Final accepted manuscript, paper to appear in *Interacting with Computers*, *Vol.16* (2004)

In Search of Effective Text Input Interfaces for Off the Desktop Computing

Shumin Zhai **Per-Ola Kristensson** **Barton A. Smith**

IBM Almaden Research Center[1], 650 Harry Road, San Jose, CA, USA

ABSTRACT

It is generally recognized that today's frontier of HCI research lies beyond the traditional desktop computers whose GUI interfaces were built on the foundation of display—pointing device—full keyboard. Many interface challenges arise without such a physical UI foundation. Text writing — ranging from entering URLs and search queries, filling forms, typing commands, to taking notes and writing emails and chat messages — is one of the hard problems awaiting solutions in off-desktop computing. This paper summarizes and synthesizes a research program on this topic at the IBM Almaden Research Center. It analyzes various dimensions that constitute a good text input interface; briefly reviews related literature; discusses the evaluation methodology issues of text input; presents the major ideas and results of two systems, ATOMIK and SHARK; and points out current and future directions in the area from our current vantage point.

Keywords

Text input; pervasive, mobile, off-desktop computing; shorthand; gesture; stylus, virtual keyboard.

INTRODUCTION

Desktop computers, as well as their more mobile cousins – laptop computers, take the form of "workstations" featuring a large and personal display, a pointing device, and a keyboard evolved from the typewriter. The display–pointing device–keyboard tripod forms the physical foundation of today's GUI interfaces.

---

[1]Alison Sue, Clemens Drews, Paul Lee, Johnny Accot, Michael Hunter and Jingtao Wang have also significantly contributed to the work reported in this overview paper and most of them have co-authored separate research papers cited in the text. Per-Ola Kristensson (Linkoping University, Sweden), Paul Lee (Stanford), Michael Hunter (BYU), and Jingtao Wang (UC Berkeley) were members of this project during their internships at the IBM Almaden Research Center in different periods of time.

Against this background, it is generally recognized that today's frontier of HCI research lies beyond the traditional desktop computer. Increasingly computing could be embedded in work and life environments, possibly with wall sized displays, or takes place in various mobile devices such as PDAs, Tablet PCs, or mobile phones. Off-desktop computing without the traditional display–pointing device–keyboard foundation faces many user interface challenges. Some will be surprisingly hard to solve. Text input is one of them. Text writing — including writing emails and chat messages, filling forms, typing commands, taking notes, authoring articles, and coding programs — constitutes one of the most frequent computer user tasks.

A good textual interface has many desired dimensions. First it should be efficient. An average user with sufficient amount of practice should be able to write text at a sufficiently fast speed without making many errors. Second, it should have a low initial usability threshold and a rapid learning curve. Third, it should impose a low cognitive, perceptual, and motor demand on the user. Fourth, it should be fun to use, although fun could well be a product of the first three dimensions. Fifth, it should be easy to access. Any requirement of attaching devices to the user, or even picking up and mounting a headset as in speech recognition, can be a significant impediment to frequent and constant use. For mobile computing, a compact form factor is obviously also important.

This paper first analyzes various dimensions that a comprehensive evaluation method should consider. It then briefly reviews the QWERTY typewriter keyboard and other recent textual interfaces related to our work. We then focus on synthesizing, updating, revising and summarizing the major ideas and results of our own research in this area. The paper ends with issues and directions we believe are critical to future research.

EVALUATION METHODS

Text typing involves rather complex and multi-faceted perceptual, cognitive and motor processes (Cooper, 1983). Because of this complexity, there is no one standard or best evaluation method in the field of text input research. Even in traditional typewriting research, a clear cut conclusion has rarely been reached on issues such as the relative superiority of the QWERTY vs. Dvorak layout (Lewis, Potosnak, & Magyar, 1997; Yamada, 1980). Developing a standard set of tasks for evaluating text input methods will be an important contribution to the field. Such a battery of procedures or tasks should consider at least the following aspects and issues of text input.

1. *Ultimate performance*. The eventual performance for an average user is often the most important goal in designing a new input method. The effective speed is usually the first question that comes to the user's mind when facing a new method. Unfortunately it is also a very difficult question to answer, because the user's speed is a function of practice and many other variables. Without a substantial amount of training, what is measured is not going to be close to the ultimate performance. Substantial amount of training is always costly, especially if more than a few participants are employed. It has been reported that it takes more than 100 hours of training on Dvorak keyboard for typists to catch up with their old QWERTY performance (Cooper, 1983). One solution for a standard is to set particular "milestone" speed performance measurements at, for example, 15 minutes, 1 hour, 5 hours, 10 hours, 20 hours, 50 hours, and 100 hours of practice.

2. *Error and Error handling*. Another important dimension to consider is error rate and ease of error correction. What matters is the effective speed, or speed after correcting any errors made. Experimentally there are different ways of handling errors in studies; each has different implications to the text input study result. Some leave errors in the text and report them separately (MacKenzie & Zhang, 1999). Others do not allow errors, e.g. the testing program will not proceed until the correct character is entered (Zhai, Sue, & Accot, 2002). Yet others require the participants to correct their errors, and measure their effective speed including errors. The amount of time needed to correct errors depends on the design of the error correction mechanism; hence there are no set rules to tradeoff errors with speed. Traditionally in typing contest each error caused the deduction of 5 words in the participant's score. Making study participants correct their errors seems most informative overall, although it lumps input and correction together.

3. *Models as performance measures*. An alternative approach to empirically measuring expert performance is theoretical modeling. Computer simulation based on theoretical modeling has been used in typing research (Rumelhart & Norman, 1982). Theoretical models, if they indeed capture the essence of user-system interaction, offer a clear description of factors that influence performance. For example, the Fitts-digraph model of stylus keyboarding (Lewis, 1992; Soukoreff & MacKenzie, 1995; Zhai, Sue, & Accot, 2002), to be reviewed later in this paper, gives us a baseline prediction of the average user performance when a stylus keyboarding method is well practiced. Note that a model is always built with strong assumptions. In the case of a stylus keyboard, it assumes that learning would reduce the visual search time to be negligible in comparison to the stylus movement time. With most new techniques, we do not know enough about human perceptual motor and cognitive skills to form strong quantitative laws as a foundation to theoretical performance prediction. Understanding atomic human performance regularities is undoubtedly important to future research and design (Accot & Zhai, 1997; Zhai, Accot, & Woltjer, 2004).

4. *Initial performance vs. longitudinal learning*. The very initial performance of a novice user and the initial slope of the learning curve often determine the fate of a new interaction technique. The general user population is very reluctant to invest a large number of hours in learning before benefiting from a new interface. To measure the initial performance is relatively easy since it does not involve much practice on the participant's part. Conducting a longitudinal study to measure the user's learning curve is costly, particularly if the study employs more than a few participants. It is possible to use the power law of learning (regression) to extract future performance (Crossman, 1959). By the nature of regression analysis, however, a reliable estimate of the exponent of the learning curve still requires a large number of sessions. See (MacKenzie & Zhang, 1999) and (Zhai, Sue, & Accot, 2002) for two examples.

