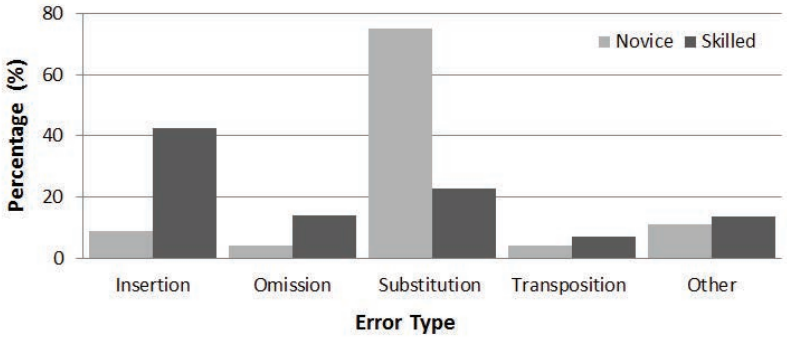

Figure 16. The most frequent types of human errors while transcribing text with a standard Qwerty keyboard.

Alternative ways of classifying errors have also been proposed. Norman (1981), for instance, categorized errors as *mistakes* or *slips* by judging the intentions of the users. A mistake is an error in the intention of the user, while a slip is an error where the intention was correct but an error was made while executing the intention. White (1932) categorized character- and word-level errors with a standard Qwerty keyboard into ten different types. However, this classification method is dependent on the errors remaining in the transcribed text and thus does not account for corrected errors. Soukoreff and MacKenzie (2003), conversely, analyzed input streams instead of transcribed texts so that corrected errors can be detected as well. They included seven new error types relative to Gentner et al.'s (1983) classification to differentiate between corrected and uncorrected errors, as discussed in Section 2.3. Based on Norman's (1981) mistake and slip approach, a recent work (Read et al., 2001) categorized text entry errors for child users, which was then reevaluated and extended in a later work (Kano et al., 2007).

2.6.5 Error Correction

Text entry errors can be corrected with two different strategies: character-level or word-level. In character-level correction any erroneous character is corrected right away. In word-level correction, erroneous key presses are corrected after several other keystrokes have happened following the incorrect one. This kind of strategy is used when experienced users chunk their input or when they do not verify their input right away. Almost all popular text entry techniques provide methods to correct errors that are committed with both strategies (Grudin, 1983[a]). Although some work focused on how frequently errors are noticed and corrected by transcription typists (Long, 1976; Shaffer and Hardwick, I969[b]), currently there is no data on how often these two error correction strategies are used in text entry. It is also unclear whether (and how) users adapt to a faulty text entry technique that frequently misinterprets user input.

2.6.5.1 Error Correction vs. Text Editing

The processes of error correction and text editing are fundamentally different. Error correction can be classified as a set of goals. As the process of committing errors is unintentional, error correction is never planned. In text editing, however, users can set goals that can be formulated by versatile plans that are then addressed by repeated problem solving (Robertson and Black, 1986).

Nevertheless, error correction in longer segments of text is virtually indistinguishable from general text editing. This makes such efforts unpredictable and harder to model. Based on the review in Section 2.5.2, especially Phenomenon 15, and Section 3.1.6.3, this document classifies *text editing* as all correction efforts that occur after inputting twelve or more characters after an erroneous one. Text editing efforts are disregarded in this work henceforth.

2.6.6 Effort vs. Learning

A theory in psychology research identified the durability of episodic memory as a positive function of the degrees of semantic involvement in processing stimuli (Craik and Lockhart, 1972). In other words, peoples' memory recall performance improved with the increased time to process the subsequent stimulus. This was verified through empirical studies that showed that deeper encodings take longer to process, but improved performance in tasks such as recall or recognition for words (Craik and Tulving, 1975). Similarly, a survey of skill acquisition research argued that manipulations that compromise the speed of acquisition could support the long-term goals of training (Schmidt and Bjork, 1992). They showed that encouraging active information retrieval from memory is a common and effective mechanism for skill acquisition in various domains. Motivated by this, prior work investigated the relationship between user effort and spatial memory in user interfaces (Cockburn et al., 2007). Results showed that interfaces requiring greater user effort improve learning for spatial tasks. Other work found that users depend more on memory retrieval than perceptually available information such as labels, when interacting with effortful strategies (Ehret, 2002). Likewise, recent work claimed that interfaces with usability problems may improve system efficiency and user experience in the long run (Riche et al., 2010). Marking menus are an example how this concept could be applied (Kurtenbach and Buxton, 1993). To force users to recall the direction of the intended menu item, they delay the display of the pie menu content. This affects interaction time for novices, but facilitates the transition to expert level (Cockburn et al., 2007). In a recent study, Labahn et al. (2008) observed that users seemed to adapt to an error-prone recognizer after using it for about half an hour. However, they did not investigate this further.

2.6.7 Errors in Gesture-Based Techniques

Most gesture recognition techniques attempt to match a performed gesture to an existing, internal gesture library and return a match score. These libraries contain some form of template for the supported gestures, often based on the number of strokes, their order, direction, and/or the speed associated with them. Section 2.2.7 provided more information on different gesture recognition strategies. When the match score is above a predetermined, but algorithm-dependent threshold, the system performs the action associated with the gesture that yielded the highest match score. In gesture-based text entry, this action is usually the output of a character. There are two types of errors that may occur in most gesture-based techniques: *misrecognitions* and *failures to recognize*.

A *misrecognition* error occurs when the match score is above the predetermined threshold but the system misinterpreted the performed gesture, and thus, outputted an incorrect letter. An example is that the user inputs “u”, but the system recognizes and outputs “v”. Such errors are usually caused by the system and are well known to occur in most gesture-based techniques (Tappert and Cha, 2007). In an informal survey several popular gesture-based text entry systems, including Path Input[11], DioPen[12], and Touch-Writer[13] were explored. The first is a technique similar to Swype (Section 2.2.8.1) for iOS devices and the latter two are character-based techniques for Android OS devices. Even in a short test, each of these systems misrecognized some performed gestures and output incorrect letters. Figure 17 illustrates two such incidents.

[11] https://itunes.apple.com/us/app/path-input-pro/id538744637

[12] http://androidaftermarket.store.aptoide.com/app/market/com.diotek.ime.diopen/3/166925/DioPen

[13] https://play.google.com/store/apps/details?id=glass.main

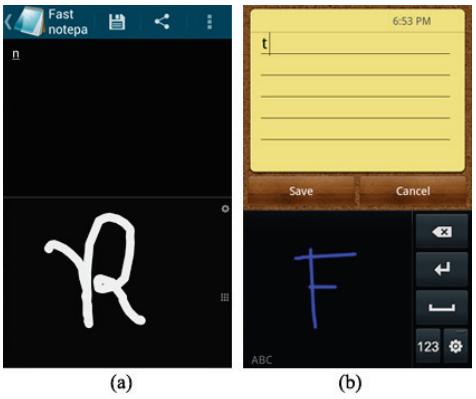

Figure 17. *Misrecognition* errors in: (a) *Touch-Writer* and (b) *DioPen*. In both cases, the user intended to input one character but the system misrecognized it as another. In (a), the user intended to input "R", but the system misrecognized it as an "n". In (b), the user intended to input "F", but the system misrecognized it as a "t".

A *failure to recognize* error occurs when the match score for all templates is below a predetermined threshold or the length of the gesture is too short to be recognized. The survey indicated that human behaviors cause the largest proportion of such errors. A clear example is when the user accidently taps on an interactive surface or prematurely aborts performing or drawing a gesture. Another example is when the user inputs a gesture that is not part of the template library. Most techniques deal with such errors in two different ways. In this situation, they either do not display any output or query the users if they want to include the new gesture in the built-in library to enrich it. Figure 18 illustrates this. Having said that, sometimes a system will fail to recognize a valid gesture due to some other technical issue with the matching algorithm, such as a mismatch between the expected sample density and the true sample density provided by the digitizer.

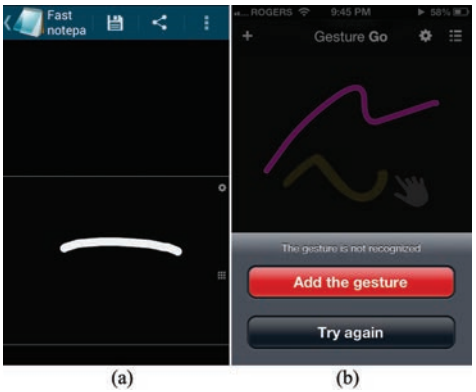

Figure 18. Error handling in: (a) *Touch-Writer* and (b) *Gesture Go*. In (a), the system displays no output when it fails to find a match for the performed gesture in the library. In (b), it asks the user to include the gesture in the library or to try again.

2.6.7.1 Alternative Methods

Some gesture-based techniques allow users to input a given character with several alternative gestures. For example, the Jot technology by Communication Intelligence Corporation provides several alternative ways for drawing some characters. It even enables users to indicate their drawing preference for those characters. Therefore, users can switch the primary method for drawing a character with an alternative one. This is useful in situations when the recognition system does not work well for a given user and a given gesture. Similarly, EdgeWrite provides several variations for drawing some characters. Some commercial products, such as DioPen and Hot Virtual Keyboard, also support alternative gestures. Sections 2.2.8 and 2.2.8.1 provide a brief overview of these techniques.

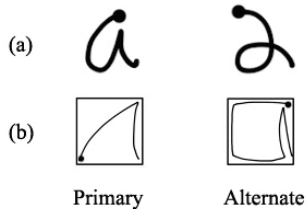

Figure 19. The primary and some alternative methods for drawing "a" with: (a) Jot and (b) EdgeWrite.

Yet alternative gestures are almost always *less intuitive* and *harder to discover* compared to the primary ones. Figure 19 illustrates the primary gesture and one of the alternative ones for inputting the character "a" with Jot and EdgeWrite. There, one can see that the alternative gestures are relatively less intuitive or harder to guess than the primary ones. Also, one has to either go to the extended tutorial (for Jot, EdgeWrite, and Hot Virtual Keyboard) or guess (for DioPen) to discover the alternative gestures.

2.6.8 Modeling Text Entry and Error Correction

Card et al. (1983) presented the GOMS technique to predict user skilled performance time. They separated the human's cognitive architecture into four basic components: *Goals*, *Operators*, *Methods for achieving the goals*, and *Selection rules for choosing among competing methods for goals*. Despite the technique's theoretical power, it was never used on a large scale in the HCI community. The most likely reason is that the cost of first learning the GOMS technique and then constructing a correct model for an interaction method is quite high relative to the accuracy of the results that can be obtained.

Researchers have proposed several variations of GOMS to make modeling easier. The Keystroke-Level Model (KLM), for instance, eliminates all elements but the primitive operators (Card et al., 1980). This

makes KLM comparatively easier to learn and to construct models, but also makes it inadequate for multi-modal techniques. The Natural GOMS Language (NGOMSL) is a high-level syntax for GOMS and based on cognitive complexity theory (Kieras, 1988). Constructing NGOMSL models requires performing a top-down, breadth-first expansion of users' top-level goals into methods and further into primitive operators. Mastering the NGOMSL technique requires significant effort, as does the construction of a correct model. The Cognitive-Perceptual-Motor GOMS (CPM-GOMS) is based on a model human processor (Gray et al., 1992). Unlike other variations of GOMS, CPM-GOMS is capable of modeling multitasking behaviors, because it does not enforce user interaction as a serial process. Nevertheless, CPM-GOMS also requires a thorough understanding of GOMS and the human cognitive architecture.

ACT-R is a cognitive architecture that aims to define the basic, irreducible cognitive as well as perceptual operations that enable the human mind (Anderson et al., 2004). As such, it looks like a programming language at first glance. Constructing an ACT-R model requires a detailed model of a task, which involves significant amount of expertise, time, and effort. Besides, the original form of ACT-R did not handle motor actions and all actions of the perceptual systems correctly, although recent versions rectify some of these shortcomings (Bothell, 2012).

To overcome the complexity of the model construction process, rapid modeling tools such as QGOMS (Beard et al., 1996) and CAT-HCI (Williams, 1993) have been developed. The problem with these tools is that, once a model has been created, it is hard to change the model. In case of upgrades or design changes, the developers have then to either construct a new model or have to calculate the effect of that change by hand. Other tools, such as CRITIQUE (Hudson et al., 1999), depend on research tools that are not commonly available. John et al. (2004) proposed a new system to overcome these problems by integrating HTML mock-ups with ACT-R. This, however, limits the scope to web browsers. Recent improvements in web browsers have made this less of a concern.

There are several models that predict text entry speed or performance. But none of them account for the cost of error correction. The KLM model mentioned above can predict text entry performance, by counting keystrokes and other low-level operations such as the mental preparation and the system's response time. A similar model (Dunlop et al., 2000) forecasts the performance of predictive text entry techniques using three timing elements from KLM. However, its timing elements were measured only for standard Qwerty keyboards. How and Kan (2005) improved that model by defining thirteen operators that map directly to operations on a mobile keypad for different text entry techniques. Later, Hollies et al. (2007) presented another keystroke-level model to measure and predict mobile phone interactions. Their model considers

even advanced interactions, such as using the embedded camera. Soukoreff and MacKenzie (1995) presented a theoretical model to predict upper and lower-bound entry speeds for using a stylus to tap on soft keyboards. The model is based on the Hick-Hyman Law for choice reaction time, Fitts' law for rapid aimed movements, and English linguistic tables for the relative frequencies of bigrams. All of these models predict the performance of particular text entry technique without accounting for error correction methods and behavior. Several other metrics to characterize a techniques' performance exist, as reviewed in Section 2.3.

Suhm (1997) developed an interactive multimodal error correction method for speech recognition. This method can account for switches between different input modalities, such as continuous speech, oral spelling, hand-drawn gestures, choosing from a list of alternatives, cursive handwriting, and typing. To predict the performance of his technique, Suhm introduced a new, high-level model for the cost of error correction, as existing models cannot predict the performance of such a multimodal technique. The model expresses the users' effort on error correction as a compound measure of the time required by the user to correct errors, the response time of the system, the accuracy of the automatic interpretation of corrected input, and the naturalness of the interaction. To overcome the model's technique dependency to some degree, Suhm separates human-specific parameters from system-specific ones. The model, however, does not contain important human-specific parameters such as the visual verification time, human movement time, and the probability of committing an error. Moreover, the relationships between various parameters were not clearly explained.

Chapter 3

Error Correction Conditions

Section 2.4 described how most text entry studies are conducted with one of three error correction conditions: *none*, *recommended*, and *forced*. To summarize, in the *none* condition, participants are not allowed to correct errors. In the *recommended* condition, correction of errors is recommended if and as participants identify them. Finally, in the *forced* condition, participants are forced to correct all errors. To investigate if these conditions have a noticeable effect on popular text entry performance metrics, this chapter presents results of a user study that compared these error correction conditions. The results constitute a notable step towards making it easier to compare different text entry user studies, especially if they involve different error correction conditions.

3.1 A User Study

This study investigated if different error correction conditions have an effect on various text entry metrics. It also examined the relationships (if any) between different error metrics, and attempted measure the rates in which different error correction strategies (as explained in Section 2.6.5) are used in practice. In other words, the study tested the following hypothesis:

($H_{1\ C3}$) *The three error correction conditions used in text entry user studies; i.e., none, recommended, and forced, influence the following performance metrics—WPM, KSPC, ER, EKS, MSDER, TER, and Visual Scan Time (VST).*

Section 2.3 discussed the first six performance metrics, while Section 3.1.3 discusses VST. In addition, the following hypothesis was also tested:

($H_{2\ C3}$) *Text entry speed during the none condition is significantly higher than the recommended and the forced condition.*

This hypothesis is based on the fact that the *none* condition does not allow users to correct errors and thus does not require additional time for error correction.

3.1.1 Participants

Twelve participants, aged from 22 to 45 years, average 27, took part in the study. Appendix A3 elaborates on the procedure used to decide the number of participants (sample size). They were recruited through local university e-mailing lists, posting flyers on campus, and by word of mouth (convenience sampling). Only experienced typists and fluent English speakers were recruited for the study to minimize learning effects. Towards this, anyone who could not achieve an average entry speed of 50 WPM on three short English phrases with a standard Qwerty keyboard was excluded from the study. Moreover, the participants were either native speakers or had spent at least five years in an English-speaking environment. Nine of them were male and all of them were right-hand mouse users. They all received a small compensation (CAD 10.00) for their participation.

3.1.2 Apparatus

A Compaq KB-0133 Qwerty keyboard and an IBM 19″ CRT monitor at 1280×960 pixel resolution were used during the study. A custom Java program logged all keystrokes with timestamps during text entry and calculated user performance directly. The 460×500 application window was positioned at the center of the screen. A fifteen point Tahoma font was used to present text on the screen.

3.1.3 Procedure

During the study, participants entered short English phrases from a phrase set (MacKenzie and Soukoreff, 2003). The corpus does not contain any numeric and special characters, and all uppercase characters were converted to lowercase. It was selected because of its high correlation with the character frequency in the English language. This makes this work comparable to others'. See Appendix A2 for more information on the phrase set. The phrases were displayed to participants one at a time on the screen in a dialog. They were asked to take the time to read, understand, and memorize the phrases, to enter them as fast and accurate as possible, and to press the Enter key when they were done to see the next phrase. Timing started from the entry of the first character and ended with the last (the character before the Enter keystroke). Participants were informed that they could rest between conditions or before inputting a phrase.

During the *none* condition participants were asked not to correct any error. They were instructed to ignore errors and carry on if they noticed errors in their transcribed text. For this condition, all edit functions, modifier, and navigation keys, and mouse operations that could be used to correct errors were disabled.

Thus, participants could not fix their errors even if they attempted to do so by mistake. During the *recommended* condition users were asked to work normally. That is, to correct their errors as they notice them. They were also informed that they could use any edit functions, modifier and navigation keys, or the mouse for error correction. During the *forced* condition an error notification function was used to inform participants of their errors—when an erroneous character was entered the application made a "ding" noise and the input text field turned red. Participants were instructed to take the necessary actions to correct that erroneous character before proceeding. Also, the system prohibited users from submitting an erroneous phrase. In other words, they had to make sure that the final transcribed text was an exact match of the presented one before proceeding to the next phrase.

The system calculated six commonly used performance metrics: WPM, KSPC, ER, EKS, MSDER, and TER. The system also recorded Visual Scan Time (VST), which signifies the time users took on average to visually scan the recently completed phrase before submitting it. The reason for measuring VST is a pilot study, where the experimenter noticed that users usually take additional time to quickly scan through the recently inputted phrase before proceeding to the next one. VST was measured from the time of the last keystroke until the Enter key. Upon completion of the study users were asked to fill a short questionnaire where they could comment on the examined error correction conditions.

3.1.4 Design

A within-subjects design was used. The within-subjects factor focused on the three error correction conditions: *none*, *recommended*, and *forced*. The dependent variables (and the metrics) were entry speed (WPM), error rates (KSPC, ER, EKS, MSDER, and TER), and VST (milliseconds). Section 2.3 defined these metrics. In each condition, participants were asked to complete 60 short English phases. There were five practice phrases prior to each condition, which were excluded from the analysis. Participants were randomly assigned into three groups in a 3×3 Latin square to minimize the effect of asymmetric skill transfer. In summary, the design was:

12 participants ×
3 conditions (three within-subjects conditions: *none*, *recommended*, and *forced* error correction, Latin square) ×
60 short English phrases
= 2,160 phrases, in total. Each participant inputted 180 phrases.

3.1.5 Results

The twelve participants took on average six minutes for each condition, 19 minutes for all three conditions, and about 30 minutes for the whole study including the demonstration and breaks. The highest and lowest average entry speed for the participants were 121 and 55 WPM, respectively.

D'Agostino Kurtosis tests on the dependent variables revealed that the data were normally distributed. A Mauchly's test confirmed that the data's covariance matrix was also circular in form. Thus, repeated-measures ANOVA was used for all analysis. The statistical tests used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%. All statistically significant results are presented with effect size (η^2) and power ($1-\beta$). See Appendix A1 and A4 for more information on η^2 and $1-\beta$, correspondingly.

3.1.5.1 Words per Minute (WPM)

An ANOVA failed to identify a significant effect of error correction condition on WPM ($F_{2,11} = 3.11$, ns). On average WPM for *none*, *recommended*, and *forced* were 81.82, 80.65, and 78.56, respectively. Figure 20 illustrates this.

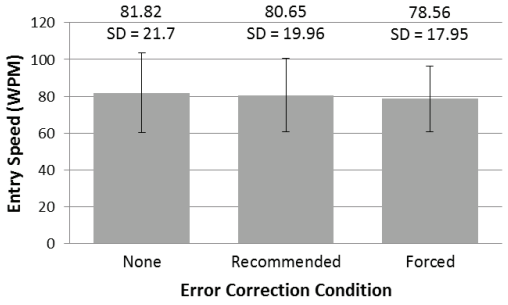

Figure 20. Average entry speed (WPM) for all investigated error correction conditions. Error bars represent ±1 standard deviation (SD).

3.1.5.2 Keystrokes per Character (KSPC)

An ANOVA identified a significant effect of error correction condition on KSPC ($F_{2,11} = 28.46$, $p < .0001$; $\eta^2 = .45$, $1-\beta = 0.94$). A Tukey-Kramer test showed that *recommended* and *forced* had significantly higher KSPC than *none*. On average these two had 8 and 9% more KSPC than *none*, respectively, as illustrated in Figure 21.

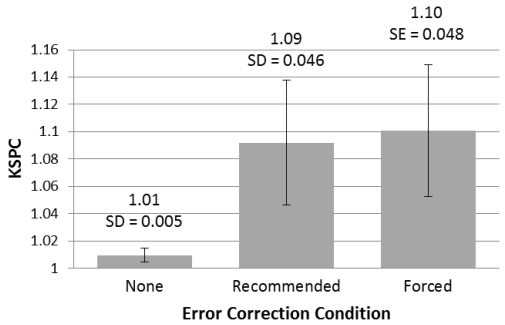

Figure 21. Average KSPC for all investigated error correction conditions. Error bars represent ±1 standard deviation (SD).

3.1.5.3 Erroneous Keystrokes (EKS) and Total Error Rate (TER)

An ANOVA identified a significant effect of error correction condition on both EKS ($F_{2,11} = 8.42$, $p < .005$; $\eta^2 = .11$, $1–\beta = 0.29$) and TER ($F_{2,11} = 9.77$, $p < .001$; $\eta^2 = .09$, $1–\beta = 0.09$). A Tukey-Kramer test showed that *recommended* and *forced* had significantly higher EKS and TER than *none*. On average the two had 66 and 62% more EKS, and 51 and 43% more TER than *none*, respectively. Figure 22 illustrates average EKS and TER.

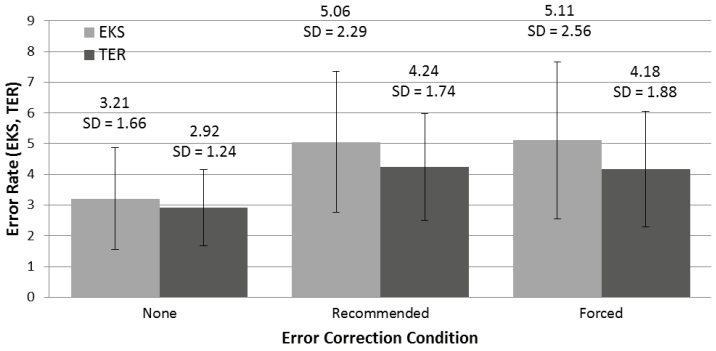

Figure 22. Average EKS and TER for all investigated error correction conditions. Error bars represent ±1 standard deviation (SD).

3.1.5.4 Error Rate (ER) and Minimum String Distance Error Rate (MSDER)

The *forced* condition made sure that the final transcribed text is error free by forcing participants to correct their each mistake. As a result, ER and MSDER measured zero errors for this condition. Thus, these two metrics were analyzed only for the *none* and the *recommended* condition. An ANOVA found a significant effect of error correction condition on both ER ($F_{1,11} = 38.91$, $p < .0001$; $\eta^2 = .53$, $1–\beta = 0.99$) and MSDER

($F_{1,11}$ = 38.65, p < .0001; η^2 = .53, $1–\beta$ = 0.99). Figure 23 presents the average ER and MSDER. Results also revealed that *recommended* had about 18% lower ER and MSDER than *none*.

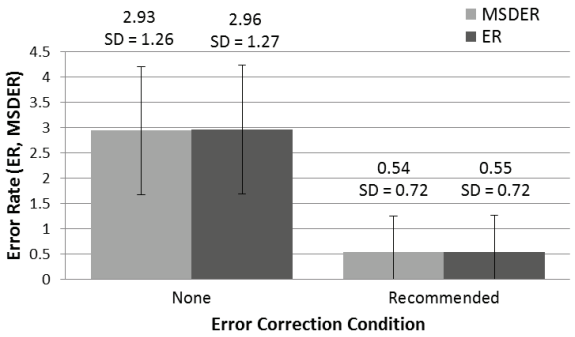

Figure 23. Average ER and MSDER for all investigated error correction conditions. Error bars represent ±1 standard deviation (SD).

3.1.5.5 Visual Scan Time (VST)

An ANOVA failed to identify a significant effect of error correction condition on VST ($F_{2,11}$ = 0.39, ns). The average VST for *none*, *recommended*, and *forced* were 294, 348, and 252 ms, respectively. Also, no obvious relationship was found between VST and the length of the transcribed text, and between VST and entry speed.

3.1.6 Discussion

The results of the study support acceptance of the hypothesis $\boldsymbol{H_{1\ C3}}$ (see Section 3.1) for all error metrics (KSPC, ER, EKS, MSDER, and TER), but not for entry speed (WPM) or Visual Scan Time (VST). The results also do not support acceptance of the hypothesis $\boldsymbol{H_{2\ C3}}$. The following subsections discuss the findings of the study in more detail.

3.1.6.1 Entry Speed

It only seemed natural to assume prior to the study that the *none* condition will yield a higher WPM than *recommended* and *forced*. This is based on the impression that entering error free phrases would require more time, as the measure of time for the former condition would be the sum of the entry time and the error correction time. Surprisingly, the data did not support this assumption. Results showed that error correction conditions did not have a significant effect on WPM for expert users. There are two potential reasons for this. First, the WPM calculation, see Equation (4), considers all the characters in the transcribed text, not only the correct ones. This means, incorrectly inputted characters during *none* and *recommended* were also

counted for the WPM calculation. Second, during the *none* condition, participants often instinctively tried to correct their errors before they remembered that they could not. Such a failed error correction attempt requires a bit of time, as participants need to mentally recover and resume the original task. Similarly, during the *recommended* condition, participants tended to correct their errors almost the moment they made them, making this condition close to the *forced* condition. This is also apparent in the rate at which edit operations, such as Backspace, Delete, and mouse clicks to reposition the cursor, were used during the conditions. An ANOVA showed that there was no significant difference between the number of edit operations in *recommended* and *forced* ($F_{1,11} = 0.65$, ns), and the edit operations did not significantly differ across conditions. Also, no obvious relationship between users' entry speed and their instinctive attempts to correct errors was found. However, novice users may display different behaviors. Figure 24 shows the average edit operations for each condition.

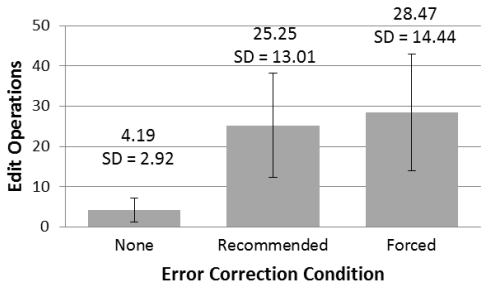

Figure 24. Average edit operations for all investigated error correction conditions. Error bars represent ±1 standard deviation (SD).

Interestingly, participants almost exclusively used the Backspace key while correcting errors. This occurred although users were informed beforehand that they could use the keyboard shortcuts or the mouse (a mouse click was considered as one operation) to correct errors, if they wanted to. Nonetheless, during the study, about 99% of the all edit operations were Backspace.

3.1.6.2 Error Rate Metrics

The result showed that there was a significant effect of error correction condition on all major error metrics. This finding underlines the importance of presenting error rate measures along with the WPM measure when comparing text entry techniques. Results also showed that *recommended* and *forced* had significantly higher KSPC than *none*. The likely reason behind this result is that the KSPC measure compares the input stream and the transcribed text, not the presented text. As both the input stream and transcribed text contain erroneous characters, the KSPC value always remains lower for the Qwerty keyboard. Yet the KSPC value is always higher than 1.0 because of the presence of edit, modifier, and navigation keystrokes in the input

stream. The results also indicate that the ER and MSDER measures are almost equivalent, see Figure 23. This matches the insights mentioned in Section 2.3.2. Other error rate measures, however, do not seem to have any simple relationship that would enable conversion from one to another.

3.1.6.3 Error Correction Strategies

Upon completion of the study, all users responded that they found the *recommended* condition closer to real-life text entry, since this condition did not restrict them from their natural process of error correction. In other words, this condition represents natural text entry behavior more closely than the other conditions. Consequently, the data for the *recommended* condition was extracted and used to investigate how often character- and word-level error correction strategies, mentioned in Section 2.6.5, were used. As a counterbalanced design was used, the effect of asymmetric skill transfer can be ignored in this context.

