DOI:10.1145/2330667.2330689

The Word-Gesture Keyboard: Reimagining Keyboard Interaction

By Shumin Zhai and Per Ola Kristensson

1. INTRODUCTION

Throughout human civilization, text has been an indispensable channel of communication. Modern computers equipped with desktop keyboards have dramatically increased the ease and volume of text-based communication in the form of email, text chat, and Web posting. As computing technologies expanded beyond the confines of the desktop, the need for effective text entry on mobile devices has been increasingly felt over the last two decades. Such a need has inspired both academic researchers and the information technology industry in pursuit of effective text entry methods alternative to the ubiquitous desktop keyboards. Since at least the early 1990s, mobile text input research can be found in almost every human-computer interaction conference. For example, a leading HCI journal dedicated an entire issue on text input in 2002.[27] Some of the influential academic research results in the past two decades include the Unistroke letter set,[13] optimized onscreen keyboard layouts,[12, 25, 29, 49] and soft keyboard error correction and prevention[14, 22] to name just a few (see more complete surveys in Kristensson,[18] MacKenzie and Soukoreff,[28] and Zhai et al.[50]).

To a very large extent, the design of text input method defined every major product in the evolution of modern mobile computing. As early as 1984, Casio released a wrist watch, the DB-1000, which had a capacitive touch screen with character recognition which enabled the user to operate the calculator or enter names and phone numbers into the databank, by using their fingertip to draw on the watch's screen. Nine years later, the Apple Newton became one of the first high profile mobile computing products, to feature handwriting recognition as a text input method.[43] The original 1996 Palm Pilot that successfully launched the PDA (personal digital assistants) industry differentiated itself from previous products with a single-stroke but Roman-letter like symbol set called Graffiti, that enabled handwritten characters to be entered more efficiently and less error-prone. The BlackBerry smartphones set a trend in the industry for many years with a miniature physical keyboard. Many versions of Palm Treo and Windows Mobile smartphones followed and further propelled the trend of miniature physical keyboards. These mobile devices also had a soft keyboard alternative, operated with a stylus on a resistive touchscreen. However, it was not until 2007, with the launch of the Apple iPhone that the use of a finger-operated capacitive touchscreen and soft keyboard became a primary text input method, one which is now the dominant form of text input on smartphones and tablets.

At some level, it is relatively easy to invent a new text entry method. After all, a text input method is a coding system for text communication. There can be potentially an infinite number of possible ways to code text by spatiotemporal means, including Morse code and the great many diverse writing systems of the world. However, to develop a mobile text entry method truly acceptable to the mass consumer market is exceptionally difficult for many reasons.

First, since text input is one of the most intensive and frequent human-computer interaction (HCI) tasks, speed is a very important consideration. Users are accustomed to fast keyboard typing on their own desktop or laptop keyboard.[20] A mobile text input method is ideally as fast as a desktop keyboard, or at least fast enough so the users do not have to defer text writing to a non-mobile setting.

Second, in order to gain wide adoption, a text entry method must impose minimal cognitive load on new users. This means that little or no learning should be required for users to start using a new text entry method. Most computer users have already invested time and effort in learning typing on Qwerty keyboard. A new method that requires even a fraction of that investment upfront is difficult for mass adoption.

Third, a successful new text input method should support development of proficiency – the ability to have performance improvement toward higher efficiency through practice in use. Unfortunately, ease of adoption and efficiency in user interface design are often at odds with each other due to having different cognitive foundations.[45] The alternatives too often reduce to determining where the load resides: easy to start but inefficient ever-after, or hard to learn but highly efficient as (hard-won) skill is acquired.

Addressing these HCI challenges requires more than applying the basic HCI research methods of usability testing and design iteration. Since the late 1990s, we have taken a research approach to text input that combines invention,[16,21,46] cognitive analysis,[46,47] human performance and statistical modeling,[1,7,21] and design, development, and deployment.[51] The result of this journey is a new paradigm that we call word shorthand gesture

This article is based on Zhai, S., Kristensson, P.O. Shorthand writing on stylus keyboard, *Proc. ACM CHI 2003*, 97–104, Kristensson, P.O. Zhai, S. SHARK^2: A large vocabulary shorthand writing system for pen-based computers, *Proc. ACM UIST 2004*, 43–52, and the authors' other publications.[15–17, 22, 23, 45, 47, 49–51]

keyboard, or word-gesture keyboard in this article. It is a re-imagination of the conventional key striking-based keyboard. The paradigm have been also known as shorthand-aided rapid keyboarding (SHARK),[21, 46] shape writer or shape writing,[23, 33, 47] and can also be called gesture, graph, stroke, trace, swipe, sweep, slide, or glide keyboard. This paradigm has not only been extensively researched in the academic literature[15, 17, 19, 21, 23, 46, 51] but has also already been embodied in many products. To date, different implementations of this novel paradigm have been marketed by a number of companies under at least the following brands: ShapeWriter, SlideIT, Swype, T9 Trace, FlexT9, and TouchPal on a great number of devices.