5. *Visual and cognitive attention*. Ideally a text input method requires little visual or cognitive attention so the user can focus on his or her writing or other real-world tasks. Most of the new input methods require more visual attention than the physical typewriter which could be "touch-typed". When a typing method does require visual attention, how should it tradeoff with typing speed? How should it be measured? Eye-tracking and secondary tasks are two possible measurement methods, but they would certainly make the already complex typing evaluation more cumbersome.

6. *Delight*. An interaction method should not only be efficient, but also fun to use. Although the interest in studying fun in interaction has emerged in the HCI field, it is poorly understood how fun is related to the other design dimensions of an interface. In the case of text entry, is it the fluidity of the method, ease of use, ease of learning, or skill challenge as in video games that make the interface fun? How could delight be reliably measured other than by self-reporting?

7. *Task Validity*. "Copy typing", the targeted task in traditional typing research, is rarely practiced today. Most of the new text input interfaces for off-desktop computing are aimed at generative typing – typing from the user's mind. From this perspective, a writing task based on some task scenarios (e.g. writing an email to schedule an appointment) is a more valid evaluation task than typing prescribed text. However, such a task could also be much less sensitive as an evaluation tool, because the task components unrelated to the efficacy of the textual interface, such as problem solving or composition could take a large portion of, or even dominate the entire evaluation task. To reduce visual attention needed in "copy typing" like tasks, familiar, memorable text can be used (MacKenzie & Zhang, 1999). However, this brings up the issue of linguistic fidelity.

8. *Linguistic fidelity*. Depending on the purpose of the evaluation, one has to decide if the text used for testing should be linguistically representative in frequency distributions (letter, word, digraph etc). Another issue to consider is if the text is open (unlimited, fresh text each time) or closed (same sentence or paragraph repeated). This is because the skill learned during the study could be specific to the words practiced and may not generalize to open text.

It is difficult, if not impossible; to incorporate all of the above dimensions in evaluating a new text input method. Depending on the purpose of the evaluation, a study may focus on and control a sub-set of these

dimensions. Results of an evaluation should be understood and interpreted with the conditions and assumptions in mind. Comparing different measurements made under different sets of conditions are always difficult and often misleading.

CURRENT APPROCHES

The need for entering text off the desktop has driven numerous inventions in text entry in recent years, although the majority has not been properly researched with either theoretical or empirical human performance studies. In this short review we focus on only a few major approaches related to our work. More comprehensive reviews of novel text input methods are provided elsewhere (Buxton, Billinghurst , Guiard, Sellen, & Zhai, 1994/2002; MacKenzie & Soukoreff, 2002; Zhai, Smith, & Hunter, 2002).

The QWERTY Typewriter Keyboard

It is informative to briefly review the typewriter keyboard although it is not an off-desktop method. The most common typewriter uses the QWERTY keyboard, credited to Christopher L. Sholes, Carlos Glidden and Samuel Soule in 1867 (Yamada, 1980). Systematic study of human performance in typewriting and optimizing the keyboard layout happened much later, with the best known example being the Dvorak study and layout (Dvorak, Merrick, Dealey, & Ford, 1936). Human factors research on typewriting continues in the modern literature (e.g. (Cooper, 1983)). The QWERTY layout as the de facto standard is a subject of debate. Research comparing QWERTY with alternative layouts such as the Dvorak simplified keyboard has shown varied, sometimes opposite results. (Cooper, 1983; Lewis, Potosnak, & Magyar, 1997; Norman & Fisher, 1982; Yamada, 1980) provide a few reviews of this literature. The QWERTY vs. Dvorak debate, interestingly, is just as heated in the business and economics literature where the proponents of "path dependence" or "lock-in" theory cite QWERTY as a classic example of "QWERTYNomics". In QWERTYNomics, an accidental sequence of events may lock technology development into a particular irreversible path - "one damn thing follows another" (David, 1985). *Technical interrelatedness*, *economics of scale*, and *quasi-irreversibility* prevent better alternative rationally optimized solutions from prevailing (David, 1985). The opponents of QWERTYNomics argue that QWERTY has not been replaced because there is no convincingly superior alternative to QWERTY (Liebowitz & Margolis, 1996; Liebowitz & Margolis, 1990).

The main rationale of the QWERTY arrangement was to minimize mechanical jamming (Cooper, 1983; Yamada, 1980). Accidentally, this design also facilitates the frequent alternation of the left and right hand, which is a key factor in rapid touch-typing with two hands. Partially because the QWERTY design scores well in alternation frequency, various attempts to replace QWERTY with more efficient layouts, such as the Dvorak simplified keyboard (Dvorak, Merrick, Dealey, & Ford, 1936), have not prevailed. The performance gain with these newer designs (estimated around 5 to 15%) has not been substantial enough to justify the cost of retraining the great number of QWERTY users (Cooper, 1983; Lewis, 1992; Lewis, Potosnak, & Magyar, 1997; Norman & Fisher, 1982; Yamada, 1980).

Montgomery (Montgomery, 1982) made another attempt to rearrange the key layout so that some common short words or word fragments can be wiped through. This attempt pointed to the direction of the SHARK method to be reviewed later, although it did not involve the key ideas and methods in SHARK such as pattern recognition and shorthand production for all words needed.

The typewriter keyboard offers numerous advantages as a human-computer interface. In addition to speed, a touch typist can focus his or her visual attention on the computer screen, not the typewriter keyboard itself. Interestingly, touch-typing (typing without looking at the keyboard) was first applied in the 1880s by L.V. Longley and F. E. McGurrin, many years after the typewriter invention, and was not widely adopted by training schools until about 1915 (Yamada, 1980). This means the low attention demand was not a rational design feature, but rather an evolutionary improvement discovered in the process of use.

Although the typewriter keyboard has been a resilient de facto standard method for text input, it has many weaknesses as a modern interface technology. First it takes effort to learn touch-typing skills; and usually takes hundreds of hours of practice to be proficient (Cooper, 1983). Second, the argument of over

specification, to be discussed later on stylus keyboards, can also be made here. The system is unable to take advantage of today's computing power in applying statistical information to reduce the load of input. Third and most importantly, it is not suited for off-desktop computing in its usual size and form.

Speech Recognition

Speech recognition has been expected to be a compelling alternative to manual typing. Despite the progress made in speech recognition technology, however, a recent study (Karat, Halverson, Horn, & Karat, 1999) showed that the effective speed of text entry by continuous speech recognition was still far lower than that of the keyboard (13.6 vs. 32.5 corrected wpm for transcription and 7.8 vs. 19.0 corrected wpm for composition). Error correction is particularly difficult with speech commands. Furthermore, the study also revealed many human-factors issues that had not been well understood. For example, many users found it "harder to talk and think than type and think" and considered the keyboard to be more "natural" than speech for text entry. It has been further argued that speech production competes for cognitive/memory resources which impede the user's performance (Shneiderman, 2000). In short, using speech as a text input method still faces many challenges.