The number of Backspace keystrokes in correction episodes was utilized to detect how quickly participants noticed and fixed errors. This was motivated by the fact that about 99% of all error correction episodes involved only the Backspace key. The remaining 1% was keyboard shortcuts, navigation, Delete keys, or the mouse. This high percentage of Backspace usage may hold only for short English phrases and not for longer texts. However, text entry and text editing are fundamentally different as the latter requires repeated problem solving and versatile planning, as explained in Section 2.6.5.1. This justifies the decision of using data for short English phrases and limiting all analysis to error correction episodes involving Backspace.

As mentioned above, character-level corrections are performed immediately after an erroneous character is entered, while word-level corrections happen later. An analysis on the data indicated that the proportion of error correction strategies was balanced. On average, 50.29% (SD = 7.2) of all correction efforts were character-level, while the remaining 49.71% (SD = 7.2) were word-level corrections. Figure 25 illustrates the different strategies of error correction by each participant.

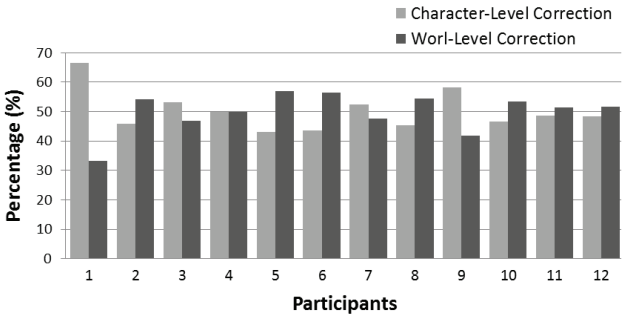

Figure 25. Character-level and word-level error corrections in text entry.

The word-level corrections were further analyzed to calculate more precisely when errors got noticed and corrected. Results indicate that 96.10% of all times an erroneous character got noticed and fixed between the second and fifth character, counted from the erroneous one, and the rest got noticed within twelve characters. A few errors were identified and corrected only when participants visually scanned the entered text *after* they were finished with the phrase. However, these correction episodes were not considered in this analysis, as this behavior occurred rarely, in 1.7% of all cases. Sometimes errors happened during the error correction process as well. An example of such an incident would be that the user first inputted "b" instead of the desired "a", and then erroneously inputted "c" while attempting to fix the previous error. Analysis showed that on average 6.68% (SD = 3.97) of the total errors were committed during the correction process. Of these, 86.11% were corrected in the second try, and the rest on the third iteration. No incidents were identified where more than three attempts were required to fix an error.

3.1.7 Limitations

The ANOVAs identified a significant effect of error correction condition on all error metrics, which includes Keystrokes per Character (KSPC), Erroneous Keystrokes (EKS), Total Error Rate (TER), Error Rate (ER), and Minimum String Distance Error Rate (MSDER). A post-hoc analysis detected a *large* effect size for KSPC, EKS, ER, and MSDER, and a *medium* effect size for TER (see Appendix A1). Further post-hoc analysis revealed that the statistical power exceeded the 0.80 threshold (see Appendix A4) at the observed *medium* to *large* effect size levels for all dependent variables, except for EKS and TER, see Table 3. In other words, there was less than adequate statistical power for EKS and TER. While a larger sample size (*N*) may be necessary to achieve a sufficiently strong statistical power for these dependent variables, due to the *medium* effect size, TER may not have a sufficiently strong effect after all. Appendix A4 elaborates on the criteria used for calculating statistical power.

Table 3. Detected effect size and measured statistical power.

Dependent Variable	Effect Size (η^2)	Power ($1-\beta$)
KSPC	Large	> 0.80
EKS	Large	<< 0.80
TER	Medium	<< 0.80
ER	Large	> 0.80
MSDER	Large	> 0.80

As the study recruited participants by using convenience sampling from the university community, the results may not generalize to a larger population. Nevertheless, an attempt was made to counteract this potential confound by recruiting not only university students but also instructors and staff.

3.2 Summary

This chapter investigated if the way errors are handled in text entry studies has any effect on popular text entry metrics. Results of a user study showed that the way human errors are handled has a significant effect on all frequently used error metrics. This chapter also investigated how often character- and word-level error correction strategies are used in text entry. Results showed that the proportion of error correction strategies is almost balanced (50-50%).

The next chapter uses the findings of this chapter to develop and validate a new model to predict the cost of error correction in character-based text entry techniques.

Chapter 4

Predicting the Cost of Error Correction

Error behavior in character-based text entry is not very well understood. All existing models for the cost of error correction account, at best, for errors in an indirect way. They either fail to account for both human- and system-specific parameters or are not general enough to be used with different text entry techniques. Based on the findings of Chapter 2 and Chapter 3, this chapter presents a new model that accounts for both human- and system-specific parameters to measure and predict the cost of error correction. The model is verified in several ways. First, by measuring the cost of error correction for three popular text entry techniques: the standard Qwerty keyboard, stylus-based virtual Qwerty keyboard, and the standard mobile keypad via data presented in Table 1. Second, by conducting a study that verifies that the predicted impact of injected system errors on a standard Qwerty keyboard matches real results.

4.1 The Cost of Error Correction

Error correction involves both human-specific elements, such as the time to tap on a key and the time to verify a correction, as well as system-specific elements, such as the key sequence required to replace a wrong character. The subsequent sections elaborate on these two elements.

4.1.1 Human Error Correction

It is necessary to have a better understanding of human error correction behavior to create a meaningful model. From the analysis of error correction behaviors, presented in Chapter 3, it is apparent that the correction process follows specific patterns. First, users may immediately verify what they have inputted and correct errors right away (character-level correction). Second and because users also chunk their input, they may verify the result only after inputting a few characters or even the whole word(s). In the later scenario, called word-level correction, users navigate to the area where they have noticed an error, correct it, and then resume their work. As explained in Chapter 3, the most common strategy in short-phrase text entry user studies is to use the Backspace key for both character-level and word-level corrections. There seems to be no fundamental difference between the two strategies, except that rewriting multiple erased characters is an integral part of word-level correction, which scales linearly with the number of characters

after which the error was noticed. Consequently, there is no strong need to distinguish between character-level and word-level error correction behaviors in the model.

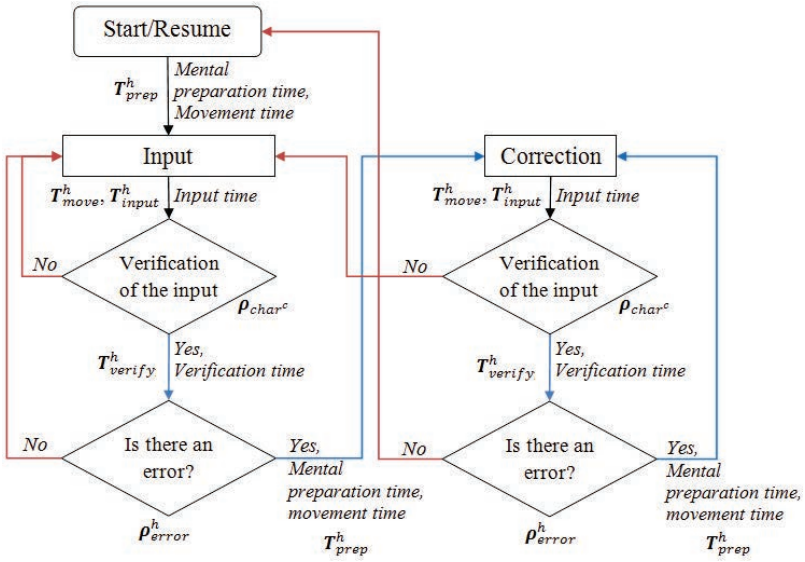

Figure 26. A flowchart representation of human text entry error correction behavior.

Error correction requires additional cognitive and motor steps compared to error-free text entry behavior. Section 2.5 listed several of such cognitive and motor steps, including preparation and movement times. Based on that list and the observations made during the study reported in Chapter 3, Figure 26 illustrates the expected sequence of steps for normal text entry and error correction in a flowchart. As illustrated, there are the following human-specific parameters for error correction.

- $\boldsymbol{T^h_{prep}}$, the *preparation time*, is the average cognitive planning and decision time to start or resume a task.
- $\boldsymbol{T^h_{move}}$, the *movement time*, is the average time required by the user to move their finger(s) to the intended key(s).
- $\boldsymbol{T^h_{input}}$, the *input time*, is the average time necessary for the user to perform a single keystroke or similar operation such as a stylus tap, a mouse click, or a gesture.
- $\boldsymbol{T^h_{verify}}$, the *verification time*, is the average cognitive time required to verify that output matches input.
- $\boldsymbol{\rho^h_{error}}$, the *human-specific error probability*, is the probability of users making an error when performing a keystroke or a similar operation.

4.1.2 System Error Correction

Text entry techniques use both *open* and *closed loop* systems. In closed or feedback loop systems, inputs trigger the processes and the processes control the outputs. For example, while entering a character, a keystroke triggers the process that decides which character to display on the screen. On the other hand, in open loop systems, user inputs trigger only the processes that convert the input to the output, while the user can continue working. In other words, an open loop system does not directly monitor the output of the process that it is controlling. In some text entry techniques such as handwriting and speech recognition a specific command or operation may be processed in the background instead of the result being immediately displayed. One technical motivation for this is that some handwriting recognition techniques cannot be performed fast enough. Figure 27 illustrates the input handling of text entry techniques in a flowchart, where the shaded tasks are performed only in closed loop systems.

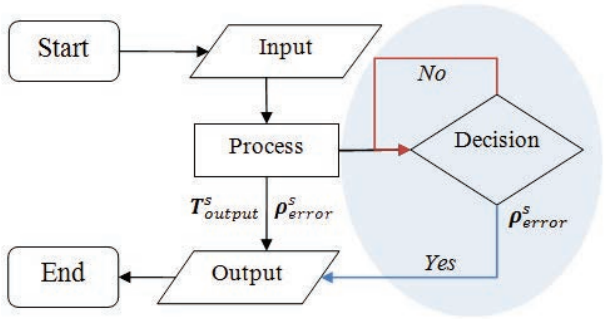

Figure 27. Input handling in text entry techniques.

Depending on the technique, some recognition tasks may take significant time. That is why it is important to identify system specific parameters that may play a role in error correction procedures.

- $\boldsymbol{s}$, the *system*, is a particular text entry technique, defined by a combination of software and hardware. Examples are Qwerty, Multi-tap, T9, etc.
- $\boldsymbol{T^s_{output}}$, the *system output time*, is the time necessary for the system $\boldsymbol{s}$ to process a keystroke or similar operations and output the result.
- $\boldsymbol{\rho^s_{error}}$, the *system-specific error probability*, is the probability of the system making an error when processing an input action.

Some techniques, especially the ones use in closed loop systems, are practically error-free or suffer only from very rare malfunction. Most keyboards are a good example for this. Other techniques, especially

recognition techniques, have to distinguish between potentially fairly similar forms of input and exhibit significant error rates. For example, in handwriting recognition techniques, a common system error misidentifies a “u” as “v”, and vice versa.

4.1.3 Compound Parameters

The following compound parameters are defined based on the human- and system-specific parameters defined above.

- $\boldsymbol{T}_{output}$, the *output time*, is the sum of the time to correct a character entered by the user and the time to process and display that input through the system.

$$\boldsymbol{T}_{output} = \boldsymbol{T}^{h}_{move} + \left(\boldsymbol{T}^{h}_{input} * \left(KSPC_f + 1\right)\right) + \boldsymbol{T}^{s}_{output} \qquad \text{Equation (10)}$$

Here, $KSPC_f$ is the KSPC metric calculated using a corpus’s letter-frequencies. This metric is commonly used in text entry studies to measure the average number of keystrokes required to generate a character of text for a given text entry technique. Here, $\boldsymbol{T}^{h}_{input}$ is scaled with $KSPC_f$ because there are techniques where it takes more or less than one keystroke to enter a character such as Multi-tap and T9. In unambiguous text entry techniques such as Qwerty, it takes exactly a single keystroke to input a character. The “$+\,1$” term after $KSPC_f$ accounts for the fact that the incorrect character has to be deleted first and that users use the dedicated Backspace and/or Delete key. This is justified as 99% of all error corrections episodes used this Backspace key to delete an erroneous character, see above. Here, $\boldsymbol{T}^{h}_{move}$ is added once, as it is usually hard to differentiate between the movement and the input time for the character immediately following a Backspace, as users often position their hand or finger on the next key while still tapping on the Backspace key, a behavior observed in the study presented in Chapter 3. Also, see the replacement span phenomena explained in Section 2.5.2. In the current context $KSPC_f$ can be calculated using the following equation (MacKenzie, 2002).

$$KSPC_f = \frac{\sum(\boldsymbol{K}_{char} * \boldsymbol{F}_{char})}{\sum(\boldsymbol{C}_{char} * \boldsymbol{F}_{char})} \qquad \text{Equation (11)}$$

Here, $\boldsymbol{K}_{char}$ is the number of keystrokes required to enter a character, $\boldsymbol{F}_{char}$ is the frequency of the character in the corpus, and $\boldsymbol{C}_{char}$ is the length of the input, which is 1 for characters. $\boldsymbol{C}_{char}$ ensures that one can generalize this notion to input of more than a single character or word (MacKenzie, 2002).

- $\boldsymbol{T}_{correct}$, the *correction time*, is the compound of the human and system time necessary to correct a single erroneous character in a single attempt. By adding the effort for the mental preparation necessary to fix an error to $\boldsymbol{T}_{output}$, one arrives at the following equation.

$$\boldsymbol{T}_{correct} = \boldsymbol{T}^{h}_{prep} + \boldsymbol{T}_{output} \qquad \text{Equation (12)}$$

Here, the mental preparation time $\boldsymbol{T}^{h}_{prep}$ is added as error correction involves substantial cognitive planning and decision making, including the time to mentally change tracks, see Section 2.5.3 and Section 2.6.

4.1.4 The Probability of Error

The whole probability for an error to happen while inputting text can be expressed using the following equation.

- $\boldsymbol{\rho}_{error}$, the *probability of error*, is the compound of the probability of the system, the user, or the both making an error.

$$\boldsymbol{\rho}_{error} = (\boldsymbol{\rho}^{h}_{error} + \boldsymbol{\rho}^{s}_{error}) - (\boldsymbol{\rho}^{h}_{error} * \boldsymbol{\rho}^{s}_{error}) \qquad \text{Equation (13)}$$

Here, $\boldsymbol{\rho}^{h}_{error} * \boldsymbol{\rho}^{s}_{error}$ represents the probability of a simultaneous error by both the system and the user. This is subtracted, as despite the fact that both parties have committed mistakes, the overall process results only in a single mistake. For instance, when users input an incorrect character and the system misinterprets the same character, the output will contain only a single erroneous character.

4.1.5 The Probability of Noticing an Error

As discussed in Section 3.1.6.3, errors are not always noticed right after they were committed. In-depth analysis of the study logs from the user study discussed in Chapter 3 indicates that the probability that an error will be identified after $\boldsymbol{c}$ characters is subject to exponential decay. This is illustrated in Figure 28, where one can also see that the data can be fit quite well with an exponential function ($R^2 = 0.97$). One may speculate the out-of-line data point after the second character to be a behavioral pattern that does not follow the general trend. Yet there is not enough data to be able to make a definite statement on this.

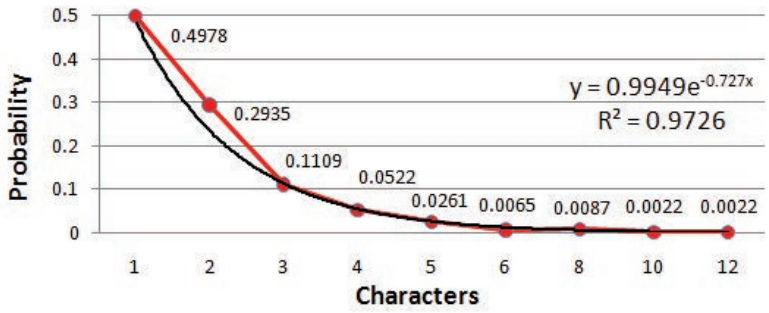

Figure 28. The probability $\rho_{char}{}^{c}$ of noticing an error after the c^{th} character is exponentially distributed.

Thus, the probability of noticing an error after $\boldsymbol{c}$ characters can be modeled reasonably accurately by an exponential distribution. More precisely, when $\boldsymbol{c}$ is a nonnegative integer and $\boldsymbol{c}$ = 1 means that the error was noticed right after committing it, then:

- $\boldsymbol{\rho}_{char}{}^{c}$, is the probability of noticing an error after $\boldsymbol{c}$ characters.

$$\boldsymbol{\rho}_{char}{}^{c} = a * e^{bc} \qquad \text{Equation (14)}$$

Where, e is Euler's number ($e \approx 2.718$), and a and b are constants that are determined by the curve fitting process.

4.1.6 A New High-Level Model for the Cost of Error Correction

Based on the above analysis of error behavior logs, as well as the analysis of human error correction strategies and the relationship of human- and system-specific parameters, this section presents a new model for predicting the cost of error correction.

- $\boldsymbol{T}_{fix}$, the average *extra* time it takes per character to *fix* errors using a particular text entry technique.

$$\boldsymbol{T}_{fix} = \underbrace{\sum_{i=1}^{\infty}\left(\boldsymbol{\rho}_{error}{}^{i} * \underbrace{\sum_{c=1}^{\infty}(\boldsymbol{\rho}_{char}{}^{c} * \boldsymbol{c} * \boldsymbol{T}_{correct})}_{x}\right)}_{y} \qquad \text{Equation (15)}$$

Here, $\boldsymbol{i}$ is the number of corrections and $\boldsymbol{c}$ expresses the number of characters inputted after the erroneous one before the error is identified, also see the previous section. The inner sum (x) expresses the effort for

correcting an error as the probability of noticing an error multiplied by the number of operations necessary to correct it as well as the effort to perform those operations. $\boldsymbol{c}$ is multiplied with $\boldsymbol{T}_{correct}$ as the number of necessary correction operations increases with $\boldsymbol{c}$. The outer sum (y) accounts for repeated error corrections (i.e., correction upon corrections) and the decreasing probability of errors on errors. As the inner and outer sums are both convergent series, one can apply the general formula for geometric sums. First, one finds for the inner term:

$$\sum_{c=1}^{\infty}(\boldsymbol{\rho}_{char}{}^{c} * \boldsymbol{c} * \boldsymbol{T}_{correct}) = \frac{\boldsymbol{\rho}_{char}}{(1-\boldsymbol{\rho}_{char})^2} * \boldsymbol{T}_{correct} \qquad \text{Equation (16)}$$

Then, using the same approach for the outer sum, one arrives at:

$$\boldsymbol{T}_{fix} = \frac{\boldsymbol{\rho}_{error} * \boldsymbol{T}_{correct} * \boldsymbol{\rho}_{char}}{(1-\boldsymbol{\rho}_{error})(1-\boldsymbol{\rho}_{char})^2} \qquad \text{Equation (17)}$$

Equation (17) expresses the *extra* cost of error correction per character, in seconds or milliseconds. As values for $\boldsymbol{\rho}_{error}$ are likely smaller than 20% in any practically useful system and using a first order approximation, one can state that that error correction effort is approximately directly proportional to the reliability of the user and the system, see also Figure 30.

4.1.7 The Cost of Error Correction vs. Error Correction Time

$\boldsymbol{T}_{fix}$ does not predict the time it takes to correct *n* characters of errors. Instead, it predicts the *extra* time it will require on average per character to fix errors with a given text entry technique—regardless if a mistake was made on that character or not. For example, if there are x characters in a phrase then one can say on average it will take additional $\boldsymbol{T}_{fix}$ seconds per character while inputting text with the evaluated technique. In contrast, $\boldsymbol{T}_{correct}$ predicts the time necessary to correct an erroneous character in a single attempt, see Equation (12). Thus, one may use the latter as a measure for the error fixing time for a specific technique.

4.1.8 Limitations of the Model

The new model targets the cost of error correction in character-based techniques where texts are inputted one character at a time. As such, it cannot be applied directly to word-based techniques such as speech, gesture, and handwriting recognition, where texts are inputted word by word. Also, the new model may not apply to more general text entry, as it assumes human error correction behaviors that were identified in the

user study in Section 3.1, where participants transcribed only short English phrases. This model will likely also not work without changes for multi-modal or predictive techniques and scanning keyboards. The reason is that the process of entering text and correcting errors is different from character-based techniques. This model also does not account for environmental distractions such as noise and/or motion. But note that practically all models for input tasks in human-computer interaction (HCI) assume a distraction-free environment.

4.2 Parameter Values

Obtaining the necessary parameters for the new model typically requires controlled experiments. Some of the parameter values are largely independent of a specific technique. This is especially true for the human-specific ones. Other values can be collected from the existing literature on text entry techniques. However, some values need to be found experimentally, such as when new techniques are tested or for existing ones that have not been well studied. To help in some of these situations, this section presents several alternatives to derive various parameter values from commonly used performance metrics. For better representation, prime symbols differentiate the derived values from the directly measured ones.

4.2.1 Calculating the Correction Time ($\boldsymbol{T_{correct}}$)

WPM is the most frequently used empirical measure of text entry performance. This metric is defined in Equation (4). From this, one can approximate $\boldsymbol{T}_{output}$, the sum of the time to correct an erroneous character by the user and the time to process and display the corrected input by the system (as explained in Section 4.1.3), using the following equation.

$$\boldsymbol{T}_{output}{}' = \frac{60}{WPM * 5} * \left(KSPC_f + 1\right) \qquad \text{Equation (18)}$$

Here, $\frac{60}{WPM*5}$ is the time it takes to enter a character, see the derivation of Equation (4). The mental preparation time $\boldsymbol{T}_{prep}^{h}$ was not added, as it has already been accounted for in the WPM value. Besides, WPM does not differentiate between the number of keystrokes made, the cognitive, or the motor time during text entry. Based on this approximation, one can estimate $\boldsymbol{T}_{correct}$ using Equation (12).

4.2.2 Calculating the Probability of Error ($\boldsymbol{\rho_{error}}$)

It is standard practice to present error rates along with WPM when introducing or comparing text entry techniques. Recall from Section 2.4 that errors are usually calculated with one of four error metrics: ER, EKS, TER, or MSDER. These metrics represent the combined errors committed by the human and the system. Hence, one can directly use these values as an approximation for the (compound) probability of error. The two error metrics ER and MSDER are practically equivalent. However, both do not consider the effort that was put into correcting errors. If users reliably corrected every erroneous character, these two metrics would still report the same value as if the text were entered error free from the start. EKS considers the cost of committing errors to some extent, but fails to show an accurate picture when some errors were not corrected. TER overcomes these shortcomings by computing the ratio between the total number of incorrect and corrected characters and the total effort to enter the text, providing more insight into the behaviors of the participants. Thus, TER yields a better approximation to $\boldsymbol{\rho_{error}}$ compared to the others.

4.2.3 Calculating the Probability of Noticing an Error ($\boldsymbol{\rho_{char}{}^{c}}$)

The error recognition delay is well described in Section 4.1.5 by an exponential distribution. Hence, it is possible to calculate $\boldsymbol{\rho_{char}{}^{c}}$ using Equation (14), as illustrated in Figure 28. There, one can see that almost 50% of all errors are noticed right after they are committed. One can assume that is a behavioral "constant", which does not vary across techniques. Therefore, the data presented in Figure 28 together with the approximation $\boldsymbol{\rho_{char}{}^{c}}$ should be applicable to any techniques where text is entered one character at a time.

4.3 Values from the Literature

This section collects data from the literature to approximate the parameters necessary to compute $\boldsymbol{T_{fix}{}'}$ for several popular character-based text entry techniques, presented in Table 4.

The time it takes to perform a mental act depends on what cognitive processes are involved and is highly variable from situation to situation, or person to person. However, Kieras (1993) argued that one could assume that for routine thinking these pauses are fairly uniform in length. Based on this argument, Table 4 reports the same preparation $\boldsymbol{T^{h}_{prep}}$ and verification $\boldsymbol{T^{h}_{verify}}$ times for unambiguous keyboards. Table 4 also presents the same value for the input time $\boldsymbol{T^{h}_{input}}$ for novices and experts for stylus-based virtual Qwerty keyboards. This is based on the observation that there is probably only a small, perhaps negligible, difference

between novices and experts in the motor act of tapping a key with a stylus (MacKenzie and Zhang, 2001). The table, however, do not include system-specific parameters, since such parameters are negligible in widely used text entry techniques. In particular, the reliability of keyboards is extremely high and the time to process and display a character is usually very low, at least compared to the human parameters. How and Kan (2005) performed a user study to derive the "repeated keystroke time" and the "compound time of moving fingers and pressing a key" for the standard mobile keypads. The table subtracted the "repeated keystroke time" from the later to calculate the movement time $\boldsymbol{T}^{h}_{move}$. Finally, $KSPC_f$ is 1 for Qwerty, stylus-based virtual Qwerty, or similar keyboards, since these have dedicated keys for all English letters.

Table 4. Human-specific parameter values for three text entry techniques, collected from the literature. All timings are in seconds.

	Qwerty		**Virtual Qwerty (Stylus)**		**Mobile Keypad (Multi-tap)**	
	Novice	*Expert*	*Novice*	*Expert*	*Novice*	*Expert*
$\boldsymbol{T}^{h}_{prep}$	1.2 [4]	0.6 [4]	0.951 [7]	0.6 [4]	1.285 [6]	0.6 [4]
$\boldsymbol{T}^{h}_{move}$	0.4 [1]	0.4 [1]	0.4 [1]	0.4 [1]	0.96 [3]	0.23 [2]
$\boldsymbol{T}^{h}_{input}$	1.2 [1]	0.12 [1]	0.153 [7]	0.153 [7]	1.21 [3]	0.39 [2]
$\boldsymbol{T}^{h}_{verify}$	1.2 [4]	0.6 [4]	0.951 [7]	0.6 [4]	0.411 [6]	0.411 [6]
$\boldsymbol{\rho}^{h}_{error}$	0.018 [8]		0.0576 [8]		0.091 [8]	
$KSPC_f$	1		1		2.0342 [5]	

References: *1* (Card et al., 1980), *2* (Holleis et al., 2007), *3* (Hudson et al., 1999), *4* (Kieras, 1993), *5* (MacKenzie, 2002), *6* (Pavlovych and Stuerzlinger, 2004), *7* (Soukoreff and MacKenzie, 1995), *8* Chapter 3 User Study.

4.3.1 Prediction and Comparison

Based on the data reported in Table 1 and Table 4, the cost of error correction $\boldsymbol{T}_{fix}{}'$ is predicted here for three popular text entry techniques: standard Qwerty keyboard, stylus-based virtual Qwerty keyboard, and the standard mobile keypad. It is not surprising that the mobile keypad requires the most time with a $\boldsymbol{T}_{fix}{}'$ of about 0.8 seconds per character, while the Qwerty keyboard has the lowest with a $\boldsymbol{T}_{fix}{}'$ of about 0.1 seconds per character, see Figure 29.

As cross-validation, the cost of error correction $\boldsymbol{T}_{fix}{}'$ was also predicted using Equation (17) from WPM values measured in the user study in Chapter 3. The intention was to observe if deriving $\boldsymbol{T}_{output}{}'$ from WPM for $\boldsymbol{T}_{fix}{}'$ gives a closer approximation to the $\boldsymbol{T}_{fix}{}'$ from measured $\boldsymbol{T}_{output}{}'$. The result is illustrated in Figure 29, where one can see that both calculations yield approximately the same result.

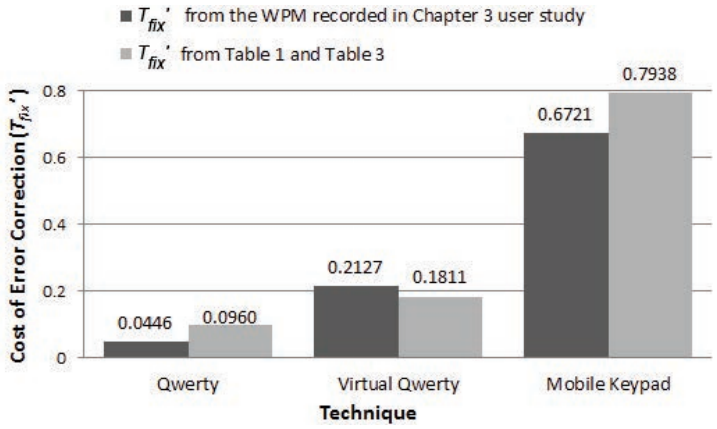

Figure 29. Comparison of the (predicted) cost of error correction per character (T_{fix}' in seconds) for different text entry techniques. The values are calculated from entry speed (WPM) measured in a user study and from human parameters collected from the literature.

4.4 System-Specific Predictions

As discussed earlier, system-specific parameters are usually not significant in commonly used techniques, such as a Qwerty keyboard. This is because most of these techniques are able to process user input with high accuracy and can display the result in very small time frames. However, in some techniques the system specific parameters may become an important factor. In order to analyze the effect of increasingly error prone techniques on the cost of error correction $\boldsymbol{T_{fix}}'$, the probability of system error $\boldsymbol{\rho^{s}_{error}}$ was gradually increased using the data presented in Table 4. The results show that the predicted $\boldsymbol{T_{fix}}'$ increases approximately linearly as the probability of a system error increases, as visualized in Figure 30. To verify this prediction, an empirical study was conducted to observe if this is true in a real-life scenario.