This article summarizes a decade-long academic research that led to the establishment of this input paradigm. We developed the basic concepts and initial prototype of a word-gesture keyboard from 2000 to 2002,[16, 46] and took many more years to mature and deploy the technology.[21, 23, 50, 47, 51] The paradigm itself is still emerging and developing, with both necessity and opportunity for further technological advances and better user behavior and performance understanding. We outline some of the future research directions at the end of this article.

2. THEORY, RATIONALE, AND DESIGN PRINCIPLES OF WORD-GESTURE KEYBOARDS

The basic type of input action on a traditional keyboard is striking an individual key. To do this well requires good tactile feedback. On a touch screen, another type of input action is possible. Instead of a striking action, one can use a continuous stroke gesture to convey information. Indeed, it is compelling to use sliding gestures on a touch keyboard for functions such as DELETE or SHIFT.[3] In early 1980's, Montgomery[32] conceived the idea of using sliding gestures on a touch keyboard to enter *characters*. He designed a "wipe activated" keyboard with a flat touch sensitive surface. The positions of the letter keys were carefully arranged to make consecutive letters commonly appear in words connected on the keyboard. The user can slide across adjacent letters to enter a string of letters. Montgomery believed such continuous "wiping" actions are more efficient hence "bringing manual input into the 20th century" from 1860's Qwerty keyboard. Perhaps ahead of its time, Montgomery's pioneering work had very limited impact, with only a few citations in the literature. Without further research or actual deployment, it was also unclear how easy or efficient it was to use such a keyboard which required detecting or remembering connected sequences of letters in order to wipe through them.

Stemmed from our work on optimizing stylus tapping keyboard,[48] we envisioned the paradigm of word shorthand gesture keyboard for touchscreen devices. On a word-gesture keyboard, instead of tapping individual keys or wiping through a sequence of letters connected on the keyboard, the user can write each and every word in a lexicon via a *word gesture* (also referred to as sokgraph—short hand on keyboard as a graph[50]). A word gesture approximately traces all letters in the intended word, regardless if they are adjacent. For example, to write the word *fun* a user touches the *f* key, slides to the *u* key then the *n* key, and lifts up. The resulting gesture is analyzed by a *statistical* model and the most likely word (in this case *fun*) is selected and entered by the system, which optionally also displays alternative N-best candidate words (Figure 1).

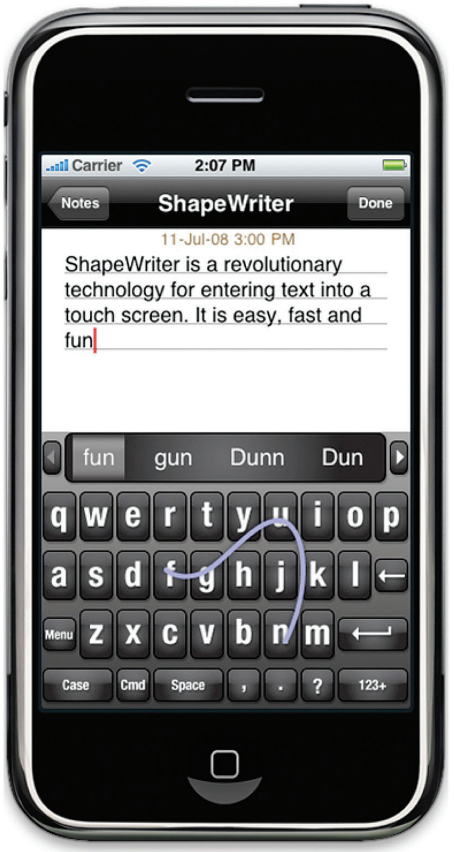
Note that the meaning of "word" in a word-gesture keyboard lexicon is broadly defined. While most words can be selected from a natural language, some can also be tokens defined by arbitrary strings of characters, such as gmail.com. Each such token in turn defines a word-gesture, or a token path, on the keyboard.