Handwriting recognition

Handwriting is a rather "natural" and fluid mode of text entry, thanks to users' prior experience from writing on paper. Handwriting recognition technology has made tremendous progress in recent years (Plamondon & Srihari, 2000). The current PDA devices tend to use alphabet character based handwriting recognition, such as *Graffiti* and *Jot*. The alphabet used can be either natural or artificially modified for reliable recognition (Goldberg & Richardson, 1993). EdgeWrite defines an alphabet around the edge of a fixture to help users with motor impairment (Wobbrock, Myers, & Kembel, 2003). The fundamental weakness of (long) handwriting, however, is the limited speed, typically estimated around 15 wpm (Card, Moran, & Newell, 1983). For Graffiti and Jot, (Sears & Arora, 2002) found between 4.3 -7.7 wpm performance for new users and 14-18 wpm for more advanced users, although other informal reports claimed higher peak performance. This speed might be good enough for entering names and phone numbers on a PDA, but too slow for writing a longer text.

Mobile physical keyboard

There are various ways to reduce the size of physical keyboards. One is to shrink the size of each key. This is commonly seen in electronic dictionaries. Typing on these keyboards is difficult due to their reduced size that prevents ten finger touch typing. Another method is to use the number pads in telephones, whereby each number corresponds to multiple letters. The ambiguity of multiple possible letters is commonly resolved by the number of consecutive taps, or by lexical models. Optical projection keyboards is yet another approach, although a key issue is the lack of tactile feedback of the keys, both vertically (the non-linear resistance of a key) and laterally (key surface features that prevent the finger drifting away). Furthermore it requires set-up and a 'desktop' space to operate.

Gesture input

There have been various *continuous-gesture*-based text methods. *Quikwriting* (Perlin, 1998) uses continuous stylus movement on a radial layout to enter letters. *Cirrin* (*Cir*cul*ar In*put) is another example) (Mankoff & Abowd, 1998). Dasher (Ward, Blackwell, & MacKay, 2000) uses continuous mouse movement to pass through traces of letters laid out by a predictive language model. Using such a technique is a novel and intriguing experience, but the primary drawback is that the user has to continuously recognize the dynamically rearranged letters. The visual recognition task may limit the eventual performance of text entry with such a method.

Stylus keyboards

Stylus keyboards display letters and numbers on a touch sensitive screen or surface. To input text, the user presses keys with a finger or stylus. Such a keyboard can be scaled to fit computing devices with varying sizes, particularly small handheld devices. One central issue is the layout of the keys in these keyboards. Due to developers' and users' existing knowledge, QWERTY tend to be also the default layout of stylus keyboards. However, QWERTY is a poor choice for stylus keyboarding. The polarizing

positions of common English digraphs in QWERTY mean that the stylus has to move back and forth more frequently and over greater distances than necessary.

The key to a good virtual keyboard is exactly opposite to the idea behind QWERTY. Common digraph letters should be close to each other so the hand does not have to travel much. The movement distance concern also points to another problem of QWERTY as a virtual keyboard layout, it is elongated horizontally, which increases the average stylus movement distances[2]. The human performance effect of relative distances between the letters can be modelled by a simple movement equation – Fitts' law (Fitts, 1954), which may serve as a foundation to optimize the letter positions in stylus keyboards.

OPTIMIZATION OF STYLUS KEYBOARD

The Fitts-digraph model of virtual keyboards

Lewis (Lewis, 1992; Lewis, Potosnak, & Magyar, 1997) was probably the first to use the well known Fitts' law and digraph frequency as bases to model stylus keyboard performance, followed by MacKenzie and colleagues (MacKenzie & Zhang, 1999; Soukoreff & MacKenzie, 1995). According to Fitts' law, the mean time to move the tapping stylus from one key i to another j for a given distance (D_{ij}) and key size (W_j) is

$$t_{ij} = a + b \log_2 (\frac{D_{ij} + W_j}{W_j}) \qquad (1)$$

where *a* and *b* are empirically determined coefficients.

The validity of Fitts' law for general tapping tasks has been proven repeatedly by a vast literature. See (MacKenzie, 1992) for a general review, and (Accot & Zhai, 2002; Accot & Zhai, 2003; McGuffin & Balakrishnan, 2002; Zhai, Conversy, Beaudouin-Lafon, & Guiard, 2003) for recent developments in the context of HCI. In the case of stylus keyboarding, by analyzing the Gaussian hits distribution of the stylus tip, we have shown that the users do effectively use the virtual keys as Fitts' law assumes – they could control their normally distributed hit points "tightly" enough to be within the key size, but also sufficiently "loosely" to utilize the given surface of the keys (Zhai, Sue, & Accot, 2002).

There is a wide range of values of the Fitts' law parameters (*a* and *b*) reported in the literature. For example, *IP* (*=1/b*) has been reported as low as 4.9 bits per second (bps) in (MacKenzie, Sellen, & Buxton, 1991), and as high as 12 bps in (Fitts, 1954). As the Fitts' law parameters change, the movement speed limit on virtual keyboards change dramatically. Without empirically measured performance specifically for stylus keyboarding, researchers (MacKenzie & Zhang, 1999) tended to use the more conservative estimates of a = 0, b = 1 / 4.9 (sec) based on results from the more general Fitts' reciprocal tapping tasks (MacKenzie, Sellen, & Buxton, 1991).

More recently, (Zhai, Sue, & Accot, 2002) estimated the values of *a* and *b* in Fitts' law in the context of stylus typing which involves a relatively small range of index of difficulty formed by tightly packed targets, the use of stylus, and visual recognition of a series of intended target letters (but excluding visual search needed by the novice users of stylus keyboards) in varied directions. Their results were $a = 0.083$ sec, $b = 0.127$ sec, or $IP = 1/b = 7.9$ bps.

Once the basic Fitts' law parameters are estimated, the *average* tapping time (per key) on a stylus keyboard can be obtained by calculating the average of all pairs of keys:

[2] From a screen space management point of view, it is better to have an elongated rather than square shaped on screen keyboard. This can be achieved by arranging the numeric and auxiliary keys on the two sides of alphabetical key set rather than horizontally stretching the alphabetical key set.

$$\bar{t} = \sum_{i,j \in S} p_{ij} t_{ij} \qquad (2)$$

where *S* is the key set, usually including the 26 letters in English and the space key, and p_{ij} is the diagraph transition probability from letter *i* to letter *j* in a language corpus.

Combining Eq (1) and (2),

$$\bar{t} = a + b \sum_{i,j \in S} p_{ij} \log_2 \left(\frac{D_{ij} + W_j}{W_j} \right) \qquad (3)$$

Using the estimates $a = 0.083$ sec, $b = 0.127$ sec, we have:

$$\bar{t} = 0.083 + 0.127 \sum_{i,j \in S} p_{ij} \log_2 \left(\frac{D_{ij} + W_j}{W_j} \right) \qquad (4)$$

Assuming 5 characters per

on trial and error exploration is OPTI (MacKenzie & Zhang, 1999). Fitaly (TextwareSolutions, 1998) is a commercial example.

(Getschow, Rosen, & Goodenough-Trepagnier, 1986) used a simple greedy algorithm that placed alphabetical letters to most easily accessible positions according the letters frequency rank order. As the authors stated, this greedy algorithm ignores many arrangements that could be substantially better, because it does not consider the letter placement with respect to each other. The opposite approach to the simple greedy algorithm is exhaustive algorithmic searching that calculates the efficiency of each and every combination of letter arrangement. However, the complexity of that search - O(n!) - is approximately 10^{28}, a too large number even for modern computing.