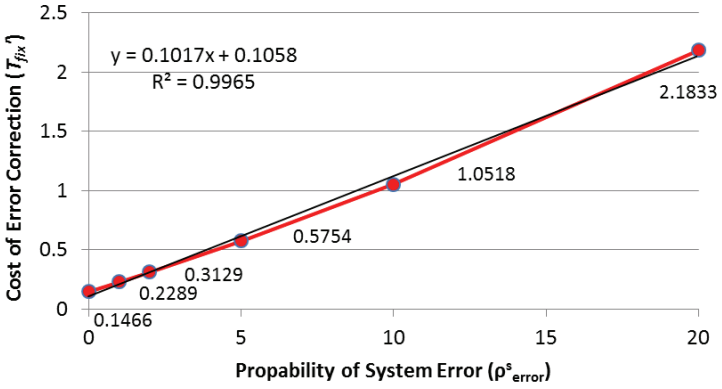

Figure 30. The increase in the cost of error correction prediction (T_{fix}') as the probability of system error (ρ^{s}_{error}) increases.

Note that the value of $\boldsymbol{T}_{fix}{}'$ estimated for the standard Qwerty keyboard with no system errors ($\boldsymbol{\rho}^{s}_{error} = 0$) is slightly higher in Figure 30 (about 0.15 seconds) compared to Figure 29 (about 0.10 seconds). This is because Figure 29 shows a $\boldsymbol{T}_{fix}{}'$ value obtained by averaging the parameter values for both novice and expert users, while Figure 30 uses only novice users. The derivation also assumes that routine thinking pauses for general, non-expert users are very close to those of novices (Kieras, 1993).

4.5 A User Study

The purpose of this study was to observe the effect of various (injected) system error rates on the measured cost of error correction $\boldsymbol{T}_{fix}$. Here, a system error signifies an erroneous output by the system. For instance, the system erroneously outputs "a" when the user inputted "b". Section 4.5.3 discusses this in detail. Thus, the study tested the following hypothesis:

($H_{1\ C4}$) *The cost of error correction (T_{fix}) for a text entry technique increases in proportion with increasing probability of (injected) system errors.*

4.5.1 Participants

Twelve participants, aged from 22 to 46 years, average 28, took part in the user study. Appendix A3 explains the procedure used to decide the number of participants (sample size). They were recruited through local university e-mailing lists, posting flyers on campus, and by word of mouth (convenience sampling). Only experienced standard Qwerty keyboard users and fluent English speakers were recruited for the study to minimize learning effects. Towards this, people with less than eight years of standard Qwerty keyboard experience were excluded from the study. They were either native speakers or had spent at least five years in an English-speaking environment. Three of them were female and all of them were right-hand mouse users.

4.5.2 Apparatus

A Compaq KB-0133 Qwerty keyboard and an IBM 19″ CRT monitor at 1280×960 pixel resolution were used during the study. A custom Java application logged all keystrokes with timestamps during text entry and calculated user performance directly. The 460×500 application window was positioned at the center of the computer screen. A fifteen point Tahoma font was used to present text on the application. Figure 31 illustrates the study setup.

4.5.3 Procedure

During the study, participants entered short English phrases from a phrase set (MacKenzie and Soukoreff, 2003). The corpus does not contain any numeric and special characters, and all uppercase characters were converted to lowercase. It was selected because of its high correlation with the character frequency in the English language. Also, it is widely used in recent text entry studies. This makes this work comparable to others'. See Appendix A2 for more information on the set. The phrases were displayed to participants one at a time on the screen in a dialog. They were asked to take the time to read, understand, and memorize the phrases, to enter them as fast and accurate as possible, and to press the Enter key when they were done to see the next phrase. Timing started from the entry of the first character and ended with the last (the character before the Enter keystroke). Participants were informed that they could rest between conditions or before inputting a phrase.

Five injected system error rates $\boldsymbol{\rho}^{s}_{error}$ were tested: 1, 2, 5, 10, and 20%. In order to imitate system error, the keyboard's input system was altered to output a predetermined amount of error prone characters, proportional to one of the five mentioned rates. Although the amounts were predetermined, the actual errors were generated randomly by replacing the inputted character with the surrounding ones on a Qwerty layout. For example, the letter "H" was randomly replaced by one of the surrounding letters "Y", "U", "J", "N", "B", or "G". Similarly for all other keys. The injected system error conditions were presented to the participants in random order. Participants were informed prior to the study that the used keyboard is not 100% trustworthy and sometimes makes mistakes in interpreting the input. They were asked to work normally. That is, they should correct their errors as they notice them. They were also told that they could use any edit function, navigation key, or the mouse to correct errors. The system calculated the average entry speed (WPM), error rate (TER), output time ($\boldsymbol{T}_{output}$), and the cost of error correction ($\boldsymbol{T}_{fix}$). Recall that $\boldsymbol{T}_{output}$ is the sum of the time to correct an erroneous character by the user and the time to process and display the corrected input by the system, while $\boldsymbol{T}_{fix}$ is the *extra* time it requires on average per character to fix errors with a particular text entry technique.

4.5.4 Design

A within-subjects design was used. The within-subjects factor focused on five injected system error rates: 1, 2, 5, 10, and 20%. The dependent variables (and the metrics) were entry speed (WPM), error rate (TER), the output time ($\boldsymbol{T}_{output}$), and the cost of error correction ($\boldsymbol{T}_{fix}$). Section 2.3, 4.1.3, and 4.1.6 defined these

metrics. There were three segments. Each segment contained five blocks, representing the five injected system error rates. In each block, participants were asked to complete sixteen phrases, excluding two practice phrases. In each segment the blocks were presented randomly to minimize the effect of asymmetric skill transfer. In summary, the design was:

12 participants ×
3 segments ×
5 blocks (within-subjects factor: 1, 2, 5, 10, and 25% injected system error rates, randomized) ×
16 short English phrases
= 2,880 phrases, in total. Each participant entered 240 phrases.

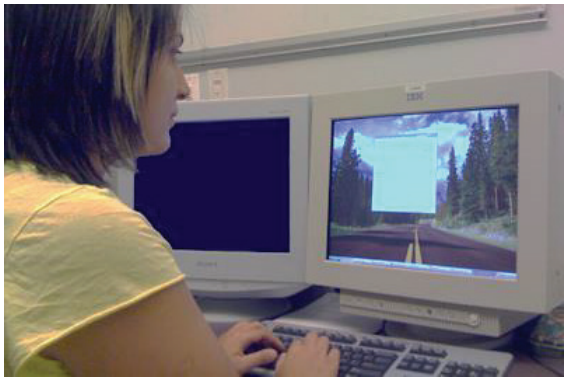

Figure 31. A participant inputting short English phrases during the user study.

4.5.5 Results

The whole user study lasted from 45 to 75 minutes including the demonstration, practice, and breaks. The highest and lowest average entry speeds were 13 and 93 WPM. Similar to the results reported in Chapter 3, participants used Backspace 99% of the time to correct their errors, even though they were able to use any edit operation, including keyboard shortcuts and the mouse.

D'Agostino Kurtosis tests on the dependent variables revealed that the data were normally distributed. Also, a Mauchly's test confirmed that the data's covariance matrix was circular in form. Thus, repeated-measures ANOVA was used for all analysis. The statistical tests used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%. All statistically significant results are presented with effect size (η^2) and power ($1-\beta$). See Appendix A1 and A4 for more information on η^2 and $1-\beta$, respectively.

4.5.5.1 Entry Speed and Error Rate

An ANOVA revealed that there was a significant effect of injected system error rate on both entry speed ($F_{4,11} = 86.05$, $p < .0001$; $\eta^2 = .42$, $1–\beta = 0.99$) and error rate ($F_{4,11} = 787.61$, $p < .0001$; $\eta^2 = .94$, $1–\beta = 1.0$). A Tukey-Kramer test showed that the 10 and 20% injected system error rates had significantly lower entry speed and higher error rate compared to the 1, 2, and 5% conditions. As a reference, an average entry speed of 57.78 WPM (SD = 20) was recorded for text entry in a pilot study without injected system errors (0%). This is higher than the performance levels for 1 and 2% injected system errors, but not significantly so ($t_2 = 17.12$, $p = 0.25$). Figure 32 and Figure 33 show the average entry speed (WPM) and error rate (TER), respectively, for all conditions.

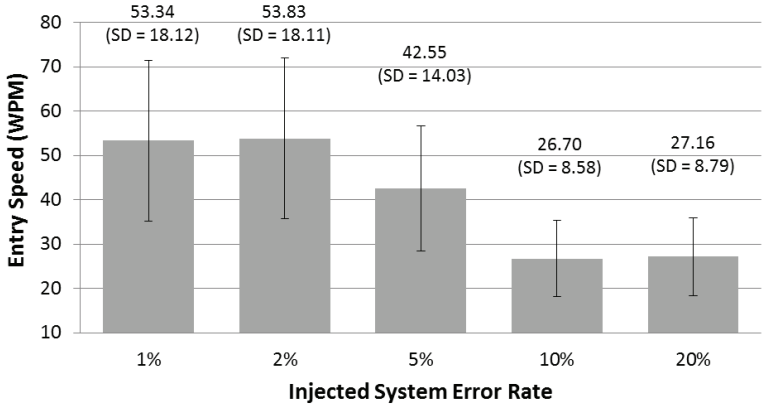

Figure 32. Average entry speed (WPM) for all injected system error rates (ρ^s_{error}). Error bars represent ±1 standard deviation (SD).

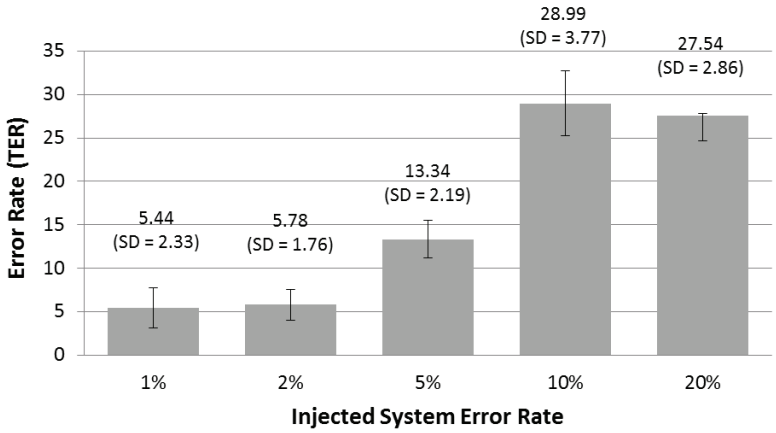

Figure 33. Average error rate (TER) for all injected system error rates (ρ^s_{error}). Error bars represent ±1 standard deviation (SD).

4.5.5.2 The Output Time

An ANOVA on the data identified a significant effect of injected system error rate on the output time $\boldsymbol{T}_{output}$ ($F_{4,11}$ = 15.54, p < .0001; η^2 = .06, $1–\beta$ = 0.04). A Tukey-Kramer test showed that the 10 and 20% injected system error rates had significantly higher output time compared to 1, 2, and 5%. Figure 34 illustrates the average output time $\boldsymbol{T}_{output}$ for all conditions.

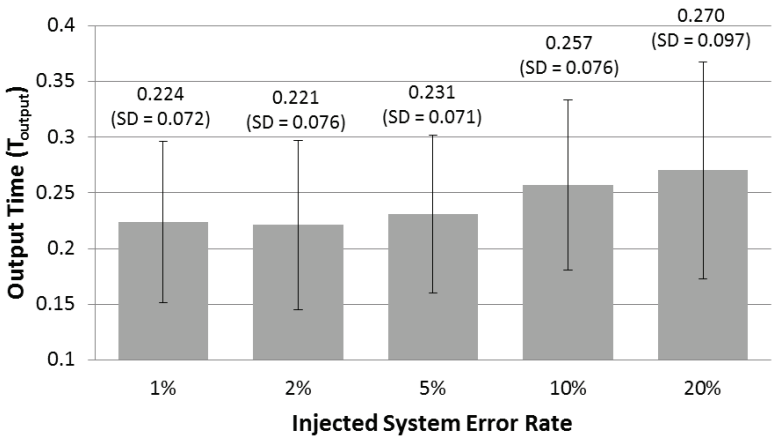

Figure 34. Average output time (T_{output}) for all injected system error rates (ρ^{s}_{error}). Error bars represent ±1 standard deviation (SD).

4.5.5.3 (Injected) System Error Analysis: Empirical Validation

The data corresponds well to the new model's primary prediction: the cost of error correction increases more or less linearly as the probability of a system error increases. Figure 10 visualizes this relationship. There, one can see that the study data fits a linear function reasonably well, with R^2 = 0.9229. An ANOVA revealed that there was a significant effect of injected system error rate on the cost of error correction ($F_{4,11}$ = 1108.42, p < .0001; η^2 = .02, $1–\beta$ = 0.01). Figure 35 illustrates the predicted and observed increase in the cost of error correction ($\boldsymbol{T}_{fix}$) as the system error rate increases.

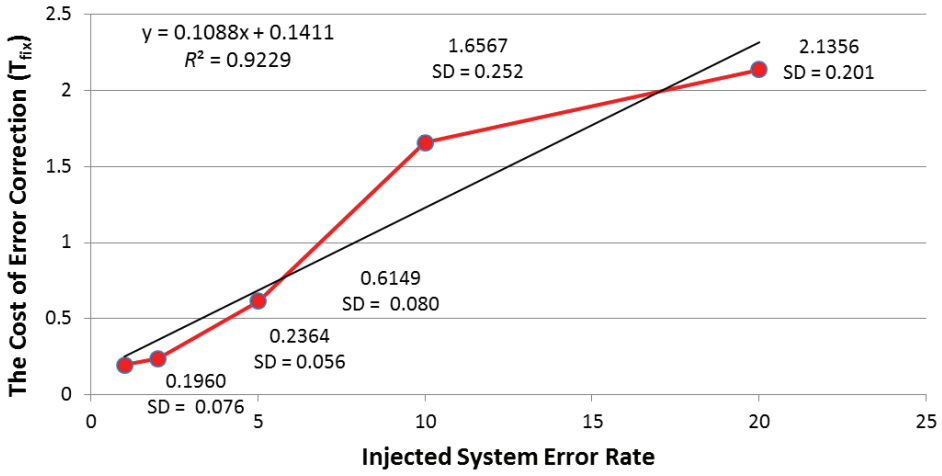

Figure 35. The increase in the cost of error correction (T_{fix}) as the probability of system error (ρ^{s}_{error}) increases. Error bars represent ±1 standard deviation (SD).

4.5.6 Discussion

The results of the study support acceptance of the hypothesis $H_{1\,C4}$ (see Section 4.5). The results match the nature of predictions of the model—the data fits a linear approximation reasonably well. See Figure 36. It is unclear why error fixing efforts were higher for the 10% value. One potential explanation is that participants may have treated both the 10% and 20% injected error rate conditions in the same way—"just an unreliable system". It is not possible to directly compare the data from this study with the initial data source presented in Chapter 3, as the average entry speed was substantially higher in the previous study (76 WPM) due to the screening for participants with high text entry speeds.

It is interesting to see that low injected system error rates (1% and 2%) had no significant effect, even though the text entry performance was somewhat lower (see Figure 33). The most probable reason is that such low error rates are indistinguishable from the average human error rates (1.8% for experienced users, see Table 1 in Section 2.4). This can be considered as an indication that keyboard failure rates of 1-2% are somewhat acceptable and have only a small negative effect on human performance, in the order of 7 to 8%. However, an error rate of 5% (95% reliability) yields a noticeable 26% drop in performance. A reliability of 80%, respectively 90%, approximately halves the entry speed (see Figure 32). This underlines how important reliability is for text entry techniques.

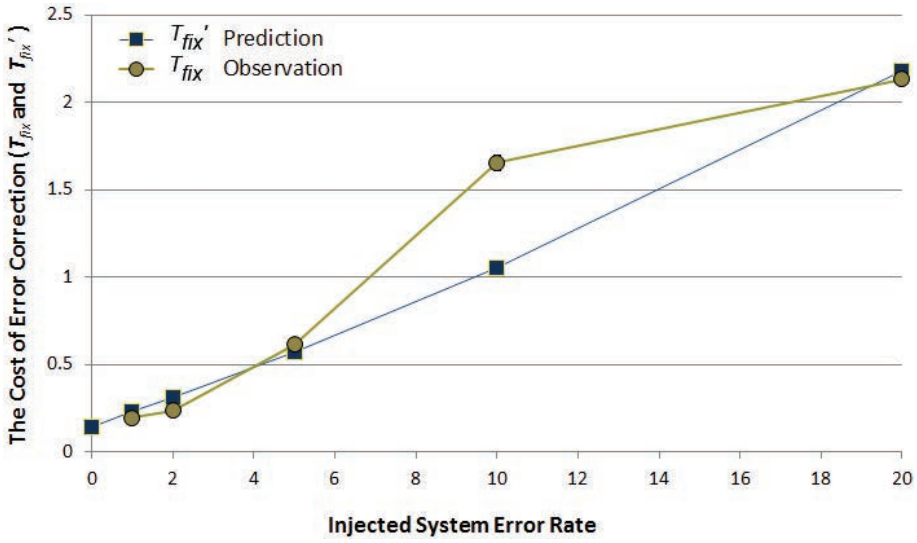

Figure 36. The increase in predicted T_{fix}' and observed T_{fix} as the probability of system error ρ_{error}^{s} increases.

4.5.7 Generalization to Other Techniques

Theoretically, the new model and the predictions it generates are directly applicable to other character-based keyboards and mobile keypad. If one can assume 100% reliable keys, only a derivation of the value of $\boldsymbol{T}_{correct}$ for each distinct technique in necessary.

Another potential application area for this work is virtual keyboards whose keys are too small to be hit reliably with a human finger, because the buttons are much smaller than the fingertip. There are currently many mobile phones that employ touchscreens together with small screen sizes. Due to the lack of tactile feedback, such techniques are likely fundamentally different from mini-Qwerty keyboards. One could then model the ratio of the size of a fingertip relative to the displayed button size as a measure of keyboard reliability. With this, it may be possible to predict the effect of varying button sizes in virtual keyboards. Thus, it should be possible to predict some of the results of a recent evaluation (MacKenzie, 2002) of touchscreen keyboards, assuming $\boldsymbol{T}_{correct}$ has been characterized.

Table 5. Detected effect size and measured statistical power.

Dependent Variable	Effect Size (η^2)	Power ($1-\beta$)
WPM	Large	> 0.80
TER	Large	> 0.80
$\boldsymbol{T}_{output}$	Medium	<< 0.80
$\boldsymbol{T}_{fix}$	Small	<< 0.80

4.5.8 Limitations

The ANOVAs identified a significant effect of injected system error rate on entry speed (WPM), error rate (TER), output time ($\boldsymbol{T}_{output}$), and the cost of error correction ($\boldsymbol{T}_{fix}$). A post-hoc analysis detected a *large* effect size for WPM and TER, a *medium* effect size for $\boldsymbol{T}_{output}$, and a *small* effect size for $\boldsymbol{T}_{fix}$ (see Appendix A1). Further post-hoc analysis revealed that the statistical power exceeded the 0.80 threshold (see Appendix A4) at the observed *small* to *large* effect size levels for all dependent variables, except for $\boldsymbol{T}_{output}$ and $\boldsymbol{T}_{fix}$, see Table 5. In other words, there was less than adequate statistical power for $\boldsymbol{T}_{output}$ and $\boldsymbol{T}_{fix}$. While a larger sample size (N) might be necessary to achieve a sufficiently strong statistical power for $\boldsymbol{T}_{output}$, due to the *medium* effect size, there is a possibility that $\boldsymbol{T}_{output}$ does not have a sufficiently strong effect after all. Alternatively, and due to the *small* effect size, it is less likely that a larger sample size (*N*) would achieve a sufficiently strong statistical power for $\boldsymbol{T}_{fix}$. Appendix A4 elaborates on the criteria used for calculating statistical power.

As the study recruited participants by using convenience sampling from the university community, the results may not generalize to a larger population. Nevertheless, an attempt was made to counteract this confound by recruiting not only university students but also instructors and staff.

4.6 Summary

This chapter investigated human error behavior in character-based text entry. Based on the transcription typing phenomena listed in Section 2.5 and the behaviors observed in Chapter 3 user study, a new model for predicting the cost of error correction is proposed. The model is verified against values derived from the literature and by conducting a user study.

The model predicted, and later the results of the study verified, that users' text entry performance decay as the system becomes more and more unreliable (error prone). This phenomenon may make one wonder, do users adapt to faulty systems (or to the errors) to improve their overall text entry performance? The following chapter attempts to answer this.

Chapter 5

Adapting to a Faulty Unistroke Gesture Recognizer

Although the origin of the famous quote, "*That's not a bug, it's a feature*" is unknown (Hafner, 2008), the computer science community is well aware of its implications. It points towards a phenomenon observed in many paradigms, including human-computer interaction, and programming languages. The quote refers to the possibility that practitioners will adapt to a non-fatal bug or system error if it remains in the system for long enough. Once users get accustomed to a system error they either actively avoid repeating actions that cause the error or start treating it as a feature. Such behavior can be indirectly explained through theories of learning, as explained in Section 1.1 and Section 2.6. Regardless of the detailed explanation, these theories imply that it is important to reduce mistakes to learn correct responses.

Section 2.2.8 provided a review of the well-known academic and commercial gesture-based techniques, gesture recognition approaches, and the challenges these techniques face. Briefly, it has been established that for both touchscreens and digital pens, current gesture-based techniques are error-prone, primarily due to imperfections in the underlying technologies (Mankoff and Abowd, 1999; Shilman et al., 2006). Recent works mostly focus on improving recognition accuracy by developing new recognition algorithms or by limiting drawing variations, such as permitting only a single way to draw a gesture. Several works focus on user tolerance; that is, how error prone a system has to be for the users to abandon it, also discussed in Section 2.2.8. However, not much work has been done on how users interact with an error prone gesture recognizer that frequently misrecognizes gestures. Do they adapt to the recognition errors? Is there a relationship between recognition error rates and the rate at which users adapt to these errors? Answers to these questions are vital as these may provide designers with guidance for future work on such technologies.

Based on the observations from several pilot studies, reported outside of this thesis (Arif and Stuerzlinger, 2012), this work hypothesizes that users gradually adapt to *misrecognition* errors and that this adaptation rate depends on how frequently such errors occur. That is, users adapt to such an error faster if it occurs more frequently. In an attempt to verify this hypothesis, this chapter presents the results of two user studies. It also speculates on the practical implications of this work.

5.1 The Custom Software

This section discusses the custom software used during both user studies. The software was designed in accordance with current trends in human and system error handling as well as the provision of alternative gesture sets for gesture-based techniques, as explained in Section 2.6.7. The software was also fine-tuned through several pilot studies reported elsewhere (Arif and Stuerzlinger, 2012). The intention was to increase the external validity of this work by making the experiment software reasonably comparable to existing systems.

5.1.1 Gesture Recognition

The application used the $1 Unistroke Recognizer to process pen-based gesture input (Wobbrock et al., 2007). This recognizer was designed for rapid prototyping of gesture-based user interfaces. It recognizes gestures using a nearest-neighbor classifier with a Euclidean scoring function, similar to a geometric template matcher. This approach is different from the original Unistrokes and Graffiti gesture recognition strategies (see Section 2.2.8). The $1 recognizer was used because, first of all, it is easy to deploy. Second, the focus here is on how users adapt to (injected) *misrecognition* errors and not (directly) on the underlying recognition strategies. And, finally, it performs well for a limited number of gestures based on very few templates. A user study reported 99% accuracy for sixteen gestures with three or more loaded templates (Wobbrock et al., 2007). The custom application developed here used fourteen gestures and loaded seven templates for each, which should make the system perform equivalent to other recent recognizers in the field. Also, see Section 5.1.3.

5.1.2 Supported Gestures

During the studies, participants inputted seven English letters, specifically “B”, “D”, “O”, “Q”, “R”, “W”, and “Y”. The custom software presented one letter at a time on the screen. Participants then had to input the presented letter with the pen on a graphic tablet using either Graffiti or Unistrokes. The system used Graffiti as the primary method of inputting the letters, while the Unistrokes were used as alternatives. That is, users were expected to primarily use Graffiti to input the letters, but were permitted to use Unistrokes if they felt their use necessary; i.e., to bypass (injected) *misrecognition* errors. Section 5.1.4.1 elaborates on this.

5.1.2.1 Unistroke vs. Multistroke Gestures

A unistroke gesture system was used instead of a multistroke one, as the latter systems usually permit different approaches for drawing the same letter. This makes it more challenging to recognize a performed gesture. It also makes it difficult to automatically identify human errors due to variations in gesture drawing across users. Section 2.2.7 elaborated on these aspects. Besides, due to multiple possible drawing variations for the same letter, users often struggle to identify their mistakes and to discover the right way for drawing a letter with multistroke systems. The $N Recognizer, for example, often fails to correctly recognize a gesture when users use more strokes than the number of strokes used to define said gesture (Anthony and Wobbrock, 2012). With adaptive multistroke recognizers, such as Gesture Search (Li, 2010[a]), it is difficult to isolate the human adaptation rate as the system adapts to human behaviors as well. Unistroke gesture systems usually do not suffer from such problems (Tappert and Cha, 2007). A more recent multistroke recognizer, $P Recognizer, resolves these issues (Vatavu et al., 2012), but was proposed after the completion of these studies.

5.1.2.2 Primary vs. Alternative Gestures

Graffiti and Unistrokes were selected as primary and alternative method for drawing the letters for two reasons. First, a longitudinal study did not find any significant difference between these techniques' entry speed, correction rate, and preparation time (Castellucci and MacKenzie, 2008). Second, Graffiti was selected as the primary method, as in almost all unistroke-based techniques the primary method is relatively more intuitive and easier to guess than the alternative one, see Section 2.6.7.1. The above-mentioned study reported that users find Graffiti more intuitive than Unistrokes due to the gestures' resemblance to their corresponding English letters. In Figure 37, one can see how the primary Graffiti gestures look like their printed counterparts. In addition, participants were encouraged to practice the primary gestures before the main studies, to familiarize participants (to a limited degree) with Graffiti, see Section 5.2.3. With this experimental design one can assume that any performance effect due to switching the gesture drawing method (from primary to alternative and vice versa) mid-study will be predominantly attributable to adaptation.

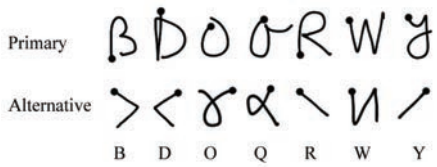

Figure 37. The seven letters and their corresponding gestures. The primary ones (above) are from Graffiti letter set, while the bottom ones (the alternatives) are from Unistrokes. Here, a dot indicates the start of a stroke.

5.1.2.3 Discoverability

Section 2.6.7.1 mentioned how in most gesture-based techniques alternative input methods are relatively harder to discover compared to the primary method. To discover alternative gestures with such techniques, one has to either go to an extended tutorial or guess. Following this, the custom software displayed the primary gestures in a panel at all times and presented the to-be-inputted letters in Graffiti. To discover an alternative gesture for a particular letter, users had to tap or right-click on the corresponding primary gesture in the panel. This displayed the alternative gesture for that letter for two seconds, and then returned to the original state, that is, displayed the primary gestures. Figure 38 illustrates this.

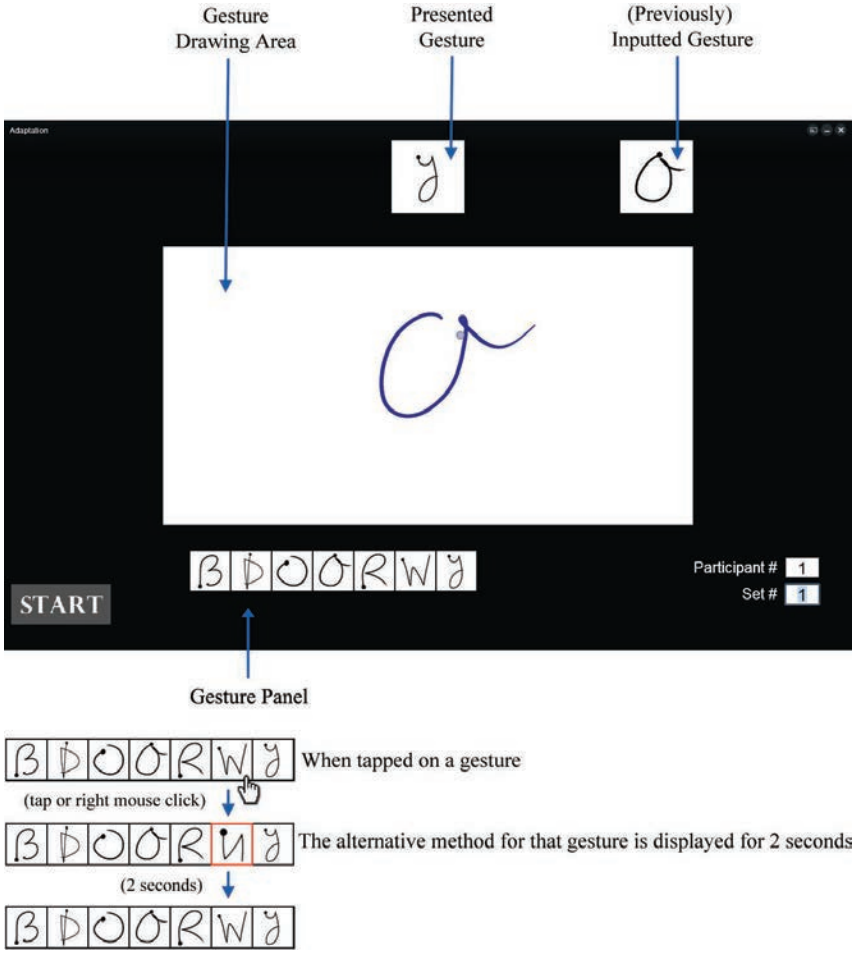

Figure 38. The custom software used during the studies. The to-be-inputted letter is presented using the primary gesture. To discover the alternative method for that letter one has to tap on the corresponding primary gesture in the bottom panel.