2.1. Gesture keyboard feasibility

The first question that may arise here is why the word-gesture keyboard paradigm is possible at all, considering that most word gestures will run across letters that are not part of the word intended. Indeed, this challenge seemed to have prevented the attempt by Montgomery in the early 1980s[32] toward establishing such a paradigm. Montgomery[32] instead proposed to rearrange the keys to maximize the chance for a user to be able to wipe through a sequence of adjacent letters that happen to make a word or a common word fragment without lifting.

However, this problem was not insurmountable. As Shannon[36] observed and elegantly demonstrated in his classic paper on information theory long ago, there are strong statistical regularities in natural languages. For example,

Figure 1. ShapeWriter on the iPhone is an example of a word-gesture keyboard.



some character sequences are more likely than others and most simply don't exist as legitimate words. The fundamental conceptual breakthrough to the word-gesture keyboard paradigm is that valid letter combinations form a finite set that can be captured in a language model, created by, for example, mining emails, blogs, and the Web. A very simple form of a language model is a lexicon—a list of all permissible words. In the case of English, a lexicon size of 20,000–100,000 words would be sufficient for most users. The words in a lexicon can be represented geometrically on a given keyboard layout as word gestures and matched against users' input gesture. Later in this paper, we explain how to efficiently classify and recognize such gestures.

Of course, an individual user may occasionally still need to write rare names and jargons, email addresses, or passwords that are out of vocabulary (OOV). Since a gesture keyboard enhances, rather than replaces, a conventional touchscreen keyboard, OOV letter sequences can always be entered by typing the individual letter keys. If these OOV sequences are frequently used then they may be added to the system's list of recognized words, either manually or automatically.

Occasionally, two words may share exactly the same starting letter, ending letter and trajectory in between (e.g., *tip* and *top* on Qwerty), causing a conflict. An analysis showed that of a 20,000 words lexicon had 537 conflicts on the Qwerty layout. This number reduced to 493 on the ATOMIK (see Figure 2) layout[49] of which 283 were Roman numerals.[21] Because they are rare, these conflicts can be addressed by manual selection from the alternative N-best suggestions, or automatically according to word context.

Having understood the technical feasibility of word-gesture keyboards, a considerable amount of research was still needed in understanding the human performance and user experience factors involved in using them. This required

Figure 2. Word-gesture keyboards can also work on alternative keyboard layouts. Shown here are ShapeWriter ATOMIK mode on the iPhone (circa 2008 top left), ShapeWriter on a Windows Tablet with the ATOMIK layout (circa 2005, top right), and an illustration of the ATOMIK layout (bottom).

conceptual analysis, controlled experiments, prototyping, and ultimately product deployment. In what follows, we first present some of the basic conceptual dimensions, rationales, and principles of gesture keyboards. Some of these were previously articulated in Zhai and Kristensson,[46, 47] but the following is synthesized with the benefit of hindsight and experience.

2.2. Efficiency

One continuous movement: In comparison to tapping-based touchscreen keyboards, gesture keyboards do not require up and down movements for each letter. Instead an entire word involves only one continuous movement. Anecdotal evidence from centuries of stenography research has pointed out the impeding effect on speed performance of repeated lifts.[30] From everyday writing, we also know that when we write fast, we write cursive—meaning multiple letters are linked as one continuous stroke. To a degree, the word gestures on a gesture keyboard in effect become a modern form of *shorthand* for words, akin to European shorthand systems.[30] Note that minimizing the number of separate actions was the main motivation in Montgomery's wipe-activated keyboard[32] and single-stroke shorthand for characters, such as Unistrokes, Graffiti, and their Roman antecedent, Notae Tironianae, developed by a slave of Cicero, Marcus Tullius, in 63 BC.[5]

The speed advantage of a single-stroke word gesture input, as opposed to single-finger (or stylus) tapping of individual letters of the same word, can also be understood in motor control modeling terms. Tapping individual letters in a word can be viewed as a sequence of discrete target pointing tasks, each can be modeled by Fitts' law.[11]

$$t_{k,k+1} = a + b\,ID \tag{1}$$

$$ID = log_2\left(\frac{D_{k,k+1}}{S} + 1\right) \tag{2}$$

where $t_{k,k+1}$ is the time duration from tapping the kth letter (key) to the $(k+1)$th letter in the word; $D_{k,k+1}$ is the movement distance from the kth letter to the $(k+1)$ letter; and S is the size of the target key. a and b are two constants of Fitts' law. ID is called Fitts' index of difficulty, measured in *bits*.