The idea of minimizing energy, or tension, in the keyboard layout brought us to explore a well-known optimization method – the Metropolis algorithm. The Metropolis algorithm is a Monte Carlo method widely used in searching for the minimum energy state in statistical physics (Beichl & Sullivan, 2000; Binder & Heermann, 1988; Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953). If we define Equation (4) as "Fitts–Digraph energy", the problem of designing a high performance keyboard is equivalent to searching for the structure of a molecule (the keyboard) at a stable low energy state determined by the interactions among all the atoms (keys). Applying this approach, we designed and implemented a software system that did a "random walk" in the virtual keyboard design space. In each step of the walk, the algorithm picked a key and moved it in a random direction by a random amount to reach a new configuration. The level of Fitts' energy in the new configuration, based on Equation (2), was then evaluated. Whether the new configuration was kept as the starting position for the next iteration depended on the following Metropolis function:

$$W(O \to N) = e^{-\Delta E / kT} \quad if\ \Delta E > 0 \qquad (5)$$
$$= 1 \qquad if\ \Delta E \le 0$$

In Equation (5), $W(O \to N)$ was the probabilty of changing from configuration O (old) to configuration N (new), ΔE was the energy change, k was a coefficient, T was "temperature", which could be interactively adjusted. The use of Equation (5) occasionally allowed moves with positive energy change in order to be able to climb out of a local energy minimum. An interactive "annealing" process bringing temperature T through up and down cycles was repeated until no further improvement was seen. We call layouts produced by this process Metropolis keyboards. The details of this design process can be found in (Zhai, Hunter, & Smith, 2000).

Alternatively, instead of performing the random walk algorithm on an open space, we have also developed a program that used the Metropolis method in a confined array of hexagons. Each walk step was taken by swapping a random pair of keys. The rest of the random walk process was same as the previous approach. Our experience shows that this was an equally effective, and more efficient approach (Zhai, Smith, & Hunter, 2002).

By means of the Metropolis algorithms, we have designed a variety of layouts with different characteristics, all with similar movement efficiency. In other words, instead of one deep and narrow Fitts–diagraph energy canyon, there is a relatively flat valley in the stylus keyboard design landscape, all much better than non-optimized keyboard but not significantly different from each other in efficiency. This means that it is possible to accommodate additional design considerations without lowering movement efficiency. We have taken advantage of this fact in various ways (Zhai, Smith, & Hunter, 2002), most notably "alphabetical tuning".

For novice users of virtual keyboards, speed is determined mostly by the need to search and find target keys rather than by the amount of motor movement. A keyboard optimized by movement efficiency only may look rather arbitrary to a novice user and hence be difficult to search. We explored the possibility of easing the novice user's search process by introducing alphabetical ordering to a virtual keyboard layout. Alphabetical layout is not a new idea. For example, Norman and Fisher studied a *strictly* alphabetical layout of the *physical* keyboard of a typewriter (Norman & Fisher, 1982). They showed that even novice

users did not type faster on such a keyboard than on a standard QWERTY keyboard. The main problem with an alphabetical keyboard, they concluded, was that the keys were laid out sequentially in multiple rows. The location of a key depended on the length of each row – the break point from which the next letter had to start at the left end of the keyboard again. (MacKenzie, Zhang, & Soukoreff, 1999) studied a virtual keyboard where the letters were laid alphabetically in two columns. Again, they did not find a performance advantage with the alphabetical layout. (Lewis, LaLomia, & Kennedy, 1999) proposed a 5 by 6 virtual keyboard layout with a strictly alphabetical sequence (See Figure 5 and 6). Such a design should suffer from the same problem as discovered by Norman and Fisher – the alphabetical discontinuity caused by row breaks.

Instead of strictly laying out the keys in an alphabetical sequence, we introduce *alphabetical tuning* in the optimization process. To produce such a keyboard, an additional term was added to the "energy" function, which, for each key, depended on the place in the alphabet for the character, and on its position on the keyboard:

$$e = \bar{t} + \lambda \sum_{i=a}^{z} \eta(i)(y_i - x_i) \qquad (6)$$

where $\bar{t}$ was the previous energy term defined by Equation (3). λ was an empirically adjusted weighting coefficient, depending on how much alphabetical order was brought to consideration at the cost of the average movement time. $\eta(i)$ was an integer number representing the place of the letter in the alphabet, with $\eta(a) = -12$, $\eta(b) = -11$, $....\eta(m) = 0$, $\eta(n) = 1$, … and $\eta(z) = 13$. x_i and y_i were the coordinates of letter i with origin (0,0) at the center of the keyboard. The term $\eta(i)(y_i - x_i)$ could be viewed as two forces. $\eta(i)y_i$ produced a force pushing the first half of the letters(a to m) upward, the second half (n to z) downwards, with a resulting energy proportional to letter distance from the center. For example, for letter a, $\eta(a) = -12$. The lowest energy state for it is the uppermost position and the highest energy state lies in the lowermost position (negative y_i). The opposite is true for letter z. ($\eta(z) = 13$). Similarly, the other force, $\eta(i)(-x_i)$ pushes the first half the letters (a to m) leftwards and later half rightwards. For the space key, a special case in this treatment, the alphabetic bias term was zero at the center of the keyboard, and increases with distance from the center.

The result of Equation (6) as an objective function was the general *trend* of letters starting out from the upper left corner moving towards the lower right corner. (Zhai, Smith, & Hunter, 2002) shows that it is possible to preserve the general tendency of alphabetical order without a significant sacrifice of movement efficiency.

In addition to alphabetical tuning, the connectivity of a word – the degree to which consecutive letters in the word are spatially adjacent, can also be important to movement efficiency, visual search and memory of the pattern of the word. This is particular important to the most common words such as *the, to, and, of, is, in,* and *it.*

We therefore introduced the third criterion in our design – Connectivity Index (CI). CI was defined as

$$CI = \sum_{i=1}^{N} f(i)c(i) \qquad (7)$$

where $f(i)$ is the percentage frequency of the i th most frequent word and $c(i)$ is the connectivity score of that word. For example, for the word “the”, if t-h and h-e are connected (adjacent), the word "the" gets

a score of 1: $c(i)$ = 1. It is multiplied by $f(i)$ = 3.38%, its frequency, before being added to *CI*. If only t-h or only h-e are connected, the word "the" gets a score of $c(i)$ = 0.5.

We call layouts satisfying all *three* critieria – movement efficiency, alphabetical tuning, and word connectivity ATOMIK (alphabetically tuned and optimized mobile interface keyboard) layouts, to reflect these criteria as well as the method by which they were produced – atomic interactions among all keys.

Figure 1 shows an ATOMIK layout. See (Zhai, Smith, & Hunter, 2002) for additional design and implementation details and considerations.