5.1.3 Errors and Error Handling

Section 2.6.7 discussed that two types of errors occur in most gesture-based techniques: *failure to recognize* error and *misrecognition* error.

Similar to other gesture-based systems and based on pilots, the system reported a *failure to recognize* error when the total number of recorded candidate points (x and y coordinates) was less than ten; i.e., when the stroke was much too short to be a gesture. Results of the pilots revealed that such gestures are almost always caused by accidental interactions. Examples are that users exited drawing prematurely, tapped on the graphic tablet with the pen, or pressed the buttons on the pen by mistake. However, this threshold may be different for different devices as the rate in which candidate points are recorded are dependent on the sensing hardware and input software. Similar to many gesture-based techniques, as described in Section 2.6.7, the custom software provided visual feedback on *failure to recognize* errors. The inputted gesture field, in top-right corner of Figure 38, displayed a special symbol in case of accidental interactions. Figure 39 shows this symbol.

Figure 39. The special symbol displayed in the inputted gesture field in case of accidental interactions.

A *misrecognition* error was identified when the recognized gesture did not match the presented gesture. Similar to almost all gesture-based systems, the custom software displayed the misrecognized gesture in the inputted gesture field. For example, when "O" was misrecognized as "Q", the system displayed "Q" in the inputted gesture field. In addition, auditory feedback was provided on both instances. That is, the system made a "ding" noise when it identified a *failure to recognize* or *misrecognition* error.

5.1.3.1 Raw Recognition Error Rate

In a pilot study with eight novice users (four female, average 21 years, all right-handed), where each user inputted the seven Graffiti gestures (see Figure 37) for forty times with the custom software ***without error injection***, the system recorded 0.3% *failure to recognize* and 0.7% *misrecognition* errors; i.e., 1% system error rate. In other words, the overall accuracy rate was 99%, which matches the gesture recognition performance of prior work (Wobbrock et al., 2007).

5.1.4 Injected Misrecognition Errors

The main purpose of this work is to investigate (whether and) how users adapt to *misrecognition* errors. Thus, a few primary gestures were randomly selected during the user studies and injected with synthetic *misrecognition* errors at different rates. That is, the system intentionally misrecognized these gestures at the given rates. For instance, if the primary gesture for "D" was injected with 5% synthetic *misrecognition* error, then five out of hundred times the system would intentionally misrecognize this gesture and would randomly display a similar gesture in the inputted gesture field, such as "B", "C", "O", or "Q". The system injected *misrecognition* error instead of *failure to recognize* error, as misrecognition is the most common type of error in gesture-based techniques (see Section 2.6.7). ***Only*** the primary gestures were injected with these errors.

Any potential bias in simulated gesture recognition errors was accounted for by randomly selecting a different set of letters for error injection for each participant. Another design constraint for the user studies is that with increasing gesture set size, error occurrences naturally decrease, which makes such errors then progressively harder to study. Consequently, the studies used only seven letters and the implementation used well-tuned gestures. As mentioned above, in the absence of injected errors, users encountered only 1% "system" errors. Such a small error rate is well below what can be studied in short-term studies. Consider that an error rate of 1% means that system errors occur only once for every hundred such letters entered. In the reported studies, participants entered 630 gestures within an hour or more. Thus they would see only 6-7 errors, which is too small to study adaptation.

5.1.4.1 Bypassing Injected Errors

Findings from pilot studies reported elsewhere (Arif and Stuerzlinger, 2012) indicate that users attempt to bypass *misrecognition* errors in two different ways. They either draw a frequently misrecognized gesture relatively slowly or start using an alternative method (if available) for drawing such a gesture. The first approach affects one's entry speed, as it takes more time to input gestures than the usual. In contrast, the second does not compromise entry speed, assuming that the alternative method is not more complex than the primary. Thus, the latter approach is a better choice for experiments, as entry speed will vary less. Tu et al. (2012) provides a methodology for classifying gestures into *simple*, *medium*, and *complex* categories.

Besides, with only a single "faulty" gesture set (and no alternatives), users are effectively stuck. If they fail to recognize the failure patterns, they need to adapt or, failing that, can only abandon the system. In many

real world situations, they would most probably abandon the system, as there are other ways to achieve their tasks. Consequently, many recent real world systems, see above, include gesture variations (alternative gestures) as a way to address this problem. To keep the results externally valid, this work chose to provide alternative gestures.

To address the issues around speed, the studies ***indirectly*** discouraged participants from drawing gestures slowly. First, users were informed prior to the studies that taking more time to draw a gesture might not enhance the system's recognition rate. Second, and in the practice period prior to the main studies, see Section 5.2.3, most users would realize that an inputted gesture does not have to be an exact match of the displayed one for the system to recognize it—so that (subconsciously) they would be much less motivated to draw gestures slowly.

5.1.5 The Seven Letters vs. Short English Phrases

Early on, a decision was made against the use of short English phrases in the studies. Two reasons motivated this. First, using English phrases would require injecting recognition errors based on letter frequencies to maintain uniformity. This needlessly complicates and lengthens the studies. Second, a pilot showed that inputting English phrases with an untrustworthy gesture-based system causes a high level of user frustration, which may negatively bias study results.

Similarly, a decision was made against using a complete gesture alphabet. The reason is that users need to experience *enough* injected *misrecognition* errors during the study duration (60-90 minutes) to be able to adapt to the system. This is vital, as a 5% injected error rate means that users will *only* face five injected errors in one hundred attempts. The use of seven letters assured that each letter appeared for a sufficient number of times. This does not invalidate this work, as the focus here is on how users adapt to (injected) *misrecognition* errors and not (directly) on how the overall text entry performance is affected.

5.1.6 Justification for a Short-term Study

While it is important to understand gradual adaptation over time, short-term usability is today a strong determinant in product success. If users do not see reliable enough performance in the short term, a product is likely to fail. Consequently, long-term investigations are interesting, but do not help in situations where users get frustrated up-front. This is a global issue that gesture recognizers have to contend with today.

5.1.7 Performance Metrics

The following metrics were calculated during the studies.

- **Alternative Method Usage (AMU):** The rate (%) at which the alternative method was used to input letters. As users were free to use either the primary or the alternative method to input/re-input a letter, this metric enable us to measure the rate at which users adapted to the alternative gestures.
- **Input Time (T^{h}_{input})**: This represents the average time (in milliseconds) it took to input a letter. This metric captures the performance aspect of learning. We also use this to analyze performance across different *misrecognition* rates. Input time was also discussed in Section 4.1.1.
- **Gestures per Character (GPC):** This denotes how many gestures it took on average to input a letter (Wobbrock et al., 2003). As most unistroke methods have dedicated gestures for all English letters, a flawless system will require a GPC of one, providing there was no human error. This was calculated to provide an overall picture of the input process, and to check whether the more faulty letters yield higher GPCs compared to less faulty ones, as one might expect.

5.2 User Study 1

This study investigated users' adaptation behavior for injected *misrecognition* error rates from 0 to 30%. In other words, the study tested the following hypothesis:

($H_{1\ C5}$) *While inputting letters with a unistroke-bases system, users adapt to frequently misrecognized letters by replacing the primary gesture for inputting those letters with the alternative ones (if available) at rates relative to those letters' misrecognition rates, providing that improving drawing quality does not improve its recognition rate.*

5.2.1 Participants

Twelve participants, aged from 21 to 30 years, average 25, participated in the study. Appendix A3 explains the procedure used to decide the number of participants (sample size). They were recruited through online social communities, local university e-mailing lists, by posting flyers on campus, and by word of mouth (convenience sampling). None of them had prior experience with pen-based devices. They were also unaware

of the existence of Unistrokes and Graffiti. Seven of them were female and one was a left-hand pen user. They all received a small compensation (CAD 10.00) for their participation.

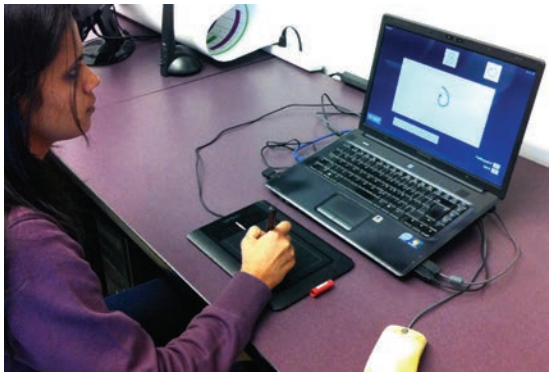

Figure 40. A participant drawing gestures using a digital pen on a Bamboo Pen & Touch Graphic Tablet.

5.2.2 Apparatus

The custom application described in Section 5.1 was used during the study. It was developed with the default Bamboo Mini SDK 2.1. The application was displayed on a 15.4″ Compaq Presario C700 Notebook PC at 1280×800 pixel resolution. Participants interacted with the application through a digital pen on a Wacom Bamboo Pen & Touch Graphic Tablet, as illustrated in Figure 40. The device's 14.73×9.14 cm active area was calibrated with respect to the application window. Its multi-touch input capability was disabled to permit participants to rest their hands on the surface while using the pen. The orientation of the tablet and the default firmware was adjusted to accommodate for left- and right-handedness. The custom application logged all interactions with timestamps and calculated user performance directly.

5.2.3 Procedure and Design

The experiment setup and software was first demonstrated to users. The experimenter verified that participants understood the primary (Graffiti) and the alternative (Unistrokes) gestures, the *failure to recognize* and the *misrecognition* errors (see Section 5.1.3), and knew how to discover alternative gestures (see Section 5.1.2.3).

A practice period followed the demonstration. During practice, participants were asked to input the seven letters five times using the primary method without error injection. The intent was to familiarize them with the setup. This also gave them some experience with how similar the presented and the performed gestures

needed to be for the system to recognize them accurately. Participants were able to extend the practice period (at most twice), as desired.

The main user study started roughly two minutes after the practice. In that part, participants inputted letters in random order and each of the seven letters occurred ninety times. Thus, each participant inputted in total 630 letters. Three out of the seven letters were randomly picked by the system and injected with 10, 20, and respectively 30% *synthetic misrecognition* errors (see Section 5.1.4). That is, in ten, twenty, and thirty out of hundred attempts the corresponding letters were intentionally misrecognized by the system. That is, the system displayed a similar letter instead of the recognized one, as discussed above. Only three letters were injected with synthetic *misrecognition* errors, to ensure that the faulty letters do not dominate the overall input process.

The letters were displayed one at a time on the screen. Participants had to input each presented letter using the pen and the graphic tablet using predominantly the primary method (Graffiti). They were informed that, unlike in the practice period, the system might not be entirely reliable. That is, it may misrecognize some of the letters, even when they were inputted correctly. However, they were not informed about error rates or the number of letters where synthetic *misrecognition* errors were injected.

A gesture was recorded from the moment one touched the graphic tablet with the pen (touch-down) to the moment it was lifted (touch-up). Upon completion of input, the recognized and the next to-be-inputted letters were displayed on the screen automatically, as illustrated in Figure 38. Participants were asked to input the gestures as fast as possible, but to focus more on the accuracy. That is, they were encouraged to reduce the *misrecognition* errors, any way they saw fit, even if it compromised their input speed. They were informed that they ***could*** use the alternative method (Unistrokes) to input a frequently misrecognized letter, if they felt that this would improve (or is improving) recognition accuracy. But they were ***neither forced nor instructed*** to use the alternatives. Users had to keep inputting a gesture until it was correctly recognized by the system. On correction attempts, ***no*** synthetic recognition errors were injected to reduce the potential for overly frustrating tasks. Thus, users who did not want to use alternatives could use the primary method on correction attempts. Auditory and visual feedback was provided, as described in Section 5.1.3. To minimize interruptions, participants were permitted to take at most two three-minute breaks during the study, as necessary. Given that participants entered 630 letters in the whole session, this gave them enough time to create a good mental model of the system and its errors. After all, each participant the set of faulty letters was constant for each participant. Upon completion of the study, they were asked to fill out a short questionnaire, where they were asked to list the frequently misrecognized letters.

Note that the focus of the study was to investigate whether users adapt to the injected *misrecognition* errors by bypassing them through usage of the alternative input method. Thus, attempts were made to ensure that the memorization of the primary method was not absolutely necessary. Towards this, the primary gesture set was displayed at all times. See Figure 38 for a screenshot. Also, participants practiced only the primary method before the study. In contrast, to discover the alternative gesture for a letter during the study, users had to tap or right-click on its corresponding primary gesture in a panel. This displayed the alternative method for inputting that letter for two seconds, and then returned to the original state, see Figure 38.

The user study used a within-subjects design, where the within-subjects factor focused on 0, 10, 20, and 30% injected *misrecognition* error rates. Twelve participants inputted the seven letters (see Figure 37), ninety times each, using the primary method (Graffiti). However, they were permitted to use the alternative method (Unistrokes) to input the frequently misrecognized letters in an attempt to improve recognition accuracy. Each participant inputted in total 630 letters. The dependent variables (and the metrics) were GPC, AMU (%), and $\boldsymbol{T}_{input}^{h}$ (ms). Section 5.1.7 defined these metrics.

5.2.4 Results

The whole study lasted from sixty to ninety minutes including the demonstration, practice, and breaks. Upon completion of the study, 59% participants were able to recognize all three error prone letters, 33% had recognized the two most error prone letters, while the remaining 8% recognized only the most error prone one.

D'Agostino Kurtosis tests on the dependent variables revealed that the data were normally distributed. Also, a Mauchly's test confirmed that the data's covariance matrix was circular in form. Thus, repeated-measures ANOVA was used for all analysis. The statistical tests used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%. All statistically significant results are presented with effect size (η^2) and power ($1-\beta$). See Appendix A1 and A4 for more information on η^2 and $1-\beta$, respectively.

To identify learning, the study data was segmented into blocks of ten appearances of each letter during the study. That is, the average of every ten times a letter was presented to the users to input was used to observe improvements over time. As all letters appeared exactly ninety times per participant, there were nine segments for each letter.

5.2.4.1 Alternative Method Usage (AMU)

An ANOVA on the data revealed that there was a significant effect of injected *misrecognition* error rate on AMU ($F_{3,11} = 5.56$, $p < .005$; $\eta^2 = .40$, $1-\beta = 0.97$). Average AMU for 0, 10, 20, and 30% injected *misrecognition* error rates were 8.5, 31.85, 27.59, and 55.19%, correspondingly. Figure 41 illustrates this. A Tukey-Kramer test showed that the 30% injected *misrecognition* error rate was significantly higher AMU than 0, 10, and 20%.

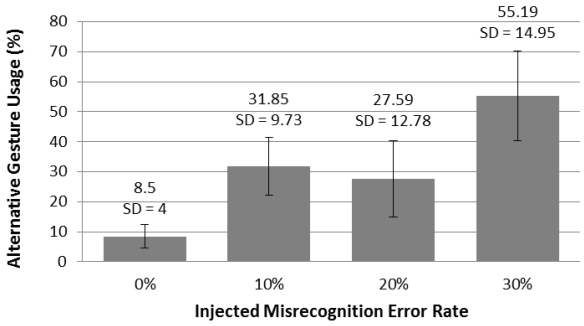

Figure 41. Average Alternative Method Usage (AMU) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation (SD).

For all injected misrecognition error rates, power functions were fitted to the data to model the power law of practice (Card et al., 1983). This is illustrated in Figure 42, where the horizontal axis represents the segments (see Section 5.2.4) and the vertical axis represents the average AMU during that segment. Recall that there were four letters where no *misrecognition* errors were injected (0%), compared to one letter for each injected *misrecognition* error rates (10, 20, and 30%). Therefore, for better representation, the 0% data points average the AMU of the ten appearances of the four non-faulty letters (10×4 appearances). An attempt to fit linear functions to the data was also made (0%: $R^2 = 0.87334$, 10%: $R^2 = 0.63659$, 20%: $R^2 = 0.95331$, and 30%: $R^2 = 0.51826$), but they did not correlate as well as power functions (0%: $R^2 = 0.92358$, 10%: $R^2 = 0.84447$, 20%: $R^2 = 0.96004$, and 30%: $R^2 = 0.73612$).

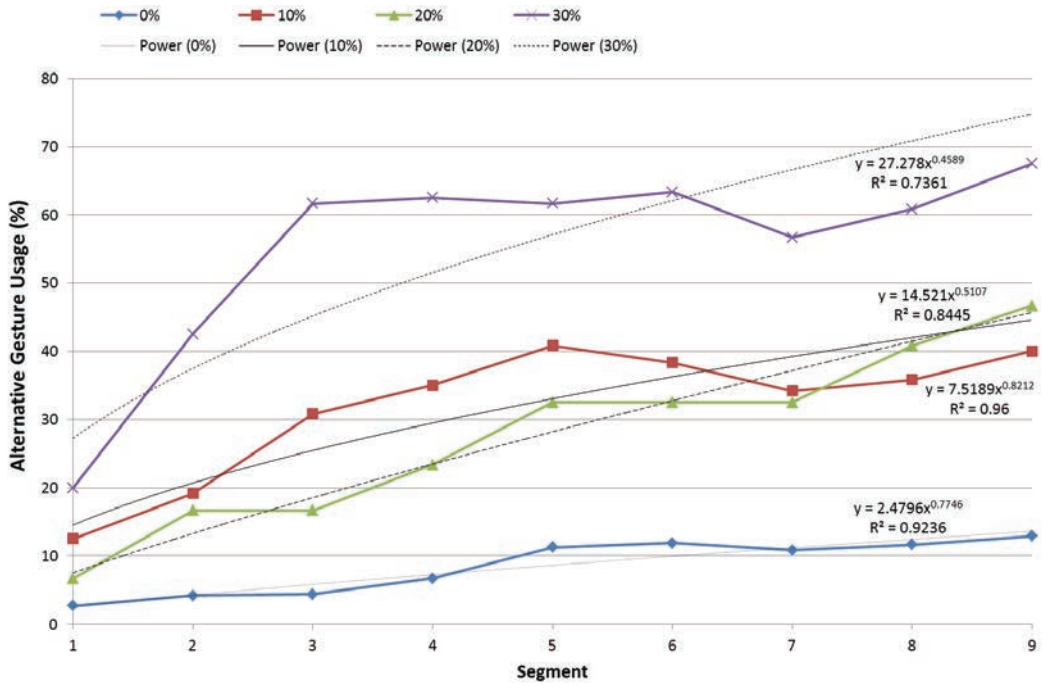

Figure 42. Average Alternative Method Usage (AMU) by injected misrecognition error rates and segments.

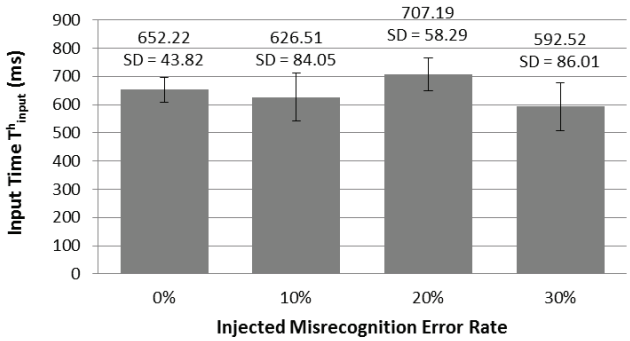

Figure 43. Average Input Time (T^h_{input}) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation (SD).

5.2.4.2 Input Time (T^h_{input})

There was global learning, as the average time over all letters to input a gesture, $\boldsymbol{T}^h_{input}$, correlated well with the power law of learning (Card et al., 1983), over all letters ($y = 750.07x^{-0.109}$, $R^2 = 0.7564$). An ANOVA on the data failed to identify a significant effect of injected *misrecognition* error rate on $\boldsymbol{T}^h_{input}$ ($F_{3,11} = 1.68$, $p > .05$). $\boldsymbol{T}^h_{input}$ for 0, 10, 20, and 30% injected *misrecognition* error rates was 652, 627, 707, and 593 milliseconds, respectively. Figure 43 illustrates this.

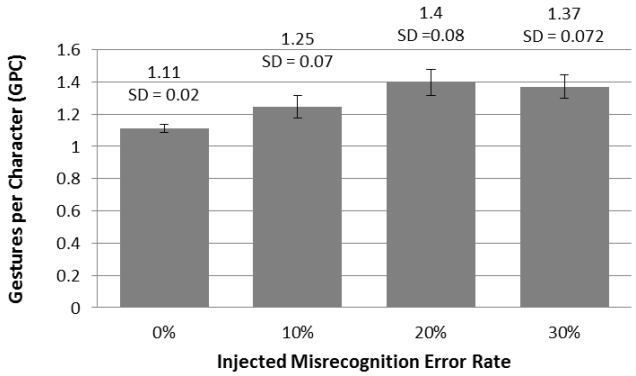

Figure 44. Average Gestures per Character (GPC) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation (SD).

5.2.4.3 Gestures per Character (GPC)

An ANOVA identified a significant effect of injected *misrecognition* error rate on GPC ($F_{3,11} = 4.39$, $p < .05$; $\eta^2 = .20$, $1–\beta = 0.81$). A Tukey-Kramer test revealed that 30 and 20% injected *misrecognition* error rates yielded significantly higher GPCs compared to 0 and 10%. Average GPC for 0, 10, 20, and 30% injected *misrecognition* error rates were 1.11, 1.25, 1.4, and 1.37, respectively, as illustrated in Figure 44. However, the data over all letters did not correlate well with the power law of learning (Card et al., 1983), ($y = 1.2638x^{0.0092}$, $R^2 = 0.1001$).

5.2.5 Discussion

The results of the study support acceptance of the hypothesis $\boldsymbol{H_{1\,C5}}$ (see Section 5.2). The results show that the use of the alternative method increased over time. Figure 42 illustrated average AMU by injected *misrecognition* error rates and segments, where one can see that participants learned to use the alternative method to input those letters where synthetic *misrecognition* errors were injected relatively faster compared to the reliable letters. A Tukey-Kramer test showed that the alternative method was used substantially more frequently for the most faulty letter (30% injected *misrecognition* error rate) compared to the less faulty ones (0-20% injected *misrecognition* error rates). This verifies the hypothesis that users adapt to a gesture-based technique's misrecognition errors and that this adaptation rate depends on how frequently they occur. That is, users adapt to an error faster if it occurs more frequently.

There was no significant effect of injected *misrecognition* error rate on $\boldsymbol{T}^h_{input}$. Instead, participants learned to input *all* letters faster with time, despite the injected *misrecognition* error rates. This verifies the

assumption discussed in Section 5.1.2.2 that switching input methods mid-study would not affect entry speed in a significant manner. This also validates the decision of using Graffiti and Unistrokes as the primary and the alternative methods, respectively.

One interesting trend visible in Figure 42 is that users adapt to the 10 and 20% injected *misrecognition* error rates roughly the same way, while adaptation to 0 and 30% is quite distinct. One could speculate that this is because users perceive 10 and 20% injected *misrecognition* error rates almost the same way, while 30% was perceived as *too* error prone. User feedback data also supports this, as most users responded that they were only able to differentiate between the 10 and 20% injected *misrecognition* error rates towards the end of the study. This behavior is similar to the results on text entry on a faulty keyboard, presented in Chapter 4, where 10 and 20% were also not found to be significantly different.

There was a significant effect of injected *misrecognition* error rate on GPC. Evidently, 30 and 20% injected *misrecognition* error rates yielded significantly higher GPCs compared to 0 and 10%. This is not unexpected as error correction was forced during the study. Therefore, participants often had to make multiple attempts to input the letters where synthetic misrecognition errors were injected. This is also apparent in Figure 44, where one can see the increase in average GPC with increasing injected *misrecognition* error rates.

To further observe user adaptation to injected *misrecognition* error rates and to investigate whether the results of this study apply to relatively lower rates or not, a second user study was conducted.

5.3 User Study 2

This study investigated users' adaptation behavior for injected *misrecognition* error rates from 0 to 10%. It also tested hypothesis $\boldsymbol{H_{1\,C5}}$, as the previous user study, see Section 5.2.

5.3.1 Participants

Twelve participants, aged from 18 to 34 years, average 24, took part in the study. Appendix A3 elaborates on the procedure used to determine the sample size; i.e., the number of participants. They were recruited through online communities, local university e-mailing lists, posting flyers on campus, and by word of mouth (convenience sampling). None of them had prior experience with pen-based devices and eleven of them had no knowledge of Unistrokes and Graffiti. One knew about these techniques, but had never used

them. Six of them were female and one was a left-hand pen user. They all received a small compensation (CAD 10.00) for their participation.

5.3.2 Apparatus, Procedure, and Design

The same apparatus as the first user study, described in Section 5.2.2, were used in this study. It also used the same procedure and design, described in Section 5.2.3. The difference is that this study investigated lower injected *misrecognition* error rates; i.e., 0, 5, 7.5, and 10%.

5.3.3 Results

The whole study lasted from fifty to ninety minutes including the demonstration, practice, and breaks. Upon completion of the study, 25% participants were able to recognize all three error prone letters, 58% recognized the two most error prone letters, and the remaining 17% recognized only the most error prone one.

D'Agostino Kurtosis tests on the dependent variables revealed that the data were normally distributed. Also, a Mauchly's test confirmed that the data's covariance matrix was circular in form. Thus, repeated-measures ANOVA was used for all analysis. The statistical tests used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%. All statistically significant results are presented with effect size (η^2) and power ($1-\beta$). See Appendix A1 and A4 for more information on η^2 and $1-\beta$, respectively.

Similar to the first study and to observe learning, the data was segmented into blocks of ten appearances of each letter in the study. That is, every ten times a given letter was presented to the users to input was treated as a single data point. As all letters appeared exactly ninety times per participant, there were nine segments for each letter.

5.3.3.1 Alternative Method Usage (AMU)

An ANOVA revealed that there was a significant effect of injected *misrecognition* error rate on AMU ($F_{3,11} = 3.52$, $p < .05$; $\eta^2 = .20$, $1-\beta = 0.82$). Average AMU for 0, 5, 7.5 and 10% injected *misrecognition* error rates were 1.09, 6.48, 5.74, and 22.69%, respectively. Figure 45 illustrates this. A Tukey-Kramer test failed to identify groupings. However, a statistically weaker Duncan's test identified two groups, 0-7.5% and 10%.

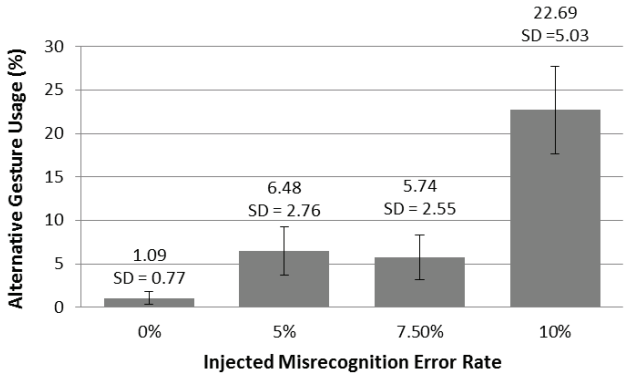

Figure 45. Average Alternative Method Usage (AMU) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation (SD).

For all injected *misrecognition* error rates, the data was again fit with power functions to analyze learning (Card et al., 1983). Figure 42 illustrates this, where the horizontal axis represents the segments and vertical axis represents the average AMU during that segment. As discussed in Section 5.2.4, the 0% condition is averaged across the four non-faulty letters. Again, an attempt was made to fit linear functions to the data (0%: $R^2 = 0.24341$, 5%: $R^2 = 0.2467$, 7.5%: $R^2 = 0.13909$, and 10%: $R^2 = 0.83695$), yet the power functions yielded marginally better results (0%: $R^2 = 0.3672$, 5%: $R^2 = 0.20167$, 7.5%: $R^2 = 0.00681$, and 10%: $R^2 = 0.84642$).