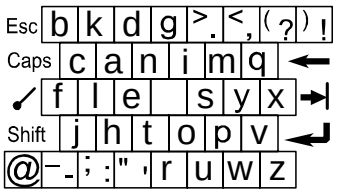


Fig. 1 The *ATOMIK* Keyboard (adapted from (Zhai, Smith, & Hunter, 2002) by permission)

SHARK SHORTHAND SYSTEM

Stylus typing suffers several drawbacks which, together with the use of inefficient QWERTY layout as a stylus keyboard, might have contributed to the relatively low rate of adoption. The simple tapping movement may feel tedious for prolonged use. Accurate tapping is also extremely demanding in visual attention. As a tool for complex and dexterous tasks such as drawing and writing, a stylus is much more expressive and fluid than a mouse. Not taking advantage of the pen's affordances is another weakness of stylus keyboard. In synergy with drawing, sketching and other pen experience in PDAs, Tablet PCs etc, a text entry method should maximally utilize the fluidity and dexterity of the pen.

Another important weakness of stylus keyboarding is the lack of support from the computer. It is well known that in any language there is a great deal of redundancy, as Shannon (Shannon, 1948) observed in the process of building his mathematical theory of communication. In this sense, tapping exactly on each individual letter in a word is an unnecessary over specification. It should be possible to reduce the amount of input information from the user and supplement it with information based on statistical and linguistic (lexical and grammatical) knowledge from the computer. The use of statistical information is common and essential in today's speech and handwriting recognition systems. The commonly available T9 method that produces the most probable word based on an ambiguous numeric key stroke sequence in mobile phones is another example. Bear in mind, however, when an interface is not entirely verbatim, it may impose more attention and cognitive burden on the user. It is an important research and design challenge to exploit machine intelligence while maintaining low attention and cognitive cost to the user.

Yet another resource that is not fully exploited with stylus tapping is the fact that people in general are very good at perceiving, recognizing, recalling and producing *patterns*.

These observations led us to design a system that allows the user to directly draw gesture patterns as a basic mode of entering words on a stylus keyboard. Each pattern of a word is formed by the trajectory through all of the letters of a word, from the first to the last in order, on the keyboard. Fig. 2 shows a few examples of such patterns defined by the *ATOMIK* keyboard in Fig. 1. Since these gesture patterns are essentially a form a shorthand for the words they represent, we call each of these patterns a *sokgraph* (*s*horthand *o*n *k*eyboard *graph)*, and the input system that uses sokgraph *SHARK* (shorthand aided rapid keyboarding).

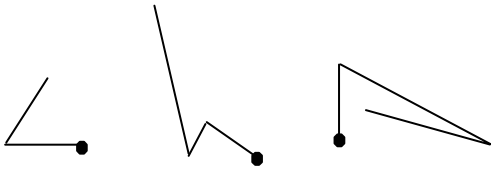


Fig. 2 Sokgraphs defined by the ATOMIK Keyboard for *the*, *word*, and *have* (a dot indicates the starting end)

There are many design rationales for such a SHARK system. We briefly review a few of them here. See (Zhai & Kristensson, 2003) for the original and more detailed discussion, but note that some of our thinking has further developed since the original SHARK paper.

Scale, location and visual attention relaxation: Sokgraphs to a certain degree can be scale and location independent when the user produces them on the keyboard. Instead of inputting each character the stylus crosses on the keyboard, the SHARK system treats the shape of a sokgraph as a pattern and recognizes it by a pattern recognition algorithm. The exact scale and location in which the sokgraph is produced, and hence the visual attention demand, are relaxed.

Efficiency: In comparison to hand writing based on alphabetic or logographic characters such as Chinese, writing a word pattern defined by a stylus keyboard can be much more efficient, with each letter constituting only one straight stroke and with the entire word as one shape. Indeed, sokgraph is a form of *shorthand*. Table 1 shows the sokgraphs of a common and critical[3] phrase "The quick brown fox jumps over the lazy dog" in comparison to the more traditional and well known English shorthand systems.

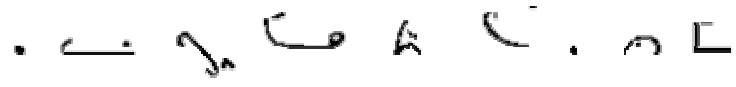
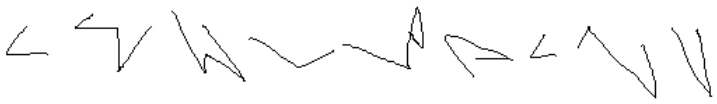
Long hand	*The quick brown fox jumps over the lazy dog*
Pitman shorthand	
Gregg shorthand	
Sokgraph/SHARK	

Table 1. The phrase "The quick brown fox jumps over the lazy dog" in traditional shorthand systems and in sokgraph. The Pitman and Gregg shorthand writings are reproduced from http://personal.riverusers.com/~busybee/handy/althandwriting.htm by permission of Eric Lee.

[3] This phrase is critical because it covers all 26 English letters in 9 words. Sokgraphs of these words are more stretched than usual as a result.

In theory sokgraphs can be defined on any keyboard layout. However if we define them on the QWERTY layout, for example, it would involve frequent left-right zig-zag strokes, because the commonly used consecutive keys are deliberately arranged on the opposite sides of QWERTY. Currently we use the *ATOMIK* layout, although a more suited layout can possibly be developed.

Duality: Traditional shorthand writing systems, such as Pitman's, take significant time and effort to master. Furthermore, one must sufficiently master a traditional shorthand system to begin actual use. A sokgraph, on the other hand, shares the same pattern as tapping on a stylus keyboard. A user could choose to use tapping or tracing individual letters or writing the sokgraph on the keyboard. Stylus tapping/tracing can serve as a bridge to sokgraph writing.

Common components and Zipf's law effect: Due to the effect of Zipf's law[4], mastering a small number (e.g. 100) of sokgraphs of the most common words can benefit the user at a disproportionably high frequency of use. Familiarity of the common fragments of words, such as *ing* and *tion*, even in a less common word, may also transfer to the user's general skill of writing sokgraphs.

Skill transition: One advantage of sokgraphs is that the user does not have to memorize any of them to begin. One can start using them by tracing the letters on the keyboard. In other words, it initially can be visually guided, closed-loop action. Over time, as the contribution of pattern recall, or open-loop action increases, their dependence on visual guidance will decrease. We expect a user's behavior to be always somewhere in between in general, but gradually shifts from closed-loop to open-loop performance with practice.

There are many previous research results related to *SHARK*. The idea of optimizing gestures for speed is embodied in the *Unistrokes* alphabet designed by Goldberg and Richardson (Goldberg & Richardson, 1993). In *Unistrokes* every letter is written with a single stroke but the more frequent ones are assigned with simpler strokes. If mastered, one can potentially write faster in the *Unistrokes* alphabet than in the Roman alphabet. The fundamental limitation of *Unistrokes*, however, is the nature of writing one letter at a time. *Quikwriting* (Perlin, 1998) uses continuous stylus movement on a radial layout to enter letters. Each character is entered by moving the stylus from the center of the radial layout to one of the eight outer zones, sometimes crossing to another zone, and finally returning to the center zone. The stylus trajectory determines which letter is selected. While it is possible to develop "iconic gestures" for common words like *the*, such gestures are rather complex due to the fact that the stylus has to return to the center after every letter. In this sense, *Quikwriting* is fundamentally a character entry method. *Cirrin* (*Cir*cul*ar In*put) (Mankoff & Abowd, 1998), is probably the closest prior art to *SHARK*. *Cirrin* operates on letters laid out on a circle. The user draws a word by moving the stylus through the letters. *Cirrin* explicitly attempts to operate on a word level – the pen lifts up at the end of each word. *Cirrin* also attempts to optimize pen movement by arranging the most common letters closer to each other. However, some of the key principles of *SHARK* such as open-loop *pattern* production rather than crossing individual keys, location and scale relaxation, and the transition from visual guidance to memory recall, are not in *Cirrin*.

SHARK Gesture Recognition

Many pattern recognition techniques have been previously invented for "online" handwriting recognition, including template matching, syntactical modeling, statistical modeling and neural networks. See (Tappert, Suen, & Wakahara, 1990) or (Beigi, 1993) for surveys of the common techniques. We have developed a *SHARK* recognition system based on the classic *elastic matching* algorithm (Tappert, 1982) which computes the minimum distance between two sets of points by dynamic programming. One set of points is from the shape that a user produces (sample gesture). The other is from a *prototype* – the sokgraph defined by the letter trace of a word on a keyboard. The word corresponding to the sokgraph

[4] Zipf's law models the observation that frequency of occurrence of some event *f*, as a function of its rank *i*, defined by the frequency, is a power-law function $P_i \sim 1/i^a$ with the exponent *a* close to unity. The effect of Zipf's law is that disproportionally large percentage of a body of text is made of the most common words.

that has the shortest distance to the sample is returned as the recognized word. See (Zhai & Kristensson, 2003) for more details of our recognition system and ways of disambiguating similar sokgraphs.

A Feasibility User Study

The *SHARK* method raises many human performance questions. We focus on the most basic question – can users learn, remember or discriminate, and produce sokgraphs at all? Can they learn a useful number of sokgraphs in a relatively short period of time?

There are reasons to expect that people can remember a large number of symbols. We use dozens of Roman letters, Arabic numbers, Greek letters, punctuation marks, and mathematic symbols. Trained stenographers learn a great number of shorthand symbols. 700 separate hieroglyph symbols were used in ancient Egypt (Gardiner, 1978). A literate Chinese person must learn at least two thousand unique characters.

We have conducted a feasibility user study on the learnability of sokgraphs. See (Zhai & Kristensson, 2003) for the details of the experiment design and execution. The results of the user study shows that all of the participants could learn to write correctly recognized sokgraph gestures for any word presented to them, if practiced enough (typically 7 to 15) times. As shown in Fig. 3, participants were able to correctly write more words in each learning session (in different days), on average about 15 words more per session. In the final test, on average they correctly produced 48.83 (between 62 and 39) words in their first attempt, and 58.67 (between 77 and 49) words if counting the second attempt when the first failed. Interestingly, the number of new words learned per session was rather constant.

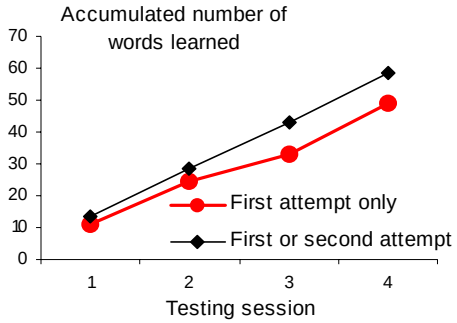


Fig. 3. Total number of words correctly written in test sessions, averaged across participants

During the study the participants were encouraged to write down comments on *SHARK*. Most of them found the method delightful, particularly in the initial sessions (“It is fun!” “Ought to be a really effective way of writing once you get a hang of it” etc). None of the participants found the method tedious during the study.

ACQUISITION OF NEW TEXT INPUT METHODS AND SKILLS

Characteristics that ease learning

In addition to inventing and studying methods that offer high performance for expert users, understanding ease of initial adoption and the ease of learning present an equally critical research challenge. There are many aspects to this challenge. One is that the method itself should be easy to start and improve. Finding characteristics that may ease learning is an important research topic. Compatibility with users’ existing skills, conceptual clarity of the new method and many other factors may play a role. A small step in this direction is the alphabetical tuning mechanism employed in the ATOMIK keyboards.

As reported in (Smith & Zhai, 2001), we conducted a study to test whether alphabetical tuning indeed could help novice users in the very initial stage of learning a new stylus layout. In the study, participants with no prior experience with stylus keyboards took part in an order-balanced within-subject experiment

with two layouts, one with alphabetical tuning and one without alphabetical tuning. With each layout, participants first tapped from the *a* to the *z* key as a brief warm-up. For the next 15 minutes, they entered memorable English sentences, such as "the quick brown fox jumps over the lazy dog", "we hold these truths to be self evident", and "all men are created equal." Results show that participants' average speed was 9% faster with alphabetical tuning.

To explain the empirical findings of alphabetical tuning, Smith and Zhai (2001) also conducted a theoretical analysis of the uncertainty in visual search, in the framework of the Hick-Hyman law (See (Keele, 1986) for a review). With the alphabetical tuning, novices may have a stronger expectation of the area that a letter is likely to appear (lower entropy), hence reduce their search time. This is particularly true to the letters at the beginning or the end of the alphabet (For example *a,b,c,d* and *x, y, z*).

Learning methods

The other aspect of easing user's learning experience is to design learning processes, perhaps eventually embodied in the form of a game for example, to help the user acquire the skills in a motivating fashion.

The literature on skill acquisition (e.g. (Proctor & Dutta, 1995)) and human memory (e.g. (Baddeley, 1998)) presents a variety of theories, models and insights on how people learn and retain skills. One compelling method from that literature that may be relevant and useful in helping people to learn novel text methods is practicing with expanding rehearsal interval (ERI) (Landauer & Bjork, 1978). ERI has been acclaimed as an important result in human memory research and is also supported by recent thoughts in the field of skill acquisition and memory (Schmidt & Bjork, 1992).

Briefly, the ERI method suggests that trial repetitions for learning should be neither totally massed nor randomly distributed. Rather, they should be optimized by systematically increasing the interval between repetitions. For example, a good way to learn a new foreign word is to rehearse the word frequently at first and gradually reduce the frequency of rehearsal as it becomes better memorized.

We have revised and applied the ERI method in two studies, one on learning stylus tapping (Zhai, Sue, & Accot, 2002) and the other on learning sokgraphs (Zhai & Kristensson, 2003). An important modification we made to ERI was to make it adaptive to user's learning speed. The rehearsal interval was expanded, kept the same, or even decreased depending on whether the user's performance indicated that the item practiced was correctly recalled. The basic goal was to keep the learner on the very edge of forgetting and hence enforce active retrieval of information from memory, which had been suggested as a key to memory retention (Schmidt & Bjork, 1992).

For example, in (Zhai, Sue, & Accot, 2002), our algorithm scheduled the reappearance of each training item (a letter, a digraph, or a word) with a gradually increasing interval. The interval for each item was increased only when the participant learned the item well enough to type it without spending a significant portion of the total typing time searching for the keys. This was judged by the algorithm with a Fitts' law prediction of the time needed to tap all the letters within that item. If the typing time was within Fitts' law prediction, the item was judged as being learned by the participant and the practice interval increased. If the participant took much longer to type the item, an error trial was registered and the item was immediately repeated. If the participant typed the item within the Fitts' law prediction the second time, the rehearsal interval for the item remained unchanged. If it took more than two trials to type the item with Fitts' law prediction, the rehearsal interval for the item was decreased.