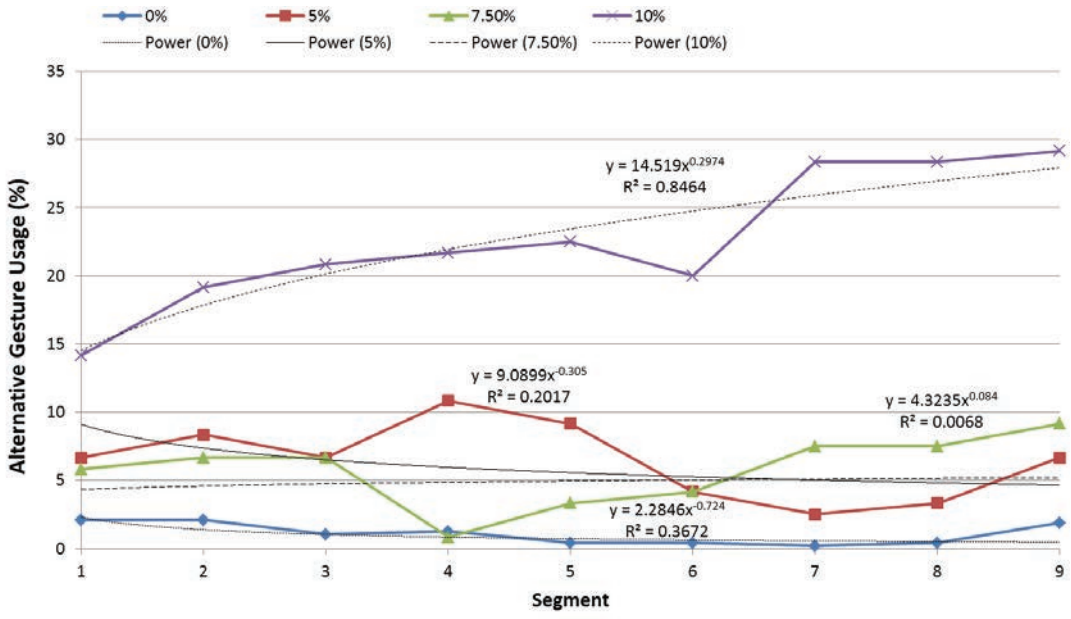

Figure 46. Average Alternative Method Usage (AMU) by injected misrecognition error rates and segments.

5.3.3.2 Input Time (T^h_{input})

An ANOVA on the data did not identify a significant effect of injected *misrecognition* error rate on $\boldsymbol{T}^h_{input}$ ($F_{3,11} = 1.34$, $p > .05$). Average $\boldsymbol{T}^h_{input}$ values for 0, 5, 7.5, and 10% injected *misrecognition* error rates were 1216, 1147, 1181, and 999 milliseconds, correspondingly. Figure 47 illustrates this. Similar to the first user study, the data over all letters correlates very well to the power law of learning (Card et al., 1983), ($y = 1534.9x^{-0.22}$, $R^2 = 0.9574$).

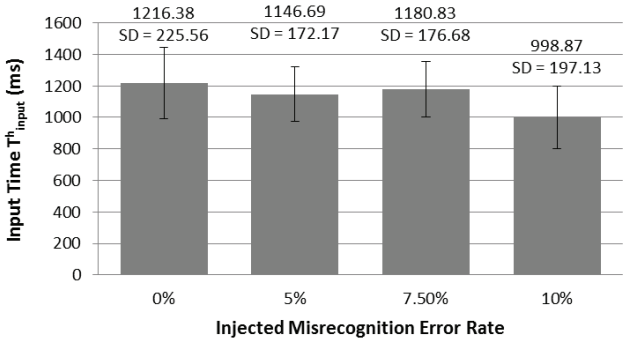

Figure 47. Average Input Time (T^h_{input}) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation (SD).

5.3.3.3 Gestures per Character (GPC)

An ANOVA identified a significant effect of injected *misrecognition* error rate on GPC ($F_{3,11} = 5.33$, $p < .01$; $\eta^2 = .20$, $1–\beta = 0.59$). A Tukey-Kramer test revealed that the 10% injected *misrecognition* error rate yielded a significantly higher GPC than the 0% injected *misrecognition* error rate. Average GPC for 0, 5, 7.5, and 10% injected *misrecognition* error rates were 1.07, 1.16, 1.21, and 1.31, correspondingly, as illustrated in Figure 48. Similar to the previous study, no strong learning effect was visible as the data correlated only weakly with the power law of learning (Card et al., 1983), ($y = 1.2619x^{-0.044}$, $R^2 = 0.6529$).

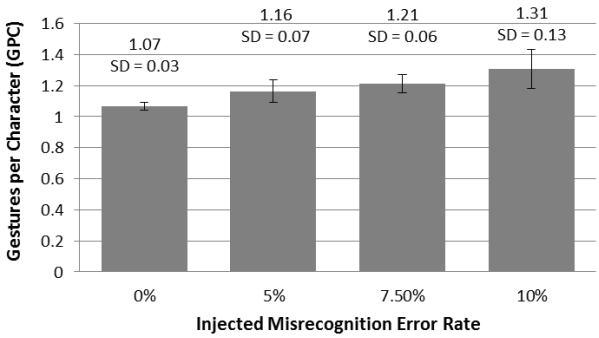

Figure 48. Average Gestures per Character (GPC) over all investigated injected misrecognition error rates. Error bars represent ±1 standard deviation (SD).

5.3.4 Discussion

The results of the study support acceptance of the hypothesis $\boldsymbol{H_{1\,C5}}$ (see Section 5.2). In fact, the results are mostly comparable to the results of the first study: there was a significant effect of injected *misrecognition* error rate on both AMU and GPC, but not on the input time ($\boldsymbol{T}^{h}_{input}$). Besides, substantial learning effects were observed for AMU and input time ($\boldsymbol{T}^{h}_{input}$), but not for GPC. Figure 46 illustrates average AMU by injected *misrecognition* error rates and segments. Similar to the first study, one can see there that participants learned to use the alternative method to input the letters where synthetic *misrecognition* errors were injected more frequently relatively faster than the other letters. Also, the 10% injected *misrecognition* condition, common to both studies, yielded somewhat comparable AMU (32% and 23%) and GPC (1.25 and 1.31) values, which shows that the results of the two experiments are reasonably consistent. Figure 42 and 46 illustrate that users adapted to the 10% *misrecognition* condition nearly the same way. Therefore, results of this study further validate the initial hypothesis, and extend the findings of the first study towards lower (injected) *misrecognition* error rates.

Figure 46 shows that adaptation to 0, 5, and 7.5% injected *misrecognition* error rates were relatively slower than 10%. This is most likely due to insufficient exposure. In the post-study questionnaire, most participants (75%) responded that they managed to identify the 5 and 7.5% faulty letters only shortly before the study ended. This is also apparent in Figure 46, where one can see a distinct trend in adaptation through an increased alternative method usage for these letters during the last three segments. An ANOVA on the data from these segments identified a significant effect of injected *misrecognition* error rate on AMU ($F_{3,11} = 3.96$, $p < .05$; $\eta^2 = .20$, $1–\beta = 0.82$). A Tukey-Kramer test identified two statistically different groups for the last three segments: 0% and 5-10%, while a statistically weaker Duncan's test identified three statistically different groups for the last three segments: 0%, 5-7.5% and 10%. Note that the average

input time ($\boldsymbol{T}^{h}_{input}$) was higher during the second study, compared to the first. This is presumably due to the inclusion of relatively more inexperienced users during the second study.

5.4 Overall Discussion and Implications

Overall, the studies showed that users learn to use alternative gestures more quickly, if the primary gestures are faultier. This validates hypothesis $\boldsymbol{H_{1\ C5}}$. The results of this work complements findings in psychology (Craik and Lockhart, 1972; 1975), skill acquisition (Schmidt and Bjork, 1992), and user interface research (Cockburn et al., 2007; Ehret, 2002; Riche et al., 2010) that imply that graphical user interfaces requiring greater efforts from users may facilitate the transition to recall-based expert behavior (also discussed in Section 2.6.6). After all, faulty gestures increase user effort to some degree. The results also indicate that gesture recognizers need to achieve substantially more than 90% accuracy in practice to appear less (or maybe even in-)distinguishable from a "perfect" system. This is similar to results reported in Chapter 4 for keyboard based text entry.

The fact that users adapt to unreliable gesture recognizers by using an alternative method for inputting the letters that are frequently misrecognized by the system should encourage developers to provide users with alternative gesture set(s) along with the primary one. They should also permit users to swap a primary gesture with an alternative one, and vice versa, as necessary. A more advanced technique can keep track of the primary and the alternative method usage for all letters and might then even automatically switch the primary gestures with the alternative ones for letters that are frequently inputted with the alternative method. This may increase the overall recognition accuracy, providing that the recognition rate is higher for the alternative method than the primary one. This can be achieved by using more distinct gestures as alternatives, since results showed that users adapt to the alternative gestures for frequently misrecognized letters, even when the alternative gestures are relatively less intuitive (and harder to discover) than the primary gestures. One may speculate such a feature can be applied not only for text entry but also for other gesture-based techniques, such as natural user interfaces and application launchers.

Indirectly, these results also indicate that gesture recognizers need to achieve substantially more than 90% accuracy to be not easily distinguishable from a "perfect" system. This is similar to other results for keyboard based text entry presented in Chapter 4. Besides, looking across both studies, one interesting observation is that about half of the users were unable to identify all faulty gestures in the system within about an hour. One could speculate that this is likely due to different cognitive strategies or personality types. Investigating this is a topic for future work.

5.5 Limitations

The ANOVAs on the data from User Study 1 identified a significant effect of injected *misrecognition* error rate on Gestures per Character (GPC) and Alternative Method Usage (AMU). A post-hoc analysis detected a *large* effect size for both of these dependent variables (see Appendix A1). Further post-hoc analysis revealed that the statistical power exceeded the 0.80 threshold (see Appendix A4) at the observed *large* effect size level for both GPC and AMU. This indicates that there was adequate statistical power for these dependent variables.

Table 6. Detected effect size and measured statistical power.

User Study	Dependent Variable	Effect Size (η^2)	Power ($1-\beta$)
Study 1	GPC	Large	> 0.80
	AMU	Large	> 0.80
Study 2	GPC	Large	< 0.80 (= .59)
	AMU	Large	> 0.80

Similarly, an ANOVA on the User Study 2 data identified a significant effect of injected *misrecognition* error rate on Gestures per Character (GPC) and Alternative Method Usage (AMU). A post-hoc analysis detected a *large* effect size for both GPC and AMU (see Appendix A1). Further post-hoc analysis revealed that the statistical power exceeded the 0.80 threshold (see Appendix A4) for AMU, but not for GPC (see Table 6). This implies that a larger sample size (N) may necessary to show sufficient statistical power. Appendix A4 elaborates on the criteria used for calculating statistical power.

As the study recruited participants by using convenience sampling from the university community, the results may not generalize to a larger population. Nevertheless, an attempt was made to counteract this potential confound by recruiting not only university students but also instructors and staff.

5.6 Summary

This chapter presented the results of two user studies that verified that users gradually adapt to *misrecognition* errors and that this adaptation rate depends on how frequently such errors occur. That is, users adapt to an error faster if it occurs more frequently. It also speculated on the practical implications of this work.

The next chapter investigates whether the use of pseudo-pressure in predictive text entry can improve the overall text entry performance by increasing entry speed and reducing errors.

Chapter 6

A New Text Entry Technique

Although much work has targeted pressure-based user interfaces and widgets for tabletops and large displays, few attempts focus on mobile devices. The main reason for this is technological. No current mobile device provides hardware support for measuring pressure. However, recent work (Graham-Rowe, 2010; Nurmi, 2009) indicates that future mobile phones may include pressure-sensitive touchscreens as an alternative interaction modality. A recent opaque touchpad, called Synaptics ForcePad[14], already provided support for detecting pressure levels.

Several software solutions are available to detect pressure on touchscreens. Yet none of these are broadly applicable, as they either increase the time to perform tasks that involve additional pressure, or are user specific; e.g., due to different finger sizes and types of touch, also discussed in Section 2.2.9. This chapter presents a new hybrid pseudo-pressure detection technique that combines the existing touch-point- and time-based approaches to detect pressure. The new technique is evaluated in a user study for two different pressure levels. An investigation if users interpret fairly general terms such are *regular* and *extra* pressure in a reasonably consistent manner and how much force is really applied for each level was carried out in a separate user study.

As discussed in Section 2.2.4.3, almost all recent virtual keyboards augment text entry with prefix-based word prediction and auto-correction. These methods suggest the most probable word(s) based on what users are typing and automatically correct *likely* misspelled words. Almost all these methods require users to tap on an area outside the virtual keyboard to reject or bypass a suggestion. This requires additional mental preparation, visual scan time, as well as a finger movement to the target. Due to the small target sizes used, users may need several attempts to reject a prediction. This increases the possibility of accidentally selecting the wrong word as well. This chapter also presents a new pressure-based technique for prediction rejection that does not require tapping outside the keyboard. Instead, it requires users to apply more pressure for the tap on the next key. The performance of this technique was compared with the

[14] http://www.synaptics.com/solutions/products/forcepad

conventional technique in an empirical study. User experience data on the new hybrid pressure detection simulation and the pressure-based predictive text entry technique have also been provided.

6.1 Hybrid Pressure Detection Simulation

None of the existing software solutions to detect pressure on touchscreens are broadly applicable. These techniques either increase the time to perform tasks that involve additional pressure or are user specific due to different finger sizes and types of touch. Please refer to Section 2.2.9.2 for a review on the existing techniques and their limitations. To counteract these issues, a new hybrid method is developed here, which combines a time- and a touch-point-based approach. The new method uses the average time it takes to perform a task and the average touch-point movement for that specific task as baselines. Then, it simulates extra pressure when users take more time and/or their touch-point moves a greater distance than the baselines while performing that task. Theoretically, the new hybrid technique will simulate pressure detection faster and more reliably. In particular and if the touch-point threshold is crossed before the time threshold, users will not have to wait to trigger extra pressure detection. In contrast, the naïve time-based approach always requires additional time to perform a task. Therefore, the new hybrid technique will not only save time but also increase the probability of detecting extra pressure (assuming that one approach will detect pressure when the other fails).

As discussed in Section 2.2.9.2, the touch-point moves further when additional pressure is applied. This movement is somewhat proportional to the force applied on the screen. The touch-point-based approach simulates pressure detection based on this movement. This approach is somewhat similar to the contact-area-based one. The main difference is that the touch-point-based approach does not use contact area but considers only the touch center coordinates (*x*- and *y*-axis). This makes it simpler, more straightforward, and theoretically even applicable to styli-based interactions (Ramos et al., 2004). As most current mobile touchscreens do not provide contact area information, many contact-area-based implementations derive contact areas from the touch coordinates with various heuristics (Boring et al., 2012). These heuristics are not necessarily 100% reliable. As the touch-point-based approach works directly on the touch point movement, which is less prone to misinterpretation, it is reasonable to consider it more reliable.

6.2 User Study 1

A user study was conducted to validate the assumption that the hybrid technique can detect pressure more efficiently. Two pressure levels were examined: *regular* and *extra*. Participants were instructed that regular

pressure represented the level of pressure typically applied on touchscreens, while extra pressure represented relatively stronger pressure. Only two levels were investigated, as the main target of this is work is to use pseudo-pressure in text entry, and prior research showed that more than two pressure levels do not work well in text entry tasks (McCallum et al., 2009; Wang et al., 2009), see also Section 2.2.9.1. Thus, the study tested the following hypothesis:

($H_{1\ C6.1}$) *On average, tap times and touch point movements are significantly different for the two different pressure levels: regular and extra.*

6.2.1 Participants

Twelve participants, aged from 21 to 29 years, average 24, participated in the user study. Appendix A3 elaborates on the procedure used to decide the number of participants (sample size). They were recruited through online social communities, local university e-mailing lists, by posting flyers on campus, and by word of mouth (convenience sampling). However, only experienced touchscreen users were recruited to ensure familiarity with touchscreens. Five of the participants were female and all were right-handed. They were all familiar with the virtual Qwerty layout. They all received a small compensation (CAD 5.00) for participating.

6.2.2 Apparatus

A custom application, developed with the iPhone SDK, was used with an Apple iPhone 4, 115.2×58.6×9.3 mm, 137 grams, at 640×960 resolution for the user study. The application's virtual Qwerty keyboard was visually identical to the iPhone's default keyboard. See Figure 49. However, the Shift and "?.123" keys were disabled, as these were not required during the study. The custom keyboard featured the key enlargement feedback of the iPhone's default keyboard. No auditory feedback was provided. The application calculated all metrics directly and logged all action events with timestamps.

6.2.3 Procedure

During the study, participants inputted all the letters of the English language, plus the Space character, using both regular and extra pressure. The application presented one character at a time in random order and in random cases to avoid ordering effects. Participants were asked to input the lowercase characters by applying regular pressure and the uppercase characters by applying extra pressure. Regular pressure was described to them as the level of pressure they usually apply on their touchscreen-based devices, while

extra pressure was described as relatively stronger pressure than that. The uppercase and lowercase space characters were represented by “sp” and “SP”, respectively.

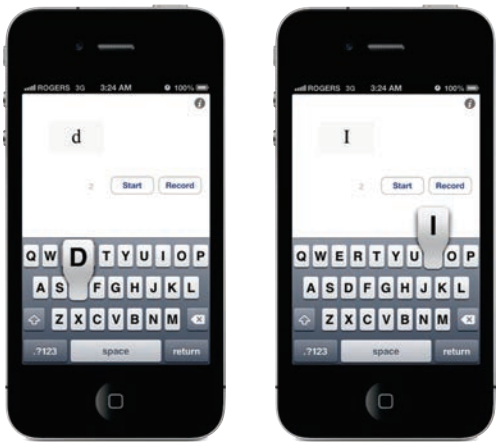

Figure 49. The custom application used during User Study 1. In the first screenshot, users had to tap on the “D” key with regular pressure. In the second screenshot, they had to tap on the “I” key with extra pressure.

Participants were instructed to first examine the presented character, understand the level of pressure they need to apply, and then to perform the specified task. They were not provided with practice trials. The application did not permit participants to correct their mistakes, as the focus was on the differences between regular and extra pressure in terms of tap time and touch point movement, and not on input accuracy. Upon completion of inputting a character, the next one was automatically presented on the screen. Participants were instructed to hold the device with their dominant hand in portrait orientation, and then to input using the thumb of that hand. The position is the most frequently used one by mobile users (Hoober, 2013). Participants were informed that they could take short breaks (maximum 5 minutes) between blocks.

The system calculated the following metrics.

1. **Tap Time (millisecond):** This signifies the time it took to input a character. This was calculated by measuring the time difference from the moment users touched the virtual keyboard until they lifted their fingers.
2. **Touch Point Movement (millimeter):** This signifies the distance users touch point traveled while inputting a character. First, the following equation was used to calculate the distance in pixels.

$$Touch\ point\ movement = \sqrt{dx^2 + dy^2}\ pixels \qquad \text{Equation (19)}$$

Where, *dx* and *dy* are the differences between the *x*- and the *y*-coordinates of the start- and the end-points, respectively. Then, the data was converted to millimeters.

6.2.4 Design

A within-subjects design was used, where the within-subjects factor focused on the regular (lowercase characters) and the extra pressure (uppercase characters). The dependent variables (and the metrics) were tap time (ms) and touch point movement (mm). There were four blocks. In each block, participants inputted 27 lowercase and 27 uppercase characters. These characters were presented one at a time in random order. In summary, the design was:

12 participants ×
4 blocks ×
54 characters (27 regular pressure lowercase and 27 extra pressure uppercase characters, randomized)
= 2,592 characters, in total. Each participant inputted 216 characters.

6.2.5 Results

Both Anderson-Darling and D'Agostino Kurtosis tests on the dependent variables revealed that the data were not normally distributed. Therefore, a Wilcoxon Signed-Rank test was used for all analysis. The statistical tests used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%.

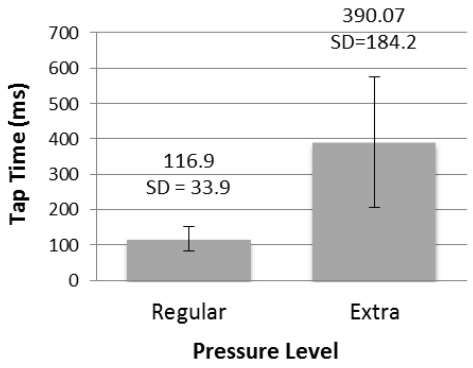

Figure 50. Average tap time (millisecond) for different pressure levels. Error bars represent ±1 standard deviation (SD).

6.2.5.1 Tap Time (Millisecond)

A Wilcoxon Signed-Rank test indicated that there is a significant difference between regular and extra pressure in terms of tap time (z = -30.49, p < .0005). The average tap times for regular and extra pressure were 116.9 ms and 390.07 ms, respectively.

6.2.5.2 Touch Point Movement (Millimeter)

A Wilcoxon Signed-Rank test revealed a significant difference between regular and extra pressure in terms of touch point movement (z = -17.76, p < .0005). The average touch point movements for regular and extra pressure were 0.289 mm and 0.503 mm, correspondingly. Figure 51 illustrates this.

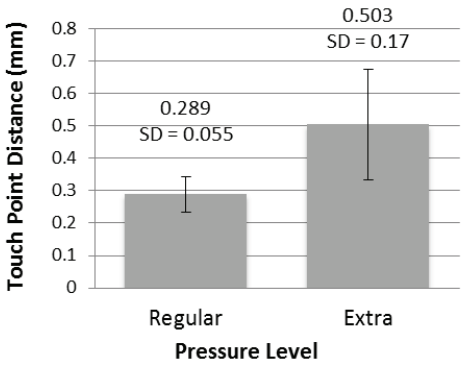

Figure 51. Average touch point movement (millimeter) for different pressure levels. Error bars represent ±1 standard deviation (SD).

6.2.6 Discussion

The results of the study support acceptance of the hypothesis $\boldsymbol{H_{1\ C6.1}}$ (see Section 6.2). The results establish that there is a significant difference between regular and extra pressure both in terms of tap time and touch point movement. On average, taps took more time and the touch point moved more when extra pressure was applied. This is a strong indication that a hybrid criterion can be useful to simulate pressure detection, at least for two pressure levels. Further study identified three distinct user groups. About 67% of users took significantly more time to tap. But their touch point did not move significantly. The touch point of 8% of all users moved significantly more. Yet these did not take significantly more time. The remaining 25% took both significantly more time and their touch point moved more with extra pressure. This indicates that an approach based on either time, contact-area, or touch-point *alone* cannot accommodate all users, as user behavior varies too much. In contrast, the hybrid approach supports all three groups.

The data was further analyzed to investigate the effect of key positions on tap time and touch point movements by segmenting the virtual keyboard into 3×1 and 2×2 grids, similar to Parhi et al. (2006). As no statistical significance was identified, this was not pursued further.

6.3 User Study 2

The results of the first study verified that the two different pressure levels are easily distinguishable through a combination of tap time and touch point movement. Also, different users seem to interpret regular and extra pressure in a reasonably consistent way. While some have investigated the amount of force applied on flat surfaces (Srinivasan and Chen, 1993; Mizobuchi et al., 2005; Ramos et al., 2004), their results do not apply directly in this work because they either explored more than two pressure levels or used a pressure-sensitive stylus. Thus, an additional study was conducted to detect the force users apply when limited to two pressure levels. The study tested the following hypothesis:

($H_{1\ C6.2}$) *The amounts of force applied on a flat surface are substantially different for the two different pressure levels: regular and extra.*

6.3.1 Participants

Fourteen participants, aged from 23 to 46 years, average 31.4 years, participated in the study. Appendix A3 elaborates on the procedure used to decide the number of participants (sample size). They were recruited through online social communities, local university e-mailing lists, and by word of mouth (convenience sampling). Four of them were female and all of them were right-handed. They all owned and frequently used a touchscreen-based mobile device.

6.3.2 Apparatus

A DYMO M5 Digital Postal Scale was used for this study. The scale had 5 lb weight capacity. It displayed the weight of an object in 0.1 oz increments with ±0.1 oz accuracy.

6.3.3 Procedure

The study used a finger posture akin to holding a touchscreen device with one hand and then tapping on it with the thumb of the same hand. For this, the digital scale was placed on the table and participants were

asked to sit in front of it. They were then instructed to place the closed fist of their dominant hand on the table, and to tap on the scale with only the thumb of that hand, as if tapping on a virtual keyboard. See Figure 52. This design eliminates the option of using arm strength to apply pressure and limits users to using only their thumb. There were two conditions: regular and extra pressure. In the regular pressure condition, participants were asked to tap on the scale six times with regular pressure. For the extra pressure condition, they were asked to do the same with extra pressure. Conditions were counterbalanced to avoid asymmetric skill transfer. Similar to the first study and during the regular pressure condition, participants were instructed to tap on the scale with the amount of pressure they usually apply on a virtual keyboard. In the extra pressure condition, they were asked to apply relatively more pressure than that. The experimenter recorded the readings in ounces (oz) with pen and paper, which were later converted to newton (N). Participants could not see the scale readings, since this might influence their performance.

Figure 52. A participant tapping on the digital scale.

6.3.4 Design

A within-subjects design was used for the two factors: regular and extra pressure. The dependent variable (and the metric) was the force applied (N) on the surface. In summary, the design was:

14 participants ×
2 conditions (regular and extra pressure, counterbalanced) ×
6 taps on the scale
= 168 taps, in total. Each participant tapped 12 times.

6.3.5 Results

Both Anderson-Darling and D'Agostino Kurtosis tests on the dependent variables revealed that the data were not normally distributed. Therefore, Wilcoxon Signed-Rank test was used for all analysis. The statistical tests

used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%.

6.3.5.1 Applied Force (Newton)

A Wilcoxon Signed-Rank test indicated that there is a significant difference between regular and extra pressure in terms of applied force (z = -2.86, p < .004). The average force applied for regular and extra pressure was 1.04 N and 3.24 N, respectively.

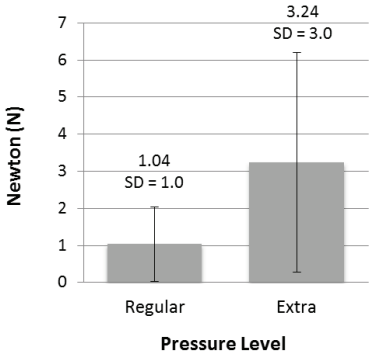

Figure 53. Average force (N) applied for regular different pressure levels. Error bars represent ±1 standard deviation (SD).

6.3.6 Discussion

The results of the study support acceptance of the hypothesis $\boldsymbol{H_{1\ C6.2}}$ (see Section 6.3). The results show that the forces applied during the regular and extra pressure conditions were significantly different. Users applied on average 1.04 N for regular and 3.24 N for extra pressure. This matches Mizobuchi et al.'s (2005) work, where they identified force levels between 0 and 3 N to be comfortable and 4 N to be (too) strong. Similarly, this study found a force level well below 3 N for regular and about 3 N for extra pressure. Nevertheless, this result is only an approximation because the data were collected on a postal scale instead of a pressure sensitive touchscreen.

6.4 Pressure-Based Predictive Text Entry

The results of the first two studies established that users comprehend regular and extra pressure in a reasonably consistent manner. Also, the first study confirmed that a hybrid of time- and touch-point-based

approaches could detect pressure reliably on touchscreens. Here these findings are applied to text entry by developing and evaluating a new pressure-based predictive text entry technique.

6.4.1 The New Technique

Cancelling a prediction requires the user to tap on an area outside of the virtual keyboard, a relatively distant target. The time to do this depends not only on mental preparation and visual scan times, but also on the distance and width of the target (the Fitts' law parameters). Furthermore, the small target size increases the potential for errors. For example, while attempting to tap on a prediction bubble to reject a prediction, one may miss the target. Tapping then on the Space key without visual verification will result in input of an entirely wrong word. These and additional issues with predictive text entry have also been discussed in Section 2.2.4.3. To address these discussed shortcomings, this section presents a new pressure-based predictive technique that does not require tapping outside the keyboard.

The new technique resembles and behaves like the default iPhone keyboard. However, one can apply extra pressure on the next target key (which may be any key) to bypass prediction. Figure 54 (c) illustrates word prediction in the new technique, where the system predicted the most probable word based on the inputted prefix. Now, one can perform any of the above-mentioned tasks: accept, reject, or ignore the prediction. To *reject* the prediction, one only has to tap on the next key with extra pressure. For example, to input "educo", one taps on the "O" key with extra pressure. As the new technique reduces the average finger movement distance, it can be hypothesized that this will not only improve text entry speed but also reduce errors. The default iPhone keyboard was used as a baseline as most users use this or a similar keyboard on their devices (Arif, 2012). Also, the intent of this work is not to evaluate the quality of the predictive system, but to evaluate pressure as a modality in predictive text entry, which is mostly independent.

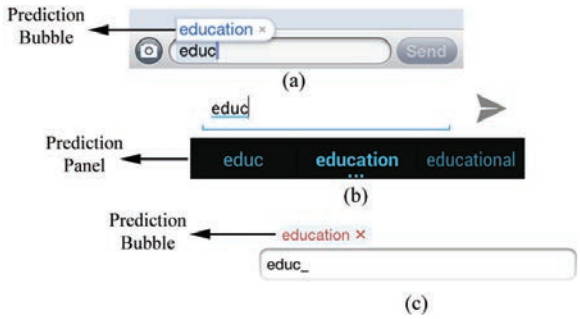

Figure 54. Default word prediction systems on the (a) Apple iPhone, (b) Android OS, and (c) the new technique.

6.4.1.1 Word Prediction

A straightforward word prediction system was created for the study. For this, a list of the most frequent 5000 English words was used (Davies, 2011), extracted from the 450 million-word Corpus of Contemporary American English (COCA). Each time users input a character the system attempts to find matches in the list and suggests the most frequent word in a prediction bubble. See Figure 54 (c). Based on several pilots, the following conditions were applied in the prediction system.