A similar method has also been applied to training users to learn sokgraphs in (Zhai & Kristensson, 2003). In each cycle of rehearsal of a particular sokgraph, the participant was asked to write it in the stylus keyboard area without actually showing the keyboard layout. This forced the user to actively retrieve the sokgraph patterns from their memory. If correctly written, the word would be rescheduled to appear at an interval twice the current value. If the participant could not recall the sokgraph of a practiced word or failed to write it correctly, the rehearsal interval kept its current value. Fig. 4 shows the ERI traces of a few sample words by one participant. As we can see, the participant could keep up with the doubling interval with some sokgraphs (e.g. *other*) but slowed down with other sokgraphs (e.g. *was*) at some ERI

cycles. Our experience is that the ERI method is effective if applied appropriately, as indicated by the training results and by the study participants' subjective satisfaction.

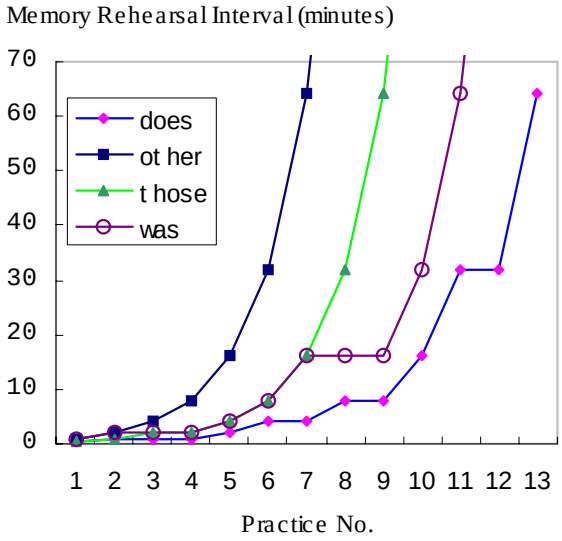


Fig. 4. Sample ERI traces of sokgraph *does*, *other*, *those* and *was*

We have also explored whether the learning curve of acquiring a new text input method could be accelerated using top-down learning strategies. In one experiment (Lee & Zhai, 2004), one group of participants learned a stylus keyboard layout with top-down methods, such as visuo-spatial grouping of letters and mnemonic techniques, to build familiarity with a stylus keyboard layout. The other (control) group learned the keyboard by typing sentences. The top-down learning group liked the stylus keyboard better and perceived it to be more effective than the control group. They also had better memory recall of the stylus keyboard layout. Typing performance after the top-down learning process was faster than the initial performance of the control group, but not different from the performance of the control group after they had spent an equivalent amount of time typing. These results suggest that the acquisition of declarative knowledge does not necessarily improve procedural skills, even during the initial cognitive stage of the virtual keyboard learning. They also suggest that top-down learning strategies can motivate users to learn a new keyboard more than repetitive rehearsal, without any loss in typing performance.

FUTURE DIRECTIONS

SHARK utilizes the expressive affordances of stylus and bridges visually guided performance (tapping or tracing) for ease of learning to recall-based open-loop actions (shorthand) for eventual high performance. While we believe SHARK, or systems embodying some of the principles in SHARK, could be a solution to off-desktop computing, many research challenges are still ahead of us. First, both the interface and the technology beneath SHARK need improvement. On the surface, further reducing visual attention in SHARK is an important design challenge. On the recognition technology side, we are developing robust sokgraph recognition algorithms that maximize the flexibility to user's gesture production and minimize recognition errors. Systematic evaluation along the dimensions outlined earlier in this paper has barely begun. Conceptually, how speed and accuracy tradeoff in gesture production and recognition is poorly understood. This is true for handwriting and other recognition technology in general. As the user produces faster but sloppier input, the risk of recognition errors will increase. At a certain point, the user's input may not contain enough information for any recognizer (humans included) to decipher. Can the amount of information in a user's gesture be theoretically quantified? Can an information theory be developed to guide sokgraph gesture interface design? Unlike stylus keyboards where Fitts' law provides

a simple baseline estimate of the ultimate performance, there does not exist any satisfying model for shorthand gesture based interaction. In parallel to this research program, we have undertaken research on "law of action" in order to be able to model multiple classes of human actions (Accot & Zhai, 1997; Zhai, Accot, & Woltjer, 2004), but the understanding so far is not enough to model sokgraph or similar gestures. Being the intimate interface between the user's expression and the computer's interpretation, gesture and similar interfaces await deeper theories that can guide the design and development of interfaces that helps the user and the machine negotiate at a level that does not require explicit attention.

Some broader issues with regard to interfaces for off-desktop computing also deserve attention. Will interfaces for off-desktop computing converge to some de facto standard as "QWERTYNomics" suggests? Or does the nature of off-desk computing inherently require diverse and specific solutions for each type of application? In short, the field is filled with interesting HCI research challenges.

ACKNOWLEDGMENT

Many colleagues and research intern students at the IBM Almaden Research have contributed to this research program, including Johnny Accot, Clemens Drews, Jon Graham, Michael Hunter, Paul Lee, Alison Sue and Jingtao Wang. We also like to thank the action editor of this paper for the very thorough and insightful comments on the initial draft of this paper.

REFERENCES

Accot, J., & Zhai, S. (1997). Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. *Proc. CHI 1997: ACM Conference on Human Factors in Computing Systems*, 295-302.

Accot, J., & Zhai, S. (2002). More than dotting the i's - foundations for crossing-based interfaces. *Proc. CHI 2002: ACM Conference on Human Factors in Computing Systems, CHI Letters 4(1)*, 73 - 80.

Accot, J., & Zhai, S. (2003). Refining Fitts' law models for bivariate pointing. *Proc. CHI 2003, ACM Conference on Human Factors in Computing Systems, CHI Letters 5(1)*, 193-200.

Baddeley, A. (1998). *Human Memory - Theory and Practice* (Revised Edition ed.). Boston: Allyn and Bacon.

Beichl, I., & Sullivan, F. (2000). The Metropolis Algorithm. *Computing in Science & Engineering, 2*(1), 65-69.

Beigi, H. S. M. (1993, July 24-25). An overview of handwriting recognition. *Proc. The 1st Annual Conference on Technological Advancements in Developing countries*, 30-46.

Binder, K., & Heermann, D. W. (1988). *Monte Carlo Simulation in Statistical Physics*. New York: Springer-Verlag.

Buxton, W., Billinghurst , M., Guiard, Y., Sellen, A., & Zhai, S. (1994/2002). *Human Input to Computer Systems: Theories, Techniques and Technology (book manuscript)*: available at http://www.billbuxton.com/inputManuscript.html.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Cooper, W. E. (Ed.). (1983). *Cognitive aspects of skilled typewriting*. New York: Springer-Verlag.

Crossman, E. R. F. W. (1959). A theory of acquisition of speed-skill. *Ergonomics, 2*(2), 153-166.

David, P. A. (1985). Clio and the Economics of QWERTY. *American Economic Review, 75*, 332-337.

Dvorak, A., Merrick, N. L., Dealey, W. L., & Ford, G. C. (1936). *Typewriting Behavior*. New York: American Book Company.