1. At least two characters have to be inputted for the system to suggest a word. For example, users have to input at least "ed" to get the prediction "education".
2. If no match was found, the system will assume that the user made a spelling mistake and will suggest the most frequent word with a Levenshtein string distance (Levenshtein, 1966) less than three to the inputted prefix. For example, with "edution" as input, the system will suggest "education", with an edit distance of two.
3. After the user rejects a prediction and similar to many other predictive systems, the system resumes suggestions on a Space, Return, or Backspace.

The prediction system was informally tested *without* pressure detection with three experienced Apple iPhone users. They all inputted random texts for ten minutes. None of them noticed any notable difference between the tested and the default iPhone prediction system, in terms of prediction accuracy or processing time.

6.4.1.2 Pressure Detection

Pressure detection was simulated based on the proposed hybrid approach. A threshold of 200 ms was used for the tap time, and a threshold of 0.389 mm for touch point movement. Extra pressure was detected when users took more time *and/or* their fingers slid more than the above-mentioned thresholds. These values were picked based on the results of the first user study, by selecting the "deepest" spot between the two alternatives as thresholds.

6.5 User Study 3

This user study compared the new pressure-based predictive text entry technique with the conventional technique (the default iPhone method). It also explored user preference for pressure as an alternative modality and (indirectly) evaluated the hybrid pressure detection approach. It tested the following hypothesis:

($H_{1\ C6.3}$) *The new technique improves mobile touchscreen predictive text entry performance in terms of speed, accuracy, and user comfort, compared to the conventional technique.*

6.5.1 Apparatus

The same physical apparatus as in the first user study was used. The custom application was modified to support predictive text entry, as discussed previously. Again, the Shift and the "?.123" keys were disabled, as users were not required to use these during the user study.

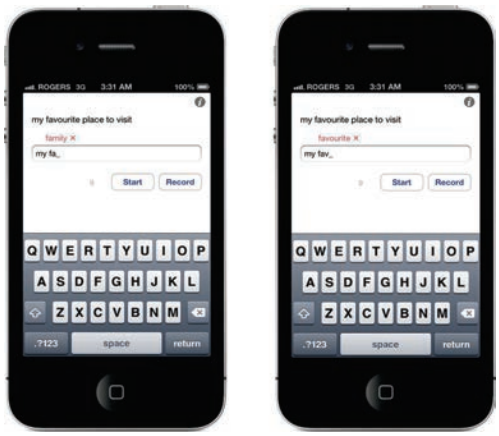

Figure 55. The custom application during User Study 3. Note the change in the prediction in the two screenshots.

6.5.2 Participants

Twelve new participants, aged from 22 to 32 years, average 28 years, participated in the study. They were recruited through online social communities, local university e-mailing lists, and by word of mouth, by using convenience sampling. The user study targeted only experienced touchscreen users and fluent English speakers to minimize learning effects. Towards this, only native speakers or people who had spent at least five years in an English speaking environment were recruited. All were also frequent mobile phone users and had prior experience with touchscreens (on average 2 years). Two of them were female and one was left-handed. They all used a virtual Qwerty keyboard on their mobile device to input text. Amongst them, six used both word prediction and auto-correction, one used only word prediction, two used only auto-correction, and the rest used none of the features. They all received a small compensation (CAD 10.00) for participating.

6.5.3 Procedure

The study compared two virtual keyboards, both of which use prefix-based word prediction: the *pressure*-based one with the new pressure-based prediction rejection technique, and the *conventional* one. During the study participants inputted short English phrases with both techniques. Phrases were taken from a widely used corpus (MacKenzie and Soukoreff, 2003) that correlates very well with the English language character frequency. See Appendix A2 for more information on the phrase set. Sixty random phrases without uppercase, numeric, or special characters were selected for each technique, which users inputted in the same order during the two conditions. For each condition, the same phrases were used to ensure relatively similar prediction rate and accuracy for all users. To reject a prediction with the pressure-based technique participants had to apply extra pressure on the next target key, while with the conventional technique they had to tap on the prediction bubble. The conditions were counterbalanced to avoid asymmetric skill transfer.

Figure 56. The experiment setup for the final user study. Here, a user is inputting short English phrases in a seated position with the custom software.

Users were instructed to hold the device in the portrait orientation with their dominant hand and then to type using the thumb of that hand. See Figure 56. The system displayed one phrase at a time and users had to transcribe that phrase. They were asked to take the time to read and understand the phrases in advance, then to enter them as fast and accurate as possible, and to press the Return key when they were finished to see the next one. No practice was given, but both methods were briefly demonstrated before the study. During this and for the pressure-based technique, special emphasis was placed on how extra pressure could be applied on any key to bypass predictions, including Space and Backspace. This was deemed necessary as users had showed uncertainty on this issue during a pilot. Participants were informed that they could take a short break (maximum 5 minutes) between conditions. Timing started from the entry of the first character and ended with the last. All key actions were performed on touch-up, similar to the default Apple iPhone keyboard. Hence, when users touched a wrong key, they could drag their finger to the right key before lifting it. They were asked to work normally, that is, to correct their errors as they noticed them. However,

they had to exclusively use the Backspace key for editing, as direct cursor control was disabled to remove a potential confounding factor.

Both keyboards used the same method for word prediction, as discussed earlier. It was verified that the frequency list contained all words used in the selected 120 phrases. Then, 10% of the words were deliberately deleted from the list for each condition. This replicates the scenario where an incorrect prediction is provided and the user is forced to bypass it. This is not uncommon in predictive text entry, as users have to input non-dictionary words, such as abbreviations, names, alphanumeric text, and slang in real life. Some users also input text with the wrong prediction dictionary activated on occasion. MacKenzie et al., (2001) also highlighted the necessity for adequate handling of non-dictionary words in evaluations of predictive text entry. The 10% deleted words were selected randomly, subject to the restriction that they consist of at least three characters and do not appear more than once in the phrases. This guaranteed that an incorrect prediction would not be offered more than once, to prevent user adaptation.

The system calculated the average text entry speed (WPM), error rate (TER), corrective operation (%), and the sum of the mental preparation and physical movement time (milliseconds) for each task. Corrective operation signifies the average percentage of Backspace use with each technique, while the sum of the mental preparation and physical movement time was measured from the end of the previous task (touch-up) to the beginning of the next task (touch-down). The system also recorded user actions on a prediction, including the rate at which predictions were accepted, rejected, and ignored. In addition, the system kept a record of how extra pressure was triggered, that is, whether through extra time, additional touch-point movement, or through both. Finally, upon completion of the study users completed a questionnaire.

6.5.4 Design

A within-subjects design was used for the two factors: conventional and pressure-based techniques. The dependent variables (and the metrics) were text entry speed (WPM), error rate (TER), corrective operation (Backspace use), mental preparation and movement time (milliseconds), and the rate of accept, reject, and ignore user actions on predictions. In summary, the design was:

12 participants ×
2 conditions (conventional and pressure-based technique, counterbalanced) ×
60 phrased per condition
= 1,440 phrases in total. Each participant entered 120 phrases.

6.5.5 Results

After filtering outliers beyond three standard deviations from the mean (1.11% of the data) D'Agostino Kurtosis tests on the dependent variables confirmed that the data were normally distributed. In addition, a Mauchly's test confirmed that the data's covariance matrix was circular in form. Thus, repeated-measures ANOVA was used for all analysis. The statistical tests used a significance level (α) threshold of 5%. That is, the null hypothesis was rejected when a probability value was below 5%. All statistically significant results are presented with effect size (η^2) and power ($1-\beta$). See Appendix A1 and A4 for more information on η^2 and $1-\beta$, respectively. A Wilcoxon Signed-Rank test was used to analyze the nonparametric questionnaire data.

6.5.5.1 Entry Speed (WPM)

An ANOVA on the data identified a significant effect of technique on entry speed ($F_{1,11} = 13.30$, $p < .005$; $\eta^2 = .02$, $1-\beta = 0.04$). The average entry speeds for the conventional and the pressure-based techniques were 16.7 and 18.23 WPM, correspondingly. Figure 57 illustrates this.

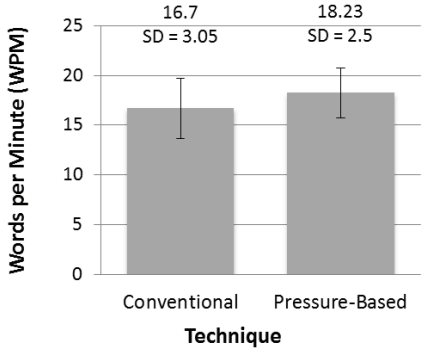

Figure 57. Average entry speed (WPM) for both techniques. Error bars represent ±1 standard deviation (SD).

6.5.5.2 Error Rate (Total Error Rate)

An ANOVA on the data identified a significant effect of technique on error rate ($F_{1,11} = 11.99$, $p < .01$; $\eta^2 = .02$, $1-\beta = 0.06$). The average TER for the conventional and the new techniques was 9.31 and 7.02%, respectively. See Figure 58.

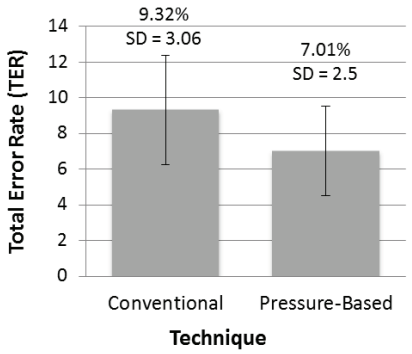

Figure 58. Average error rate (TER) for both techniques.
Error bars represent ±1 standard deviation (SD).

6.5.5.3 Corrective Operation (Backspace Use)

An ANOVA identified a significant effect of technique on corrective operations ($F_{1,11} = 6.81$, $p < .05$; $\eta^2 = .09$, $1-\beta = 0.69$). Average corrective operations for the conventional and new techniques were 8.31 and 6.49%, respectively. Figure 59 shows this. This metric considered only Backspace, as direct cursor control was disabled during the study.

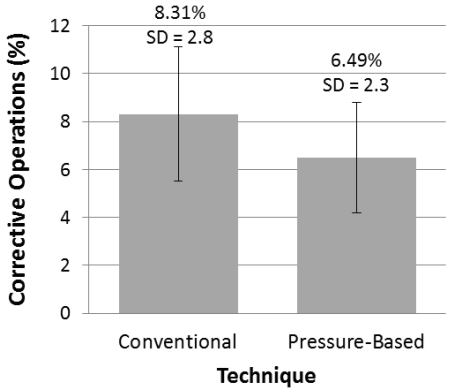

Figure 59. Average corrective operations (%) for both techniques.
Error bars represent ±1 standard deviation (SD).

6.5.5.4 Mental Preparation and Movement Time (Milliseconds)

An ANOVA did not identify a significant effect of technique on the sum of mental preparation and physical movement time ($F_{1,11} = 3.65$, $p = .08$; $\eta^2 = .10$, $1-\beta = 0.81$). The averages for the conventional and new techniques were 848.72 ms and 721.99 ms, respectively. Figure 60 illustrates this.

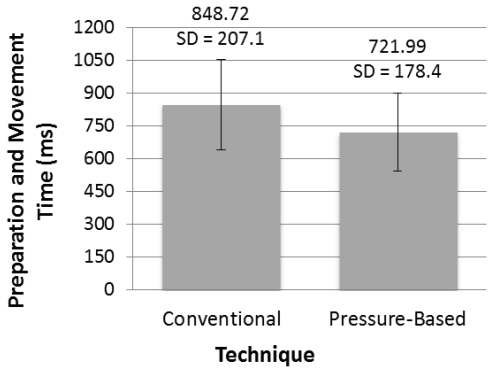

Figure 60. Average sum of mental preparation and physical movement time for both techniques. Error bars represent ±1 standard deviation (SD).

6.5.5.5 User Actions on Predictions (Accepted, Rejected, Ignored)

There was no significant effect of technique on accepted prediction rate ($F_{1,11} = 0.32$, ns). However, there was a significant effect on rejected prediction rate ($F_{1,11} = 6.48$, $p < .05$; $\eta^2 = .09$, $1–\beta = 0.69$), and also on ignored prediction rate ($F_{1,11} = 5.93$, $p < .05$; $\eta^2 = .05$, $1–\beta = 0.43$). Figure 61 illustrates the average user actions on predictions for both techniques.

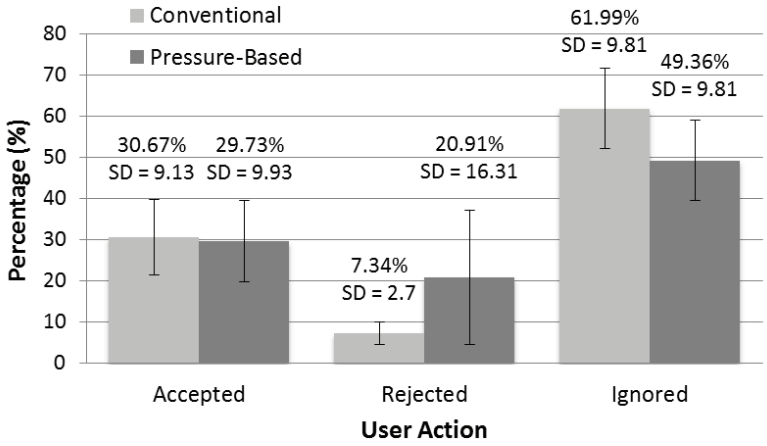

Figure 61. The average user actions on predictions (accepted, rejected, or ignored) for both techniques. Error bars represent ±1 standard deviation (SD).

6.5.5.6 Hybrid Pressure Detection

The data from the pressure-based condition was further analyzed to identify the rate at which the individual pressure detection simulation criteria were used by the hybrid method. Results showed that 58.59% of the time the hybrid technique detected extra pressure with the time-based approach, 30.8% with the touch-

point-based approach, and the remaining 10.61% with both criteria simultaneously. Figure 62 illustrates this. A Friedman test found these three to be significantly different from one another ($\chi^2 = 17.92$, $p < .0005$, $df = 2$).

6.5.6 User Evaluation

Upon completion of the study participants responded to several questions on a seven-point Likert scale. A Wilcoxon Signed-Rank test was used to analyze the questionnaire data. The seven-point scales were later converted to three-point scales using linear transformation to calculate ratios (%). All ratings below four on the seven-point scale were mapped to one, all fours to twos, and all ratings above four to three. Some responses were converted to binomial data. Everything above four was rated as "accept" and below four as "reject" or vice versa, depending on the phrasing of the question. Ratings of four were disregarded. Such a mapping is common practice in statistics (Dawes, 2008).

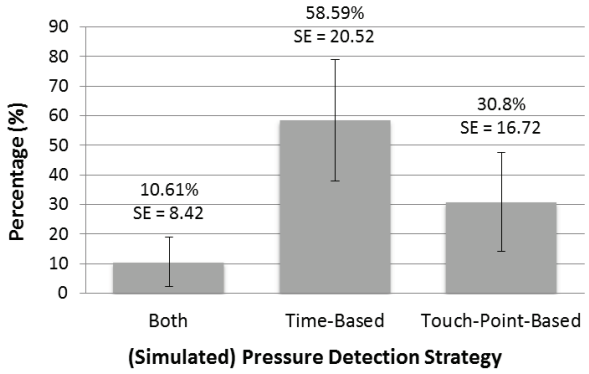

Figure 62. The average use of the extra pressure detection simulation criteria by the hybrid method. Error bars represent ±1 standard deviation (SD).

6.5.6.1 Ease of Use

A Wilcoxon Signed-Rank test revealed that the two techniques differ significantly in their perceived ease of use ($z = -2.72$, $p < .05$). Average user ratings for the conventional and the new techniques were 3.08 and 5.75, respectively. See Figure 63. On average, 83% found the new technique easier to use than the conventional one. Also, most users (83%) responded that they felt no fatigue or discomfort while using the new technique.

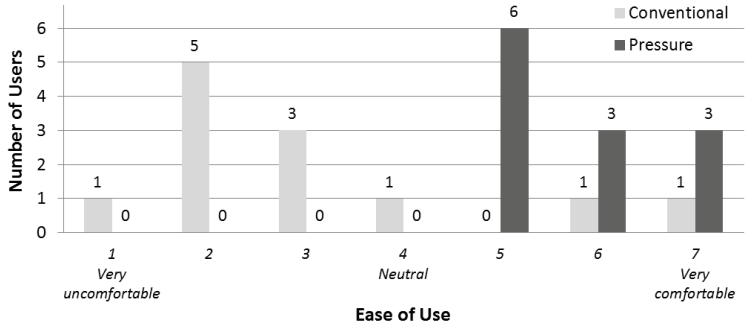

Figure 63. User feedback on how easy users found inputting text with the techniques, on a seven-point Likert scale.

6.5.7 Speed and Accuracy

A Wilcoxon Signed-Rank test identified significance with respect to user perceived entry speed (z = -2.85, p < .005) and accuracy (z = -2.05, p < .05). Average user ratings for the conventional and the new techniques were 3.75 and 5.25 for entry speed, and 4.17 and 5.25 for accuracy. Figure 64 illustrates this. 83% users found inputting text with the new technique faster and 58% found it more accurate compared to the conventional technique.

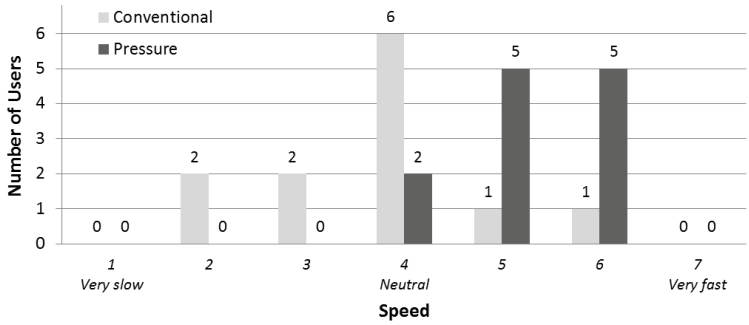

Figure 64. User feedback on how fast they thought their text entry was with the two techniques on seven-point Likert scales.

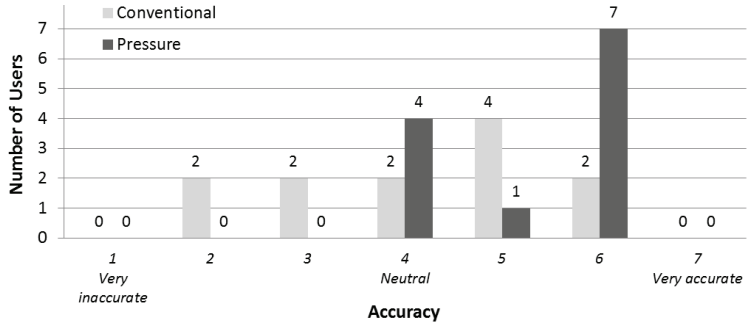

Figure 65. User feedback on how accurate they thought their text entry was with the two techniques on seven-point Likert scales.

6.5.8 Pressure Detection Simulation

Participants were also asked to rate the accuracy of the pressure detection simulation technique. Results showed that 75% found the new pressure detection approach (Section 6.1) accurate. See Figure 65. A Chi-squared test on the three-point scale derived from the original seven-point Likert scale found this to be significant ($X^2_{(2)}$=9.5, p<.01).

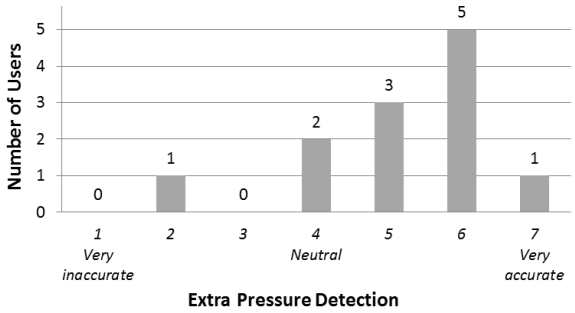

Figure 66. User feedback on how accurate they thought the pressure detection simulation was during the pressure-based condition on a seven-point Likert scale.

6.5.9 Overall Rating

A Wilcoxon Signed-Rank test identified significance with respect to participants' overall rating of the two techniques (z = -2.27, p < .05). Average ratings for the conventional and the new techniques were 3.75 and 5.50, respectively. See Figure 67. Results showed that most users (83%) favored the new technique over the conventional one.

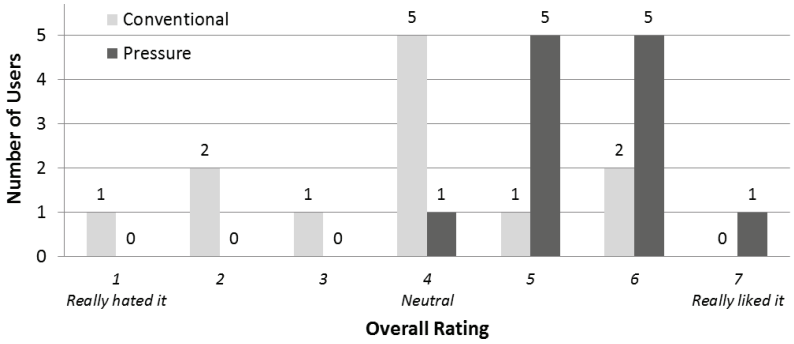

Figure 67. User feedback on how much users liked the examined techniques on a seven-point Likert scale.

6.5.10 Discussion

The results of the study support acceptance of the hypothesis $\boldsymbol{H_{1\ C6.3}}$ (see Section 6.5) and show that the new technique improves overall text entry performance in terms of speed, accuracy, and user comfort. On average, entry speed increased by 9% and error rate decreased by 25% with the new pressure-based technique compared to the conventional technique. The 22% decrease in the corrective operations also provides indirect evidence in that users had to fix fewer mistakes with the new technique. However, post-hoc power analysis identified these effects as weak. Section 6.5.12 discusses this in detail. No significant effect of technique was identified on the accepted prediction rate. This is not unexpected as both techniques enable users to accept predictions by the same method—by tapping on the Space key.

However, there was somewhat significant effect of technique on the percentage of rejected and ignored predictions. This means users rejected (and ignored) more predictions with the new technique. As far as one can tell, most of these rejected predictions were instances where the prediction was not the desired word. There was also no significant effect of technique on the sum of the mental preparation and physical movement times. It can be seen as corroborating evidence that this factor did not contribute significantly to the observed differences. Therefore, one can speculate that the main difference was that users accepted fewer predicted words incorrectly with the new technique. This reduced the overall error percentage and error fixing time. The decrease in corrective operations also supports this observation.

Most users found text entry with the new technique easier than with the conventional one. Most also thought that their entry speed was higher and a majority believed to make fewer errors with the new technique. This means that the new technique is perceived as “faster” and “more accurate”. Therefore, it is not surprising that most users (83%) favored the new technique over the conventional one.

The default iPhone keyboard enables special character input by tap-holding the relevant keys for about a second. For instance, to input the character “Ē” one first holds down the “E” key for about a second to reveal a second level menu containing “Ē” and other diacritics. Then one selects the intended character by dragging the finger to the intended character in the menu. This feature was disabled, as users were not required to input special characters during the study. However, theoretically this feature could coexist with the proposed pressure-based technique, as the pressure-based technique uses a threshold of 200 ms for tap time, while the threshold used for special character input is roughly 1000 ms, as identified through video recordings.

6.5.11 Hybrid Pressure Detection Simulation

The results for the rate at which the hybrid technique relied on the pressure detection simulation criteria here support the observation from the first study that there are three distinct behaviors. Results of this study also indicated that a single criterion is not adequate for all users. This again highlights the utility of the hybrid pressure detection approach. The percentage of detections of extra pressure via the touch-point-based approach was larger (8% vs. 31% in the last study). Thus, one can state that in text entry almost one third of all extra pressure taps are best detected through a touch-point-based approach. Besides, user feedback data revealed that most users found the new technique to be "accurate". The hybrid method also sped up text entry significantly. The average tap times for regular and extra pressure were 117 and 390 ms, faster than Quick-release's 200 and 400 ms and Dwell's 1400 and 1700 ms (Brewster and Hughes, 2009), respectively.

6.5.12 Limitations

The ANOVAs identified a significant effect of technique on entry speed (WPM), error rate (TER), and correction operations (Backspace use). A post-hoc analysis detected a *small* effect size for all of these dependent variables (see Appendix A1). Additional post-hoc analysis revealed that the statistical power did not exceed the 0.80 threshold (see Appendix A4) for these dependent variables. In other words, at the *small* effect size level, there was less than adequate statistical power for WPM, TER, and correction operations (see Table 7). Although a larger sample size (*N*) may achieve a sufficiently strong statistical power for these dependent variables, it is less likely due to the *small* effect size.

Table 7. Detected effect size and measured statistical power.

Dependent Variable		Effect Size (η^2)	Power ($1-\beta$)
WPM		Small	<< 0.80
TER		Small	<< 0.80
Corrective Operation		Small	< 0.80 (= 0.69)
Additional	Rejected Prediction Rate	Medium	< 0.80 (= 0.69)
	Ignored Prediction rate	Medium	< 0.80 (= 0.43)

Similarly, the ANOVAs identified a significant effect of technique on additional dependent variables, such as rejected prediction rate (%) and ignored prediction rate (%). A post-hoc analysis detected a *medium* effect size for these two dependent variables (see Appendix A1). Further post-hoc analysis revealed that the statistical power did not exceed the 0.80 threshold (see Appendix A4) at the observed *medium* effect size level. In other words, there was less than adequate statistical power for these dependent variables (see Table 7), but it seems likely that a larger sample size (*N*) may achieve a sufficiently strong statistical power

for these dependent variables. However, due to the *medium* effect sizes, there is a possibility that these are not sufficiently strong effects after all. Appendix A4 explains the criteria used for calculating statistical power.

As the study recruited participants by using convenience sampling from the university community, the results may not generalize to a larger population. Nevertheless, an attempt was made to counteract this potential confound by recruiting not only university students but also instructors and staff.

6.6 Summary

This chapter presented a new pressure detection technique that combines the existing time- and touch-point-based approaches to detect pressure on standard touchscreens. Results of two independent user studies showed that the new hybrid technique distinguishes reliably between (at least) two pressure levels: regular with about 1 N, and extra with about 3 N. It then presented a new pressure-based predictive text entry technique that used the new pressure detection approach to enable users to bypass incorrect predictions by applying more pressure on the next target key. Results of a user study showed that when inputting short English phrases containing 10% non-dictionary words, the new technique increased entry speed by 9% and reduced errors by 25% compared to the conventional technique. However, post-hoc analysis identified these to be weak effects. Nonetheless, user feedback data showed that most users (75%) found the hybrid pressure detection technique accurate and most (83%) favor the pressure-based predictive text entry technique.

Chapter 7

Conclusions

This dissertation focused on the effect of errors in character-based text entry techniques from theoretical, behavioral, and practical standpoints. The theoretical part develops a mathematical model for predicting the cost of error correction for a given text entry technique by carefully observing human error correction behaviors. Towards that, a user study was first conducted to investigate the effect of different error correction conditions on popular text entry performance metrics. Results showed that the way errors are handled has a significant effect on all frequently used error metrics. The outcomes also provided an understanding of how users notice and correct errors. Building on this, the dissertation presented a new high-level and method-agnostic model for predicting the cost of error correction. Unlike existing models, it accounts for both human and system factors and is general enough to be used with most character-based techniques. An empirical study verified the model through measuring the effects of a faulty keyboard on text entry performance. The behavioral part of the dissertation investigated potential user adaptation to frequently occurring text entry errors by conducting two user studies. The studies explored user adaptation to a gesture recognizer's misrecognition errors. Results showed that users gradually adapt to misrecognition errors by replacing erroneous gestures with alternative ones, if available. Also, they adapt to a frequently misrecognized gesture faster if it occurs more frequently than other error-prone gestures. Finally, based on the findings from the theoretical and behavioral investigations, the practical part of the dissertation attempted to improve users' overall mobile text entry performance by developing a more efficient virtual keyboard. This part presented a new pressure-based text entry technique that does not require tapping outside the virtual keyboard to reject an incorrect or unwanted prediction. Instead, the technique requires users to apply extra pressure for the tap on the next target key. Results of a user study showed that for inputting short English phrases with 10% non-dictionary words, the new technique increases entry speed by 9% and decreases error rates by 25%, compared to the conventional technique. Also, almost all users favor the new technique over the conventional one. Together, the research presented in this dissertation gave more insight into on how errors affect text entry and also presented an improved text entry method. The following sections summarize the findings from each chapter.

This dissertation started with a review of the existing literature regarding character-based text entry and transcription typing in Chapter 2. The review covered all important character-based text entry techniques, along with their benefits and shortcomings, and mentioned the factors that influence text entry performances with these techniques, such as tactile feedback and size. Several alternative modalities used in text entry,

such as pressure and chords, were also discussed. The review collected data from the literature to understand where these techniques stand globally in terms of speed and accuracy. Towards this, the most popular text entry performance metrics were also analyzed. In addition, experimentally established cognitive, perceptual, and physical phenomena in transcription typing were summarized. Finally, the review provided a comprehensive evaluation of human and system errors and error correction behaviors through existing concepts, theories, and error classifications strategies.

Chapter 3 investigated if different error correction conditions used in user studies affects the ratings of popular performance metrics. Results of a user study showed that the way human errors are handled has a significant effect on all commonly used error metrics. Furthermore, results showed that the proportion of character- and word-level error corrections in text entry is almost balanced 50-50%.

Based on Chapter 2 and Chapter 3, Chapter 4 proposed a new model for predicting the cost of error correction for character-based text entry techniques. The model was verified against values derived from the literature and by conducting a user study. The model predicted and the results of the subsequent study verified that human text entry performance decreases with increasingly error prone systems. Potential applications of the new model were also discussed.

It is commonly assumed that users gradually adapt to a non-fatal bug or system error if it remains in the system for long enough. However, no empirical study explored this (so far hypothetical) phenomenon. Therefore, Chapter 5 conducted two user studies to investigate whether users adapt to a unistroke gesture recognizer's (injected) *misrecognition* errors or not. Results confirmed that users gradually adapt to such errors and this adaptation rate depends on how frequently they occur. That is, users adapt to an error faster if it occurs more frequently than the others. Based on this, several recommendations were made for gesture-based user interface designers that could enhance the accuracy and the usability of their techniques. This chapter also speculated on potential implications of this work.

Chapter 6 presented a new pressure detection simulation technique, which is a hybrid between the existing time-based and a new touch-point-based approach to detect pressure on standard touchscreens. The new technique was evaluated in two user studies that confirmed that it could distinguish (at least) between two pressure levels: regular (~1 N) and extra (~3 N). This approach was applied in a new predictive text entry technique for touchscreen-based mobile devices. It enabled users to bypass incorrect word predictions by applying more pressure on the next target key. The intention was to improve the overall text entry performance by reducing movement, target selection, and visual scan times. The new technique was compared with the conventional technique in a user study. Results showed that the new technique increases

entry speed by 9% and reduces errors by 25% compared to the conventional one. Post-hoc analysis, however, identified these to be weak effects. Yet almost all users favor the new technique over the conventional one.

The following chapter elaborates on several potential extensions of this work and ends with the expectation that the findings of this work will encourage researchers and practitioners to further investigate the effect of human and system errors in text entry and user interfaces.

Chapter 8
Future Work

Chapter 3 identified that the way error correction is handled in text entry user studies has significant effect on all major error rate metrics such as KSPC, ER, EKS, MSDER, and TER. Further examination will be carried out to investigate whether different phrase lengths have an effect on these metrics.

Chapter 4 presented a cost of error correction model for character-based text entry techniques. The model cannot be applied on word-based techniques. In the future, attempts will be made to generalize the model to word-at-a-time input techniques, such as speech and handwriting recognition. As the nature of error correction with these techniques is fairly similar to character-based techniques and also usually includes some form of *undo* operation, generalizing the model for such techniques seems feasible. In addition, the current model will be used to investigate what happens when various human and system parameters such as display and input time are changed. It will be interesting to examine the effect of various properties of input techniques such as average KSPC on the model too. Chapter 4 also presented the following parameters: input time ($\boldsymbol{T}^{h}_{input}$) and correction time ($\boldsymbol{T}_{correct}$). Further investigation will be carried out in the future to investigate if these parameters could be used as performance metrics in general text entry user studies.

Chapter 5 showed that users gradually adapt to a faulty unistroke gesture recognizer. That is, they start using an available alternative method for inputting gestures that are frequently misrecognized by the system. However, in the second study no clear pattern for adaptation to the alternative method for injected *misrecognition* error rates below 10% was observed before within the first 70 instances of each letter. Thus, in the future a longitudinal study is planned to observe adaptation to injected *misrecognition* error rates well below 10%. Also, this work focused on unistroke gesture recognizers, mainly for simplicity. In the future, further experiments may be conducted to investigate whether these results apply for multistroke recognizers as well or not. Results also indicated that users' adaptation to misrecognition errors is dependent on how frequently they occur—they adapt to an error faster, if it occurs more frequently. Based on this, a mathematical model could be developed to predict adaptation rates. Finally, one can speculate that the behavioral traits observed here for gesture-based text entry are relevant to user interface design in general, that is, users behave in a similar manner when interacting with any faulty user interface. This is also a topic worthy of investigation.

Chapter 6 presented a new hybrid method to simulate pressure detection in standard touchscreen-based devices. This work verified the effectiveness of the method for two different pressure levels. Future investigation will explore whether more than two pressure levels could be detected. The work presented in this chapter also applied the hybrid pressure detection technique to predictive text entry. In the evaluation users inputted short English phrases in stationary setting and in the portrait position. A future user study may verify the statistical strength of the differences for the new method relative to previous work. Moreover, the new technique will be evaluated in mobile settings, such as while walking, and in landscape mode with two-handed text entry. It may also be examined on relatively larger touchscreen devices such as tablets. Finally, the use of pressure in user interfaces will be further explored. One possibility is to use pressure for switching between various keyboard modes. For example, the use of three pressure levels to switch between lowercase, uppercase, and a special character layout in touchscreen-based virtual keyboards. Another possibility is the use of various pressure levels for authenticating mobile users. Such technique could enable users to use passwords that require tapping on the keys with various pressure levels, which may enhance mobile security.

References

Anderson, D., Bailey, C., and Skubic, M. (2004). Hidden Markov Model symbol recognition for sketch-based interfaces. *AAAI Fall Symposium 2004*, 15-21.

Anderson, J. R., Bothell, D. and Byrne, M. D. (2004) An integrated theory of the mind. *American Psychological Association: Psychological Review 4*, 111, 1036-1060.

Anthony, L. and Wobbrock, J. O. (2012) $N-protractor: a fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012* (GI '12). Canadian Information Processing Society, Toronto, Ontario, Canada, Canada, 117-120.

Arif, A. S., Pahud, M., Hinckley, K., and Buxton, W. (2014) Experimental study of stroke shortcuts for a touchscreen keyboard with gesture-redundant keys removed. In *Proceedings of Graphics Interface 2014* (GI '14). Canadian Information Processing Society, Toronto, Ontario, Canada, 43-50.

Arif, A. S. and Stuerzlinger, W. (2012) How do users adapt to a faulty system? In *CHI '12 Workshop on Designing and Evaluating Text Entry Methods*. 11-14.

Arif, A. S. and Stuerzlinger, W. (2013). Evaluation of a new error prevention technique for mobile touchscreen text entry. In *Proceedings of the 25th Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction* (OzCHI '13). ACM, New York, NY, USA, 397-400.

Arif, A. S. (2012) A survey on mobile text entry handedness: how do users input text on handheld devices while nomadic? In *Proceedings of the 4th International Conference on Intelligent Human Computer Interaction* (IHCI '12). IEEE, Washington, DC, USA, 266-271.

Arif, A. S. and Sylla, C. (2013) A comparative evaluation of touch and pen gestures for adult and child users. In *Proceedings of the 12th International Conference on Interaction Design and Children* (IDC '13). ACM, New York, NY, USA, 392-395.

Arif, A. S., Iltisberger, B., and Stuerzlinger, W. (2011) Extending mobile user ambient awareness for nomadic text entry. In *Proceedings of the 23rd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction* (OzCHI '11). ACM, New York, NY, USA, 21-30.

Arif, A. S., Lopez, M. H., and Stuerzlinger, W. (2010) Two new mobile touchscreen text entry techniques. *Poster at the 36th Graphics Interface Conference* (GI '10). CEUR-WS.org/Vol-588, 22-23.

Ayres, R. U. and Martinás, K. (2005) 120 wpm for very skilled typist. In *On the Reappraisal of Microeconomics: Economic Growth and Change in a Material World*. Edward Elgar Publishing Limited, Cheltenham, UK, 41-41.

Barrett, J. and Krueger, H. (1994) Performance effects of reduced proprioceptive feedback on touch typists and casual users in a typing task. *Behaviour & Information Technology 13*, 6, 373-381.

Beard, D. V., Smith, D. K., and Denelsbeck, K. M. (1996) Quick and dirty GOMS: A case study of computed tomography interpretation. *Human-Computer Interaction 11*, 2, 157-180.

Bender, G. T. (1999) Touch screen performance as a function of the duration of auditory feedback and target size. Doctoral Dissertation. Wichita State University, Wichita, KS, USA.

Benko, H., Wilson, A. D., and Baudisch, P. (2006) Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06), R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. Olson (Eds.). ACM, New York, NY, USA, 1263-1272.

Boring, S., Ledo, D., Chen, X. A., Marquardt, N., Tang, A., and Greenberg, S. (2012) The fat thumb: using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services* (MobileHCI '12). ACM, New York, NY, USA, 39-48.

Bothe, S., Gärtner, T., and Wrobel, S. (2010) On-line handwriting recognition with parallelized machine learning algorithms. *Lecture Notes in Computer Science 6359*, 82-90.

Bothell, D. (2012) ACT-R 6.0 reference manual: working draft. ACT-R Research Group, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA, USA, 131. Retrieved 2013, July 12. http://act-r.psy.cmu.edu/actr6/reference-manual.pdf

Brewster, S. (2002) Overcoming the Lack of Screen Space on Mobile Computers. *Personal Ubiquitous Computing 6*, 3, 188-205.

Brewster, S. A. and Hughes, M. 2009. Pressure-based text entry for mobile devices. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '09). ACM, New York, NY, USA, Article 9, 4 pages.

Brodbeck, F. C., Zapf, D., Prumper, J., Frese, M. (1993) Error handling in office work with computers: a field study. *Journal of Occupational & Organizational Psychology 66*, 4 (1993), 303-317.

Butsch, R. L. C. (1932) Eye movements and the eye-hand span in typewriting. *Journal of Educational Psychology 23*, 2, 104-121.

Butts, L. and Cockburn, A. (2002) An evaluation of mobile phone text input methods. *Australian Computer Science Communications 24*, 4, 55-59.

Buxton, W. (1995). Chapter 7: touch, gesture & marking. In *Readings in Human Computer Interaction: Toward the Year 2000*, R. M. Baecker, J. Grudin, W. Buxton and S. Greenberg, S. (Eds.). Morgan Kaufmann, San Francisco, CA, USA.

Buxton, W. (2013, March 19) Multi-touch systems I have known and loved. Retrieved 2013, June 28. http://www.billbuxton.com/multitouchOverview.html

Buxton, W. and Kurtenbach, G. (1995) Graphical keyboard. US Patent 6 094 197, May 17, 1995.

Buxton, W. Hill, R. and Rowley, P. (1985) Issues and techniques in touch-sensitive tablet input. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '85). ACM, New York, NY, USA, 215-224.

Cao, X. and Balakrishnan, R. (2003) VisionWand: interaction techniques for large displays using a passive wand tracked in 3D. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (UIST '03). ACM, New York, NY, USA, 173-182.

Cao, X. and Balakrishnan, R. (2005) Evaluation of an on-line adaptive gesture interface with command prediction. In *Proceedings of Graphics Interface 2005* (GI '05). Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 187-194.

Cao, X. and Zhai, S. (2007) Modeling human performance of pen stroke gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 1495-1504.

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

Card, S. K., Moran, T. P., and Newell, A. (1980) The keystroke-level model for user performance time with interactive systems. *Communications of the ACM* 23, 7, 396-410.

Carroll, J. M. and Rosson, M. B. (1987) Paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer interaction*, J. M. Carroll (Ed.). MIT Press, Cambridge, MA, USA, 80-111.

Castellucci, S. J. and MacKenzie, I. S. (2008) Graffiti vs. Unistrokes: an empirical comparison. In *Proceeding of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 305-308.

Castellucci, S. J. and MacKenzie, I. S. (2008) Unigest: text entry using three degrees of motion. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '08). ACM, New York, NY, USA, 3549-3554.

CBC. (2013, May 1) BlackBerry timeline: a tech titan's roller coaster ride. *CBC News*. Retrieved 2013, June 29. http://www.cbc.ca/news/interactives/timeline-rim

Cechanowicz, J., Irani, P., and Subramanian, S. (2007) Augmenting the mouse with pressure sensitive input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 1385-1394.

Cho, M. G. (2006) A new gesture recognition algorithm and segmentation method of Korean scripts for gesture-allowed ink editor. *Inf. Sci.* 176, 9 (May 2006), 1290-1303.

Choi, E., Bang, W., Cho, S., Yang, J., Kim, D., and Kim, S. (2005) Beatbox music phone: gesture-based interactive mobile phone using a tri-axis accelerometer. In *Proceedings of the International Conference on Industrial Technology* (ICIT '13). IEEE, Washington, DC, USA, 97-102.

Clarkson, E., Clawson, J., Lyons, K., and Starner, T. (2005) An empirical study of typing rates on mini-QWERTY keyboards. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05). ACM, New York, NY, USA, 1288-1291.

Clawson, J., Lyons, K., Rudnick, A., and Iannucci, Jr., R. A., and Starner, T. (2008) Automatic whiteout++: correcting mini-QWERTY typing errors using keypress timing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 573-582.

Cockburn, A., Kristensson, P. O., Alexander, J., and Zhai, S. (2007) Hard lessons: effort-inducing interfaces benefit spatial learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 1571-1580.

Cohen, J. (1988). *Statistical power analysis for the behavior sciences (2nd Ed.)*. Lawrence Erlbaum, Hillsdale, NJ, USA, 283.

Cohen, J. (1992) Statistical power analysis. *Current directions in psychological science 1*, 3, 98-101.

Colle, H. A. and Hiszem, K. J. (2004) Standing at a kiosk: effects of key size and spacing on touch screen numeric keypad performance and user preference. *Ergonomics 47*, 13, 1406-1423.

Cooper, W. E. (1983) Introduction. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, USA, 1-38.

Coover, J. E. (1923). A method of teaching typewriting based upon a psychological analysis of expert typewriting. *Addresses and Proceedings – National Education Association of the United States 61*, 561-567.

Craik, F. I. M. and Lockhart, R. S. (1972) Levels of processing: a framework for memory research. *Journal of Verbal Learning and Verbal Behavior 11*, 6, 671-684.

Craik, F. I. M. and Tulving, E. (1975) Depth of processing and the retention of words in episodic memory. *Journal of Experimental Psychology: General 104*, 3, 268-294.

Cunningham, J. B. and McCrum-Gardner, E. (2007) Power, effect and sample size using GPower: practical issues for researchers and members of research ethics committees. *Evidence Based Midwifery 5*, 132-136.

Deininger, R. L. (1960) Human factors engineering studies of the design and use of pushbutton telephone sets. *Bell System Technical Journal 39*, 995-1012.

Dietz, P. H., Eidelson, B., Westhues, J., and Bathiche, S. (2009) A practical pressure sensitive computer keyboard. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09). ACM, New York, NY, USA, 55-58.

Drury, C. G. and Hoffmann, E. R. (1992) A model for movement time on data-entry keyboards. *Ergonomics 35*, 129-147.

Dunlop, M. D. and Crossan, A. (2000) Predictive text entry methods for mobile phones. *Personal & Ubiquitous Computing 4*, 2-3, 134-143.

Dvorak, A., Merrick, N. L., Dealey, W. L., and Ford, G. C. (1936) *Typewriting Behavior: Psychology Applied to Teaching and Learning Typewriting*. American Book Company, New York, NY, USA.

Ehret, B. D. (2002) Learning where to look: location learning in graphical user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02), 211-218.

Evans, A. and Wobbrock, J. (2012) Taming wild behavior: the input observer for text entry and mouse pointing measures from everyday computer use. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 1947-1956.

Faul, F., Erdfelder, E., Lang, A., and Buchner, A. (2007) G*Power 3: a flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods 39*, 2, 175-191.

Fendrick, P. (1937). Hierarchical skills in typewriting. *Journal of Educational Psychology 28*, 8, 609-620.

Forlines, C. and Shen, C. (2005) DTLens: multi-user tabletop spatial data exploration. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (UIST '05). ACM, New York, NY, USA, 119-122.

Fox. J. G. and Stansfield, R. G. (1964). Digram keying times for typists. *Ergonomics 7*, 3, 317-320.

Frese, M. and Sabini, J. (1985) *Goal Directed Behavior: The Concept of Action in Psychology*. Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, USA.

Garay-Vitoria, N. and Abascal, J. (2006) Text prediction systems: a survey. *Universal Access in the Information Society 4*, 3, 188-203.

Gentner, D. R. (1982) Evidence against a central control model of timing in typing. *Journal of Experimental Psychology: Human Perception and Performance 8*, 793-810.

Gentner, D. R. (1983[a]) Keystroke timing in transcription typing. In W. E. Cooper, Ed. *Cognitive Aspects of Skilled Typewriting*. Springer-Verlag, New York, NY, USA, 95-120.

Gentner, D. R. (1983[b]) The acquisition of typewriting skill. *Acta Psychologica 54*, 1-3 (1983), 233-248.

Gentner, D. R., Grudin, J. T., Larochelle, S, Norman, D. A., and Rumelhart, D. E. (1983) A glossary of terms including a classification of typing errors. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, 39-43.

Goldberg, D. (1997) Unistrokes for computerized interpretation of handwriting. US Patent 5 596 656, January 21, 1997.

Goldberg, D. and Richardson, C. (1993) Touch-typing with a stylus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '93). ACM, New York, NY, USA, 80-87.

Gong, J., Haggerty, B., and Tarasewich, P. (2005) An enhanced Multitap text entry method with predictive next-letter highlighting. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05). ACM, New York, NY, USA, 1399-1402.

Gopher, D. and Raij, D. (1988) Typing with a two-hand chord keyboard: will the QWERTY become obsolete? *IEEE Transactions on Systems, Man and Cybernetics 18*, 4, 601-609.

Gopher, D., Hilsernath, H., and Raij, D. (1985) Steps in the development of a new data entry device based upon two hand chord keyboard. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 29*, 2, 132-136.

Graham-Rowe, D. (2010, January 27) Mobile touch screens could soon feel the pressure. MIT Technology Review. Retrieved 2013, March 2. http://www.technologyreview.com/news/417246/mobile-touch-screens-could-soon-feel-the-pressure

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009) A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 5, 855-868.

Gray, W. D., John, B. E. and Atwood, M. E. (1992) The precis of Project Ernestine or an overview of a validation of GOMS. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '92). ACM, New York, NY, USA, 307-312.

Grudin, J. T, and Larochelle, S. (1982) Digraph frequency effects in skilled typing. Technical Report. University of California, Center for Human Information Processing, San Diego, CA, USA, CHIP 110.

Grudin, J. T. (1983[a]) Error patterns in novice and skilled transcription typing. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, USA, 121-143.

Grudin, J. T. (1983[b]). Non-hierarchic specification of components in transcription typing. *Ada Psychologica 54*, 1-3, 249-262.

Guimbretière, F., Stone, M., and Winograd, T. (2001) Fluid interaction with high-resolution wall-size displays. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (UIST '01). ACM, New York, NY, USA, 21-30.

Gunawardana, A, Paek, T., and Meek, C. (2010) Usability guided key-target resizing for soft keyboards. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (IUI '10). ACM, New York, NY, USA, 111-118.

Gupta, A., Milanesi, C., Cozza, R., and Lu, C. K. (2013, May 2013) Market share analysis: mobile phones, worldwide, 1Q13. Market analysis and statistics, Gartner, Stamford, CT, USA, 1Q13.

Gustafson, S., Bierwirth, D., and Baudisch, P. (2010) Imaginary interfaces: spatial interaction with empty hands and without visual feedback. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 3-12.

Hafner, K. (2008, March 25) Did Bill Gates really say that? *The New York Times: Bits*. Retrieved 2012, April 2. http://bits.blogs.nytimes.com/2008/03/25/did-bill-gates-really-say-that

Hashimoto, M. and Togasi, M. (1995) A virtual oval keyboard and a vector input method for pen-based character input. In *Conference Companion on Human Factors in Computing Systems* (CHI '95), I. Katz, R. Mack, and L. Marks (Eds.). ACM, New York, NY, USA, 254-255.

Henze, N., Rukzio, E., and Boll, S. (2012) Observational and experimental investigation of typing behaviour using virtual keyboards for mobile devices. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 2659-2668.

Heo, S. and Lee, G. (2011) Forcetap: extending the input vocabulary of mobile touch screens by adding tap gestures. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (MobileHCI '11). ACM, New York, NY, USA, 113-122.

Herot, C. F. and Weinzapfel, G. (1978) One-point touch input of vector information for computer displays. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '78). ACM, New York, NY, USA, 210-216.

Hershman, R. L. and Hillix, W. A. (1965). Data processing in typing: typing rate as a function of kind of material and amount exposed. *Human Factors 7*, 5, 483-492.

Hesseldahl, A. (2002, September 18) Typing on the table. *Forbes*. Retrieved 2013, June 28. http://www.forbes.com/2002/09/18/0918tentech.html

Hertzberg, E. (2011) Chapter Four: Xerox v. Palm – Litigation and License. In *Exclusive Rights: Issues in Intellectual Property Law*. AuthorHouse, Bloomington, IN, USA, 54-72.

Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., and Smith, M. (2004) Stitching: pen gestures that span multiple displays. In *Proceedings of the working conference on Advanced visual interfaces* (AVI '04). ACM, New York, NY, USA, 23-31.

Hoffmann, A., Spelmezan, D., and Borchers, J. (2009) TypeRight: a keyboard with tactile error prevention. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 2265-2268.

Hoggan, E., Brewster, S. A., and Johnston, J. (2008) Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, 1573-1582.

Holleis, P., Otto, F., Hussmann, H., and Schmidt, A. (2007) Keystroke-level model for advanced mobile phone interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 1505-1514.

Hoober, S. (2013, February 18) How do users really hold mobile devices? *UXmatters*. Retrieved 2013, February 18. http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

How, Y. and Kan, M. (2005) Optimizing predictive text entry for short message service on mobile phones. In *Proceedings of Human Computer Interfaces International* (HCII '05). Lawrence Erlbaum, Hillsdale, NJ, USA.

Hu, J. Brown, M. K., and Turin, W. (1996) HMM based online handwriting recognition. *Pattern Analysis & Machine Intelligence 18*, 10, 1039-1045.

Hudson, S. E., John, B. E., Knudsen, K., and Byrne, M. D. (1999) A tool for creating predictive performance models from user interface demonstrations. In *Proceedings of the 12th annual ACM symposium on User interface software and technology* (UIST '99). ACM, New York, NY, USA, 93-102.

Hwang, S. and Wohn, K. (2012) PseudoButton: enabling pressure-sensitive interaction by repurposing microphone on mobile device. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '12). ACM, New York, NY, USA, 1565-1570.

Inhoff, A. W. and Wang, J. (1992) Encoding of text, manual movement planning, and eye hand coordination during copytyping. *Journal of Experimental Psychology: Human Perception and Performance* 18, 2 (1992), 437-448.

Isokoski, P. (1999) A minimal device-independent text input method. Technical Report. Department of Computer Science, University of Tampere, Finland, Report A-1999-14.

Isokoski, P. (2004) Performance of menu-augmented soft keyboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 423-430.

James, C. L. and Reischel, K. M. (2001) Text input for mobile devices: comparing model prediction to actual performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '01). ACM, New York, NY, USA, 365-371.

John, B. E. (1988) Contributions to engineering models of human-computer interaction. Doctoral Dissertation. Department of Psychology, Carnegie-Mellon University, PA, USA.

John, B. E. (1993) A quantitative model of expert transcription typing. Technical Report. Department of Computer Science, Carnegie-Mellon University, PA, USA, CMU-CS-93-120.

John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. (2004) Predictive human performance modeling made easy. . In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 455-462.

Jong, S. d., Kirkali, D., Schraffenberger, H., Jillissen, J., Rooij, A. d., and Terpstra, A. (2010) One-press control: a tactile input method for pressure-sensitive computer keyboards. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '10). ACM, New York, NY, USA, 4261-4266.

Juhlin, O. and Önnevall, E. (2013) On the relation of ordinary gestures to TV screens: general lessons for the design of collaborative interactive techniques. In *Proceedings of the 2013 ACM annual conference on Human factors in computing systems* (CHI '13). ACM, New York, NY, USA, 919-930.

Kaaresoja, T. and Linjama, J. (2005) Perception of short tactile pulses generated by a vibration motor in a mobile phone. In Proceedings of the First Joint Eurohaptics Conference & Symposium on Haptic Interfaces for Virtual Environment & Teleoperator Systems (WHC '05). IEEE, Washington, DC, USA, 471-472.

Kaaresoja, T., Brown, L. M., and Linjama, J. (2006) Snap-crackle-pop: Tactile feedback for mobile touch screens. In *Proceeding of the EuroHaptics Conference* (EuroHaptics '06), 565-566.

Kallio, S., Kela, J., and Mantyjarvi, J. (2003) Online gesture recognition system for mobile interaction. In *Proceedings of the International Conference on Systems, Man, and Cybernetics* (SMC '03). IEEE, Washington, DC, USA, 2070- 2076.

Kano, A., Read, J. C., Dix, A., and MacKenzie, I. S. (2007) ExpECT: An expanded error categorisation method for text input. In *Proceedings of the 21st British CHI Group Annual Conference on HCI 2007: People & Computers XXI: HCI. But Not As We Know It - Volume 1*. British Computer Society, Swinton, UK, 147-156.

Karam, M. and schraefel, m. c. (2006) Investigating user tolerance for errors in vision-enabled gesture-based interactions. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (AVI '06). ACM, New York, NY, USA, 225-232.

Karat, C., Halverson, C., Horn, D. and Karat, J. (1999) Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '99). ACM, New York, NY, USA, 568-575.

Karlson, A. K., Bederson, B. B., and SanGiovanni, J. (2005) AppLens and LaunchTile: two designs for one-handed thumb use on small devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 201-210.

Kieras, D. E. (1988) Towards a practical GOMS model methodology for user interface design. In M. Helander (Eds.) *Handbook of Human-Computer Interaction*. Elsevier, Amsterdam, Netherlands.

Kieras, D. E. (1993) Using the keystroke-level model to estimate execution times. Technical Report. University of Michigan, Ann Arbor, MI, USA.

Kim, S., Son, J., Lee, G., Kim, H., and Lee, W. (2013) TapBoard: making a touch screen keyboard more touchable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 553-562.

Kinkead, R. (1975). Typing speed, keying rates, and optimal keyboard layouts. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 19*, 2, 159-161.

Költringer, T. and Grechenig, T. (2004) Comparing the immediate usability of graffiti 2 and virtual keyboard. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '04). ACM, New York, NY, USA, 1175-1178.

Korth, H. (1995) Method and device for optical input of commands or data. US Patent 0 554 492, April 21, 1995.

Koskinen, E., Kaaresoja, T., and Laitinen, P. (2008) Feel-good touch: finding the most pleasant tactile feedback for a mobile touch screen button. In *Proceedings of the 10th International Conference on Multimodal Interfaces* (ICMI '08). ACM, New York, NY, USA, 297-304.

Kristensson, P. and Zhai, S. (2004) SHARK^2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST '04). ACM, New York, NY, USA, 43-52.

Kurtenbach, G. and Buxton, W. (1993) The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (CHI '93). ACM, New York, NY, USA, 482-487.

Labahn, G., Lank, E., Marzouk, M., Bunt, A., MacLean, S., and Tausky, D. (2008) MathBrush: a case study for pen-based interactive mathematics. In *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling* (SBM '08), C. Alvarado and M. Cani (Eds.). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 143-150.

Lahy, J. M. (1924). Motion study in typewriting: a record of experiments. In *Studies and Reports 3*. International Labour Office, Geneva, Switzerland.

LaLomia, M. (1994) User acceptance of handwritten recognition accuracy. In *Proceeding of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '94). ACM, New York, NY, USA, 107-108.

Larochelle, S. (1983). A comparison of skilled and novice performance in discontinuous typing. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, USA, 67-94.

Larochelle, S. (1984). Some aspects of movements in skilled typewriting. In *Attention & Performance X*, H. Bouma and D. G. Bouwis (Eds.). Lawrence Erlbaum, Hillsdale, NJ, USA, 43-54.

Lee, H., Verma, B., Li, M., and Rahman, A. (2012) Machine learning techniques in handwriting recognition: problems and solutions. In *Machine Learning Algorithms for Problem Solving in Computational Applications: Intelligent Techniques*, S. Kulkarni (Ed.). IGI Global, Hershey, PA, USA.

Lee, S. and Zhai, S. (2009) The performance of touch screen soft buttons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, 309-318.

Leedham, C., Downton, A., Brooks, C. & Newell, A. (1984) OnLine acquisition of Pitman's handwritten shorthand as a means of rapid data entry. In *Proceedings of the IFIP Conference on Human-Computer Interaction* (INTERACT '84), Vol. 2, B. Shackel (Ed.). Elsevier Science, London, UK, 86-91.

Leonard, J. A. and Newman, R. C. (1965) Formation of higher habits. *Nature 203*, 550-551.

Lessenberry, D. D. (1928) Analysis of Errors. Syracuse, NY, USA. Reprinted in Dvorak, Merrick, Dealy, and Ford. (1936) *Typewriting behavior*. American Book Company, New York, NY, USA.

Levenshtein, V. I. (1966) Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady* 10, 707-710.

Levy, D. (2002) The Fastap keypad and pervasive computing. In *Proceedings of the the First International Conference on Pervasive Computing* (PerCom '02). Springer-Verlag, Berlin, Heidelberg, 58-68.

Lewis, J. R., Potosnak, K. M., and Magyar, R. L. (1997) Keys and keyboards. In *Handbook of Human-Computer Interaction*, M. G. Helander, T. K. Landauer, and P. V. Prabhu (Eds.). Elsevier Science, Amsterdam, Netherlands, 1285-1315.