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

Gardiner, A. (1978). *Egyptian Grammar : Being an Introduction to the Study of Hieroglyphs (3rd edition)*: Aris & Phillips.

Getschow, C. O., Rosen, M. J., & Goodenough-Trepagnier. (1986). A systematic approach to design a minimum distance alphabetical keyboard. *Proc. RESNA (Rehabilitation Engineering Society of North America) 9th Annual Conference*, 396-398.

Goldberg, D., & Richardson, C. (1993). Touching-typing with a stylus. *Proc. INTERCHI, ACM Conference on Human Factors in Computing Systems*, 80-87.

Karat, C.-M., Halverson, C., Horn, D., & Karat, J. (1999). Patterns of entry and correction in large vocabulary continuous speech recognition systems. *Proc. CHI'99: ACM Conference on Human Factors in Computing Systems*, 568-574.

Keele, S. W. (1986). Motor Control. In K. R. Boff, L. Kaufman & J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (pp. 30.31-30.60). New York: John Wiley & Sons.

Landauer, T. K., & Bjork, R. A. (1978). Optimum rehearsal patterns and name learning. In M. M. Gruneberg, P. E. Morris & R. N. Sykes (Eds.), *Practical Aspects of Memory* (pp. 625-632). London: Academic Press.

Lee, P., & Zhai, S. (2004). Top-down learning strategies: can they faciliate stylus keyboard learning? *International Journal of Human-Computer Studies, to appear*.

Lewis, J. R. (1992). *Typing-key layouts for single-finger or stylus input: initial user preference and performance* (Technical Report No. 54729). Boca Raton, FL: International Business Machines Corporation.

Lewis, J. R., Kennedy, P. J., & LaLomia, M. J. (1999). Development of a Digram-Based Typing Key Layout for Single-Finger/Stylus Input. *Proc. The Human Factors and Ergonomics Society 43rd Annual Meeting*.

Lewis, J. R., LaLomia, M. J., & Kennedy, P. J. (1999). Evaluation of Typing Key Layouts for Stylus Input. *Proc. The Human Factors and Ergonomics Society 43rd Annual Meeting*.

Lewis, J. R., Potosnak, K. M., & Magyar, R. L. (1997). Keys and Keyboards. In M. G. Helander, T. K. Landauer & P. V. Prabhu (Eds.), *Handbook of human-computer interaction* (2nd ed., pp. 1285-1315). Amsterdam: Elsevier Science.

Liebowitz, S., & Margolis, S. E. (1996). Typing Errors. *Reason Magazine* http://reason.com/9606/Fe.QWERTY.shtml.

Liebowitz, S. J., & Margolis, S. E. (1990). The Fable of the Keys. *Journal of Law and Economics, XXXIII*.

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human computer interaction. *Human-Computer Interaction, 7*, 91-139.

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. *Proc. CHI'91: ACM Conference on Human Factors in Computing Systems*, 161-166.

MacKenzie, I. S., & Soukoreff, R. W. (2002). Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction, 17*(1).

MacKenzie, I. S., & Zhang, S. X. (1999). The design and evaluation of a high-performance soft keyboard. *Proc. CHI'99: ACM Conference on Human Factors in Computing Systems*, 25-31.

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

Mankoff, J., & Abowd, G. D. (1998). Cirrin: a word-level unistroke keyboard for pen input. *Proc. ACM Symposium on User Interface Software and Technology (UIST), Technical Note*, 213 - 214.

Mayzner, M. S., & Tresselt, M. E. (1965). Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic Monograph Supplements, 1*(2), 13-32.

McGuffin, M., & Balakrishnan, R. (2002). Acquisition of Expanding Targets. *Proc. CHI 2002: ACM Conference on Human Factors in Computing Systems, CHI Letters 4(1)*, 57-64.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M., N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics, 21*(6), 1087-1092.

Montgomery, E. B. (1982). Bringing manual input into the 20th century, *Computer* (pp. 11-18): IEEE.

Norman, D. A., & Fisher, D. (1982). Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors, 24*(5), 509-519.

Perlin, K. (1998). Quikwriting: Continuous Stylus-based Text Entry*. Proc. ACM Symposium on User Interface Software and Technology (UIST), Technical Note*, 215 - 216.

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

Proctor, R. W., & Dutta, A. (1995). *Skill acquisition and human performance*. Thousand Oaks, CA: Sage.

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

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

Sears, A., & Arora, R. (2002). Data entry for mobile devices: An empirical comparison of novice performance with Jot and Graffiti. *Interacting with Computers, 14*(5), 413-433.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal, 27*, 379-423, 623-656.

Shneiderman, B. (2000). The limits of speech recognition. *Communications of the ACM, 43*(9), 63-65.

Smith, B. A., & Zhai, S. (2001). Optimised Virtual Keyboards with and without Alphabetical Ordering - A Novice User Study*. Proc. INTERACT'2001 - IFIP International Conference on Human-Computer Interaction*, 92-99.

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

Tappert, C. C. (1982). Cursive Script Recognition by Elastic Matching. *IBM Journal of Research & Development, 26*(6), 756-771.

Tappert, C. C., Suen, C. Y., & Wakahara, T. (1990). The State of the Art in On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(8).

TextwareSolutions. (1998). *The Fitaly one-finger keyboard* (No. http://fitaly.com/fitaly/fitaly.htm).

Ward, D., Blackwell, A., & MacKay, D. (2000). Dasher - A data entry interface using continuous gesture and language models*. Proc. The 13th ACM Symposium on User Interface Software and Technology (UIST)*, 129-136.

Wobbrock, J. O., Myers, B. A., & Kembel, J. (2003). A High-Accuracy Stylus Text Entry Method*. Proc. ACM Symposium on User Interface Software and Technology, UIST'03 (CHI Letters)*, 61-70.

Yamada, H. (1980). A historical study of typewriters and typing methods: from the position of planning Japanese parallels. *Journal of Information Processing, 2*(4), 175-202.

Zhai, S., Accot, J., & Woltjer, R. (2004). Human Action Laws in Electronic Virtual Worlds - an empirical study pf path steering performance in VR. *Presence, to appear*.

Zhai, S., Conversy, S., Beaudouin-Lafon, M., & Guiard, Y. (2003). Human On-Line Response to Target Expansion*. Proc. ACM Conference on Human Factors in Computing Systems, CHI Letters 5(1)*, 177-184.

Zhai, S., Hunter, M., & Smith, B. A. (2000). The Metropolis Keyboard - an exploration of quantitative techniques for virtual keyboard design*. Proc. The 13th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 119-218.

Zhai, S., & Kristensson, P.-O. (2003). Shorthand Writing on Stylus Keyboard*. Proc. CHI 2003, ACM Conference on Human Factors in Computing Systems, CHI Letters 5(1)*, 97-104.

Zhai, S., Smith, B. A., & Hunter, M. (2002). Performance Optimization of Virtual Keyboards. *Human-Computer Interaction, 17*(2,3), 89-129.

Zhai, S., Sue, A., & Accot, J. (2002). Movement Model, Hits Distribution and Learning in Virtual Keyboarding*. Proc. CHI 2002: ACM Conference on Human Factors in Computing Systems, CHI Letters 4(1)*, 17-24.