Li, F. C. Y., Guy, R. T., Yatani, K., and Truong, K. N. (2011) The 1line keyboard: a QWERTY layout in a single line. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 461-470.

Li, Y. (2010[a]) Gesture search: a tool for fast mobile data access. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (UIST '10). ACM, New York, NY, USA, 87-96.

Li, Y. (2010[b]) Protractor: a fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 2169-2172.

Liebowitz, S. J. and Margolis, S. E. (1990) The fable of the keys. *Journal of Law & Economics 30*, 1, 1-26.

Liebowitz, S. J. and Margolis, S. E. (1996, June) Typing errors: the myth of Qwerty economics. *Reason (Magazine)*. Retrieved 2013, June 28. http://reason.com/archives/1996/06/01/typing-errors

Liwicki, M., Graves, A., and Bunke, H. (2012) Neural networks for handwriting recognition. *Computational intelligence Paradigms in Advanced Pattern Classification: Studies in Computational Intelligence 386*, 5-24.

Logan, G. D. (1983). Time, information, and the various spans in typewriting. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, USA, 197-224.

Logan, G. D. and Crump, M. J. C. (2009). The left hand doesn't know what the right hand is doing: the disruptive effects of attention to the hands in skilled typewriting. *Psychological Science 20*, 10, 1296-1300.

Logan. G. D. (1982). On the ability to inhibit complex movements: a stop-signal study of typewriting. *Journal of Experimental Psychology: Human Perception and Performance 8*, 6, 778-792.

Long, J. (1976) Visual feedback and skilled keying: differential effects of masking the printed copy and the keyboard. *Ergonomics 19*, 1, 93-110.

Lyons, K., Plaisted, D., and Starner, T. (2004[a]) Expert chording text entry on the Twiddler one-handed keyboard. In *Proceedings of the 8th International Symposium on Wearable Computers* (ISWC '04). IEEE, Washington, DC, USA, 94-101.

Lyons, K., Starner, T., and Gane, B. (2006) Experimental evaluations of the Twiddler one-handed chording mobile keyboard. *Human-Computer Interaction* 21, 4, 343-392.

Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., and Looney, E. W. (2004[b]) Twiddler typing: one-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, 671-678.

MacKenzie, I. S. (1991). Fitts' law as a performance model in human-computer interaction. Doctoral dissertation. University of Toronto, Toronto, Ontario, Canada.

MacKenzie, I. S. (2002) KSPC (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction* (Mobile HCI '02), Fabio Paternò (Ed.). Springer-Verlag, London, UK, 195-210.

MacKenzie, I. S. (2013). Designing HCI Experiments. In *Human-computer interaction: an empirical research perspective*. Morgan Kaufmann, Waltham, MA, USA, 157-189.

MacKenzie, I. S. and Soukoreff, R. W. (2002[a]). A model of two-thumb text entry. In *Proceedings of Graphics Interface 2002* (GI '02). Canadian Information Processing Society, Toronto, Ontario, Canada, 117-124.

MacKenzie, I. S. and Soukoreff, R. W. (2002[b]). Text entry for mobile computing: models and methods, theory and practice. *Human-Computer Interaction*, 17, 147-198.

MacKenzie, I. S. and Soukoreff, R. W. (2003) Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, 754-755.

MacKenzie, I. S., and Zhang, S. X. (2001). An empirical investigation of the novice experience with soft keyboards. *Behaviour & Information Technology 20*, 411-418.

MacKenzie, I. S., Chen, J., and Oniszczak, A. (2006) Unipad: single stroke text entry with language-based acceleration. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles* (NordiCHI '06), A. Mørch, K. Morgan, T. Bratteteig, G. Ghosh, and D. Svanaes (Eds.). ACM, New York, NY, USA, 78-85.

MacKenzie, I. S., Kober, H., Smith, D., Jones, T., and Skepner, E. (2001) LetterWise: prefix-based disambiguation for mobile text input. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software & Technology* (UIST '01). ACM, New York, NY, USA, 111-120.

MacKenzie, I. S., Zhang, S. X., and Soukoreff, R. W. (1999) Text entry using soft keyboards. *Behaviour & Information Technology 18*, 4, 235-244.

Mankoff, J. and Abowd, G. D. (1998) Cirrin: a word-level unistroke keyboard for pen input. In *Proceedings of the 11th annual ACM symposium on User interface software and technology* (UIST '98). ACM, New York, NY, USA, 213-214.

Mankoff, J. and Abowd, G. D. (1999) Error correction techniques for handwriting, speech, and other ambiguous or error prone systems. Technical Report. Georgia Institute of Technology, Atlanta, GA, USA, GVU-99-18.

Masui, T. (1998) An efficient text input method for pen-based computers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '98), C-M. Karat, A. Lund, J. Coutaz, and J. Karat (Eds.). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 328-335.

Mazur, J. E. and Hastie, R. (1978) Learning as accumulation: A reexamination of the learning curve. *Psychological Bulletin 85*, 6, 1256-1274.

McCallum, D. C., Mak, E. Irani, P., and Subramanian, S. (2009) PressureText: pressure input for mobile phone text entry. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '09). ACM, New York, NY, USA, 4519-4524.

McDermott-Wells, P. (2006) Evaluation of three stylus-based text entry methods on a Pocket PC™ mobile device. In *Proceedings of the annual IEEE Region 3 Technical, Professional, and Student Conference* (SoutheastCon '06). IEEE, Washington, DC, USA, 228-234.

Miller, G. A., Galanter, E., and Pribram, K. H. (1960) *Plans and the Structure of Behavior*. Adams Bannister Cox Pubs, New York, NY, USA.

Miller, L. A. and Thomas Jr., J. C. (1977) Behavioral issues in the use of interactive systems. *International Journal of Man-Machine Studies 9*, 5, 509-536.

Mizobuchi, S., Mori, K., Ren, X., Michiaki, Y. (2002) An empirical study of the minimum required size and the minimum number of targets for pen input on the small display. *Human Computer Interaction with Mobile Devices: Lecture Notes in Computer Science 2411*, 184-194.

Mizobuchi, S., Terasaki, S., Keski-Jaskari, T., Nousiainen, J., Ryynanen, M., and Silfverberg, M. (2005) Making an impression: force-controlled pen input for handheld devices. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05). ACM, New York, NY, USA, 1661-1664.

Myers, C. S. (1980) A comparative study of several dynamic time warping algorithms for speech recognition. Doctoral Dissertation. Massachusetts Institute of Technology, Department of Electrical Engineering & Computer Science, Cambridge, MA, USA.

Newell, A. and Rosenbloom, P. S. (1981) Mechanisms of skill acquisition and the law of practice. In *Cognitive Skills and Their Acquisition*, J. R. Anderson (Ed.). Lawrence Erlbaum, Hillsdale, NJ, USA.

Nielsen. (2012, July 7) Two thirds of new mobile buyers now opting for smartphones. Retrieved 2013, June 28. http://www.nielsen.com/us/en/newswire/2012/two-thirds-of-new-mobile-buyers-now-opting-for-smartphones.html

Norman, D. A. (1981) Categorization of action slips. *Psychological Review* 88, 1, 1-15.

Noyes, J. (1983) The QWERTY keyboard: a review. *International Journal of Man-Machine Studies 18*, 3, 265–281.

Nurmi, M. User Interface. US Patent 0256 807, October 15, 2009.

O'Brien, M. A., Rogers, W. A., and Fisk, A. D. (2008) Text entry interface design requirements at a glance. *Ergonomics in Design: The Quarterly of Human Factors Applications 16*, 4, 16-22.

Olsen, R. A. and Murray, R. A. (1976). Finger motion analysis in typing of texts of varying complexity. In Proceedings of the 6th Congress of the International Ergonomics Association (IEA '76), 446-450.

Oslry, D. J. (1983). Determinants of interkey times in typing. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, USA, 225-246.

Parhi, P., Karlson, A. K., and Bederson, B. B. (2006) Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services* (MobileHCI '06). ACM, New York, NY, USA, 203-210.

Patel, S. N., Pierce, J. S., and Abowd, G. D. (2004) A gesture- based authentication scheme for untrusted public terminals. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software & Technology* (UIST '04). ACM, New York, NY, USA, 157-160.

Pavlovych, A. and Stuerzlinger, W. (2003) Less-tap: a fast and easy-to-learn text input technique for phones. In *Proceedings of Graphics Interface 2003* (GI '03). A. K. Peters, Natick, MA, USA, 97-104.

Pavlovych, A. and Stuerzlinger, W. (2004) Model for non-expert text entry speed on 12-button phone keypads. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 351-358.

Pittman, J. A. (1991) Recognizing handwritten text. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '91), Scott P. Robertson, Gary M. Olson, and Judith S. Olson (Eds.). ACM, New York, NY, USA, 271-275.

Plamondon, R. and Srihari, S. N. (2000) Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 1, 63-84.

Pogue, D. (2007, June 27) iPhone keyboard secrets. *The New York Times*. Retrieved 2010, January 12. http://pogue.blogs.nytimes.com/2007/06/27/iphone-keyboard-secrets

Rabbitt, P. (1978). Detection of errors by skilled typists. *Ergonomics 21*, 11, 945-958.

Raisamo, R. Evaluating different touched-based interaction techniques in a public information kiosk. In *Proceedings of the 21st Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction* (OzCHI '99). Human Factors and Ergonomics Society of Australia Inc., Baulkham Hills, NSW, Australia, 169-171.

Ramos, G., Boulos, M., and Balakrishnan, R. (2004) Pressure widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 487-494.

Randell, B., Wilkes, M. V., and Ceruzzi, P. E. (2003) Digital computers history. In *Encyclopedia of Computer Science*, A. Ralston, E. D. Reilly, and D. Hemmendinger (Eds.). John Wiley & Sons Ltd., Chichester, West Sussex, UK, 545-570.

Rasmussen, J. (1984) *Human error data. Facts or fiction.* Technical Report. Risø National Laboratory, Roskilde, Denmark, Rep Risø-M-2499, 22.

Rayner, K. (1998) Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124, 3, 372-422.

Read, J. C., MacFarlane, S. J., and Casey, C. (2001) Measuring the usability of text input methods for children. In *Proceedings of the 15th British CHI Group Annual Conference on HCI 2001: People & Computers XV*. Springer-Verlag, London, UK, 559-572

Reason, J. (1990) *Human Error*. Cambridge University Press, New York, NY, USA.

Riche, Y., Riche, N. H., Isenberg, P., and Bezerianos, A. (2010) Hard-to-use interfaces considered beneficial (some of the time). In *CHI '10 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '10). ACM, New York, NY, USA, 2705-2714.

Robertson, S. P. and Black, J. B. (1986) Structure and development of plans in computer text editing. *Human-Computer Interaction 2*, 3, 201-226.

Roeber, H., Bacus, J., and Tomasi, C. (2003) Typing in thin air: the Canesta projection keyboard – a new method of interaction with electronic devices. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '03). ACM, New York, NY, USA, 712-713.

Rothkopf, E. Z. (1980). Copying span as a measure of the information burden in written language. *Journal of Verbal Learning and Verbal Behavior 19*, 5, 562–572.

Rubine, D. (1991) Specifying gestures by example. *SIGGRAPH Computer Graphics 25*, 4, 329-337.

Rumelhart, D. E. and Norman. D. A. (1983). Studies of typing from the LNR Research Group. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, New York, NY, USA, 45-66.

Rumelhart. D. E. and Norman, D. A. (1982). Simulating a skilled typist: a study of skilled cognitive-motor performance. *Cognitive Science 6*, 1, 1-36.

Sad, H. H. and Poirier, F. (2009) Using pictographic representation, syntactic information and gestures in text entry. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, J. A. Jacko (Ed.). Springer-Verlag, Berlin, Heidelberg, Germany, 735-744.

Sager, I. (2012, June 29) Before iPhone and Android came Simon, the first smartphone. *The Business Week*. Retrieved 2013, June 28. http://www.businessweek.com/articles/2012-06-29/before-iphone-and-android-came-simon-the-first-smartphone

Salthouse, T. A. (1984[a]) Effects of age and skill in typing. *Journal of Experimental Psychology: General 113*, 3, 345-371.

Salthouse, T. A. (1984[b]). The skill of typing. *Scientific American* 250, 128-135.

Salthouse, T. A. (1985) Anticipatory processing in transcription typing. *Journal of Applied Psychology 70*, 264-271.

Salthouse, T. A. (1986) Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological Bulletin 99*, 3, 303-319.

Salthouse, T. A. and Saults, J. S. (1987) Multiple spans in transcription typing. *Journal of Applied Psychology 72*, 2 (1987), 187-196.

Schmidt, R. A. and Bjork, R. A. (1992) New conceptualizations of practice: common principles in three paradigms suggest new concepts for training. *Psychological Science 3*, 4, 207-217.

Schomaker, L. R. B. (1994) User-interface aspects in recognizing connected-cursive handwriting. In *Proceedings of the IEE Colloquium on Handwriting & Pen-based Input 1994/065*. The institution of Electrical Engineers, London, UK, 8/1-8/3.

Sears, A. (1991) Improving touchscreen keyboards: design issues and a comparison with other devices. *Interacting with Computers 3*, 3, 253-269.

Sears, A. and Zha, Y. (2003) Data entry for mobile devices using soft keyboards: understanding the effects of keyboard size and user tasks. *International Journal of Human-Computer Interaction 16*, 2, 163-184.

Sears, A., Revis, D., Swatski, J., Crittenden, R., and Shneiderman, B. (1993) Investigating touchscreen typing: the effect of keyboard size on typing speed. *Behaviour & Information Technology 12*, 1, 17-22.

Sears, A., Revis, D., Swatski, J., Crittenden, R., and Shneiderman, B. (1993) Investigated touchscreen typing: the effect of keyboard size on typing speed. *Behaviour & Information Technology 12*, 1, 17-22.

Sezgin, T. M. and Davis, R. (2005) HMM-based efficient sketch recognition. In *Proceedings of the 10th International Conference on Intelligent User Interfaces* (IUI '05). ACM, New York, NY, USA, 281-283.

Shaffer L. H. and Hardwick, J. (I969[a]). Errors and error detection in typing. *Quarterly Journal of Experimental Psychology 21*, 3, 209-213.

Shaffer L. H. and Hardwick, J. (I969[b]). Reading and typing. *Quarterly Journal of Experimental Psychology 21*, 4, 381-383.

Shaffer, L. H. (1973). Latency mechanisms in transcription. In *Attention and performance IV*, S. Komblum (Ed.). Academic Press, New York, NY, USA, 435-446.

Shaffer, L. H. (1975[a]) Control processes in typing. *Quarterly Journal of Experimental Psychology 27*, 3, 419-432.

Shaffer, L. H. (1975[b]) Multiple attention in continuous verbal tasks. In *Attention and performance, V*, P. M. A. Rabbin and S. Domic (Eds.). Academic Press, New York, NY, USA, 157-167.

Shaffer, L. H. (1978) Timing in the motor programming of typing. *Quarterly Journal of Experimental Psychology 30*, 2, 333-345.

Shaffer, L. H. and French, A. (1971). Coding factors in transcription. *Quarterly Journal of Experimental Psychology 23*, 3, 268-274.

Shaffer, L. H. and Hardwick, J. (1968). Typing performance as a function of text. *Quarterly Journal of Experimental Psychology 20*, 4, 360-369.

Shaffer, L. H. and Hardwick, J. (1970) The basis of transcription skill. *Journal of Experimental Psychology 84*, 3, 424-440.

Shilman, M., Tan, D. S., and Simard, P. (2006) CueTIP: a mixed-initiative interface for correcting handwriting errors. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software & Technology* (UIST '06). ACM, New York, NY, USA, 323-332.

Shulansky, J. D. and Herrmann, D. J. (1977). The influence of linguistic structure on typing. *Language and Speech 20*, 1, 80-85

Silfverberg, M. (2007) Historical overview of consumer text entry technologies. In *Text Entry Systems: Mobility, Accessibility, Universality*, I. S. MacKenzie and K. Tanaka-Ishii (Eds.). Morgan Kaufmann, San Francisco, CA, USA, 3-25.

Sirisena, A. (2002) Mobile text entry. Department of Computer Science, University of Canterbury, Christchurch, New Zealand.

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

Soukoreff, R. W. and MacKenzie, I. S. (2001) Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '01). ACM, New York, NY, USA, 319-320.

Soukoreff, R. W. and MacKenzie, I. S. (2003) Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, USA, 113-120.

Srinivasan, M. and Chen, J. (1993) Human performance in controlling normal forces of contact with rigid objects. *ASME Winter Annual Meeting 49*, 119-125.

Sternberg, S., Knoll, R. L., and Wright, C. E. (1978) Experiments on temporal aspects of keyboard entry. In *Gelling It Together: Research and Applications in Human Factors*, J. P. Duncanson (Ed.). Human Factors Society, Santa Monica, CA, USA, 28-50.

Strong, E. P. (1956) A comparative experiment in simplified keyboard retraining and standard keyboard supplementary training. US General Services Administration, Washington, DC, USA.

Suhm, B. (1997) Empirical evaluation of interactive multimodal error correction. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding* (ASRU '97). IEEE, Washington, DC, USA, 583, 590.

Tanaka-Ishii, K., Hayakawa, D., and Takeichi, M. (2003) Acquiring vocabulary for predictive text entry through dynamic reuse of a small user corpus. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics – Volume 1* (ACL '03). Association for Computational Linguistics, Morristown, NJ, USA, 407-414.

Tang, H., Beebe, D. J., and Kramer, A. F. (2001) A Multilevel input system with force-sensitive elements. *International Journal of Human-Computer Studies 54*, 4, 495-507.

Tapp, K. M. and Logan, G. D. (2011). Attention to the hands disrupts skilled typewriting: the role of vision in producing the disruption. *Attention, Perception & Psychophysics 73*, 8, 2379-2383.

Tappert, C. C. and Cha, S. (2007) English language handwriting recognition interfaces. In *Text Entry Systems: Mobility, Accessibility, Universality*, I. S. MacKenzie and K. Tanaka-Ishii (Eds.). Morgan Kaufmann, San Francisco, CA, USA, 123-137.

Tappert, C. C., Suen, C. Y., and Wakahara, T. (1990) The state of the art in online handwriting recognition. *Pattern Analysis & Machine Intelligence 12*, 8, 787-808.

Terzuolo, C. A. and Viviani, P. (1980) Determinants and characteristics of motor patterns used for typing. *Neuroscience 5*, 6, 1085-1103.

Thomas, E. A. C. and Jones, R. G. (1970) A model for subjective grouping in typewriting. *Quarterly Journal of Experimental Psychology 22*, 3, 353-367.

Tomasi, C., Rafii, A., and Torunoglu, I. (2003) Full-size projection keyboard for handheld devices. *Communications of the ACM 46*, 7, 70-75.

Trnka, K., McCaw, J., Yarrington, D., McCoy, K. F., and Pennington, C. (2009) User interaction with word prediction: the effects of prediction quality. *ACM Transactions on Computer-Human Interaction 1*, 3, Article 17, 34 pages.

Tsukada, K. and Yasumura, M. (2002) Ubi-finger: gesture input device for mobile use. *Transactions of Information Processing Society of Japan 43*, 12, 3675-3684.

Tu, H., Ren, X., and Zhai, S. (2012) A comparative evaluation of finger and pen stroke gestures. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 1287-1296.

Vatavu, R., Anthony, L., and Wobbrock, J. O. (2012) Gestures as point clouds: a $P recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction* (ICMI '12). ACM, New York, NY, USA, 273-280.

Venolia, D. and Neiberg, F. (1994) T-Cube: a fast, self-disclosing pen-based alphabet. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '94), Beth Adelson, Susan Dumais, and Judith Olson (Eds.). ACM, New York, NY, USA, 265-270.

Wang, F., Cao, X., Ren, X., and Irani, P. (2009) Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09). ACM, New York, NY, USA, 23-32.

Watanabe, Y., Makino, Y., Sato, K., and Maeno, T. (2012) Contact force and finger angles estimation for touch panel by detecting transmitted light on fingernail. In *Proceedings of the 2012 international conference on Haptics: perception, devices, mobility, and communication- Volume Part I* (EuroHaptics'12), P. Isokoski and J. Springare (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 601-612.

West, L. J. (1967). Vision and kinesthesis in the acquisition of typewriting skill. *Journal of Applied Psychology 51*, 2, 161-166.

West, L. J. and Sabban, Y. (1982) Hierarchy of stroking habits at the typewriter. *Journal of Applied Psychology 67*, 3, 370-376.

White, W. T. (1932) *Typing for accuracy*. H. M. Rowe Company, Baltimore and Chicago, USA.

Wigdor, D. and Balakrishnan, R. (2003) TiltText: using tilt for text input to mobile phones. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software & Technology* (UIST '03). ACM, New York, NY, USA, 81-90.

Wigdor, D. and Balakrishnan, R. (2004) A comparison of consecutive and concurrent input text entry techniques for mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 81-88.

Williams, K. E. (1993) Automating the cognitive task modeling process: An extension to GOMS for HCI. In *Proceedings of the Fifth International Conference on Human-Computer Interaction Poster Sessions: Abridged Proceedings* (IHCI '93), 3, 182.

Wilson, A. and Shafer, S. (2003) XWand: UI for intelligent spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, USA, 545-552.

Wobbrock, J. O. (2006) A robust design for accessible text entry. *SIGACCESS Accessible Computing 84*, 48-51.

Wobbrock, J. O. (2007) Measures of text entry performance. In *Text Entry Systems: Mobility, Accessibility, Universality*, I. S. MacKenzie and K. Tanaka-Ishii (Eds.). Morgan Kaufmann, San Francisco, CA, USA, 47-74.

Wobbrock, J. O. and Myers, B. A. (2005) Gestural text entry on multiple devices. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility* (Assets '05). ACM, New York, NY, USA, 184-185.

Wobbrock, J. O. and Myers, B. A. (2006) Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction 13*, 4, 458-489.

Wobbrock, J. O., Myers, B. A., and Kembel, J. A. (2003) EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software & Technology* (UIST '03) ACM, New York, NY, USA, 61-70.

Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007) Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software & Technology* (UIST '07). ACM, New York, NY, USA, 159-168.

Wu, C. and Liu, Y. (2004[a]) Modeling behavioral and brain imaging phenomena in transcription typing with queuing networks and reinforcement learning algorithms. In *Proceedings of the International Conference on Cognitive Modeling* (ICCM '04). Lawrence Erlbaum, Mahwah, NJ, USA.

Wu, C. and Liu, Y. (2004[b]) Modeling human transcription typing with queuing network-model human processor (QN-MHP). In *Proceedings of the 48th Annual Meeting of Human Factors and Ergonomics Society* (HFES '04). Human Factors and Ergonomics Society, New Orleans, LA, USA, 381-385.

Yamada, H. (1980) A historical study of typewriters and typing methods: From the position of planning Japanese parallels. *The Journal of Information Processing 2*, 4, 175-202.

Zapf, D., Brodbeck, F. C., Frese, M., Peters, H. and Prumper, J. (1992) Errors in working with office computers. A first validation of a taxonomy for observed errors in a field setting. *International Journal of Human-Computer Interaction 4*, 4 (1992), 311-339.

Zeleznik, R., Miller, T., and Forsberg, A. (2001) Pop through mouse button interactions. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (UIST '01). ACM, New York, NY, USA, 195-196.

Zhai, S. and Kristensson, P. (2003) Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, USA, 97-104.

Zhai, S. and Kristensson, P. (2012) The word-gesture keyboard: reimagining keyboard interaction. *Communications of the ACM 55*, 9, 91-101.

Disclaimer

All the materials used in this document; i.e., images, tables, etc., are either self-generated or in public domain, unless stated otherwise.

Portions of this work have been published in the following international conferences.

- Arif, A. S. and Stuerzlinger, W. (2009) Analysis of text entry performance metrics. In *Proceedings of the IEEE Toronto International Conference—Science and Technology for Humanity* (TIC-STH '09). IEEE, Washington, DC, USA, 100-105.
- Arif, A. S. and Stuerzlinger, W. (2010) Predicting the cost of error correction in character-based text entry technologies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 5-14.
- Arif, A. S. and Stuerzlinger, W. (2014) User adaptation to a faulty unistroke-based text entry technique by switching to an alternative gesture set. In *Proceedings of Graphics Interface 2014* (GI '14). Canadian Information Processing Society, Toronto, Ontario, Canada, 183-192.
- Arif, A. S. and Stuerzlinger, W. (2013) Pseudo-pressure detection and its use in predictive text entry on touchscreens. In *Proceedings of the 25th Australian Computer-Human Interaction Conference* (OzCHI '13). ACM, New York, NY, USA, 383-392. ***Best paper award.***

Several additional articled have also been published, which are indirectly related to this work. Some of these articles are cited in this document, as appropriate.

Appendices

The appendices are sorted alphabetically.

A1. Effect Size (η^2)

Eta-squared (η^2) is a measure of effect size (Cohen, 1988). This measure is intended for ANOVAs, instead of Cohen's *d*, which was designed for t-tests. η^2 describes the ratio of variance explained in the dependent variable by an independent variable while controlling for other independent variables. It is a biased estimator of the variance explained by the model in the population. As the sample size (*N*) gets larger the amount of bias decreases. Stated simply, it tells us how much an independent variable has affected the dependent variable in an empirical study. On average η^2 overestimates the variance explained in the population. It ranges between 0 and 1. Cohen (1988) offered conservative threshold criteria for η^2, where η^2 = 0.0099 constitutes a small, η^2 = 0.0588 a medium, and η^2 = 0.1379 a large effect[15].

The following equation was used in this dissertation to calculate η^2.

$$\eta^2 = \frac{SS_{Treatment}}{SS_{Total}} \qquad \text{Equation (20)}$$

Here, $SS_{Treatment}$ is the sum of squares for the effect of interest, and SS_{Total} is the total sum of squares for all effects, interactions, and errors in the ANOVA.

A2. Phrase Set

Almost all text entry user studies present participants with preselected short phrases of text, which are retrieved randomly from a set and are presented to participants one at a time to enter (see Section 2.1).

[15] Note that the threshold values to distinguish between small, medium, and large effects are different for other effect size measures such as Cohen's *d*, *f*, f^2, Glass's Δ, and Hedges's *g*. The following Wikipedia page provides more information and the corresponding threshold values for these measures: http://en.wikipedia.org/wiki/Effect_size.

During the studies reported in this document, participants entered short English phrases from MacKenzie and Soukoreff's (2003) phrase set. The phases used in the set are moderate in length (28.61 characters on average), easy to remember, and representative of the English language. The phrases do not contain any numeric and special characters. MacKenzie and Soukoreff (2003) argued that it is best to exclude these characters from the interaction, as they do not assist to differentiate the. A few phrases contained uppercase characters, which were converted to lowercases for the same reason – as with the investigated techniques these characters are inputted using the same keys. This corpus's high correlation with the character frequency in the English language also encouraged researchers to use it in almost all recent text entry studies (see Section 2.4). The corpus is available online[16].

A3. Sample Size (N)

It is possible to calculate the power of statistical tests prior to a study (a priori) to determine the sample size (*N*), that is, the number of participants required. However, a priori power analysis is rarely done in human-computer interaction research, since it requires knowing the variance in a sample and the difference in the means on the dependent variable (effect size) before the data are collected (MacKenzie, 2013). Thus, the recommended procedure is to study the existing literature (MacKenzie, 2013). If a similar study reports statistically significant results with a particular number of participants, then using that many participants is a reasonable choice. The user studies reported in this document follow this recommendation. Section 2.4.3, particularly Table 1, presented results from existing text entry studies, similar to the ones reported here, along with their sample size.

A4. Statistical Power (1–β)

This work used post-hoc power analysis, as motivated in Appendix A3.

Cohen (1992) defined: "*The power of a statistical test of a null hypothesis (H_0) is the probability that the H_0 will be rejected when it is false, that is, the probability of obtaining a statistically significant result*".

[16] http://www.yorku.ca/mack/PhraseSets.zip

Statistical power analysis exploits the mathematical relationship between the four variables in statistical inference: power (*1–β*), false positive rate (*α*), sample size (*N*), and effect size (*f*). The relationship permits one to determine the value of one variable when the other variable values are known. Based on this, post-hoc power analysis detects a hypothesized *1–β* for specified *α*, *N*, and *f*. The false positive rate *α* is also referred to as the probability of a Type I error, while *β* is referred to as false negative rate or the probability of a Type II error.

Cohen (1992) suggested the use of a threshold of .80, that is, *β* = .20, for a level of desired power when no other basis for setting the value is available. The reason behind this is that it is more misleading to make a false positive claim (larger α) than a false negative claim (larger *β*). As the convention for the significance level in HCI is to use a .05 threshold, the use of .80 for desired power (*β* = .20) makes *β* four times more likely than α, which is a reasonable reflection of their relative importance (Cohen, 1992).

This work calculated the power of a statistical test using the G*Power software package (Cunningham and McCrum-Gardner, 2007). For this purpose, the correlation among the repeated measures, α, N, and *f* were calculated individually for each test and input into the package to obtain the statistical power, *1-β*. Cohen's *f* was calculated from η^2, see Appendix A1 above, using the following equation (Faul et al., 2007).

$$f = \sqrt{\frac{\eta^2}{1-\eta^2}} \qquad \text{Equation (21)}$$