Similarly, as a baseline a word gesture on a keyboard can be viewed as a "continuous crossing" movement sweeping through a sequence of "goals". Each goal is a letter key needed in the word. According to the study of Accot and Zhai,[1] each goal-crossing task in this continuous crossing process also obeys Equation (1) but is faster (due to different a and b parameters) than tapping on the same sized targets as long as ID is less than 4 bits. On a keyboard layout such as Qwerty, the maximum ID (from one end of the keyboard to another) is less than 4 bits since each row of the keyboard has a maximum of 10 keys.

Rick[33] presents another Fitts' law-based model of word-gesture keyboard that takes the angles between different segments of the stroke into consideration. Cao and Zhai[7] developed a time complexity model of gesture strokes based on the corners, line segments, and curvatures (CLC) in a stroke and each type of elements is in turn modeled by motor control laws. The CLC model can make baseline predictions of the time efficiency of different gesture sets according to, for example, keyboard layout.

Auto word ending and spacing: Because a word-gesture keyboard works at the word level, there is a natural separation between words: each time a user lifts the finger from the touch surface, a word and a space are entered. According to our calculation based on the American National Corpus (http://www.anc.org/), the average length of an English word is 4.7 letters. This means that one in every 5.7 key strokes when typing English texts is devoted to entering spaces (or other punctuation keys). Not having to enter a space character after each word is another efficiency advantage of a gesture keyboard.

Error-tolerance: Since a word-gesture keyboard can perform error-tolerant gesture recognition, users do not have to precisely slide though every letter in the intended word. The input stroke only needs to be closer to the intended word gesture than other distractors as judged by the recognition algorithm. Error tolerance allows the user to cut corners, to be inaccurate but fast.

One finger operation: However, in comparison to two-handed typing (with ten fingers or two thumbs on the keyboard), a gesture keyboard also has a speed disadvantage. This is particularly true when the keyboard layout is the conventional QWERTY on which consecutive letters of a word tend to alternate between the left and right side of the keyboard. With two handed-typing, when one hand strikes one letter the other hand can, to some degree, move towards the next letter in parallel.[9] Such parallelism with bimanual typing is one speed advantage a gesture keyboard currently lacks.

2.3. Ease-of-use

For many reasons, a gesture keyboard is also easy to use. First, typing on a keyboard is a familiar text input method to most, if not all computer and smartphone users. A gesture keyboard can be viewed as a conventional touch keyboard that also affords gestures. Importantly, every gesture keyboard is still a tapping keyboard. Simultaneously, enabling tapping and gesturing behavior, without requiring even a switch, a gesture keyboard imposes a low adoption entry threshold.

Second, drawing or doodling is a fun and easy action that even children enjoy doing. A gesture is in some sense a more appropriate action than serial tapping on a conventional keyboard.

Third, the user does not have to have learned any gestures before using a word-gesture keyboard. As a beginner, the user simply slides the finger from one letter to another, driven by visual guidance to the next letter key on the keyboard.

When using a bare finger rather than a sharp stylus to operate a gesture keyboard, the fact that the finger is wider than the virtual keys on smartphones is an impediment to ease of use. For some beginners, this "fat finger" problem is particularly challenging because they may doubt that the letter under the finger is the correct letter. To address this concern, a version of the SHARK gesture keyboard for the Tablet PC had two keyboards, a sensing keyboard and a "phantom"

keyboard. When the user's finger or stylus moves on the sensing keyboard, the stroke ink that moves in parallel is displayed on the phantom keyboard that is not obscured by the hand. However, such a design was in our experience proven unnecessary after the first large-scale word-gesture keyboard (ShapeWriter WritingPad) release on the iPhone. Most users quickly gained confidence, stopped worrying about the letter underneath their fingertip, and realized they only need to approximately cross the intended letters.

2.4. Progression from ease to efficiency

One of the most important rationales of gesture keyboards lies in facilitating transition from ease to efficiency.

Writing with a gesture keyboard is a mixture of two types of behavior. The first type, used by beginners or for unfamiliar words, is letter-to-letter tracing. Such a process is visually guided, closed-loop, and relatively slow. This visual recognition-based process is easy because it does not require any prior memory. The second type, used by proficient users for familiar words, is memory-driven gesturing. This process in contrast is recall-driven, open-loop, efficient, and fast.

The two types of behavior are two ends of a continuum. Our main behavioral theory of word shorthand gesture keyboards is that their use automatically shifts from the ease end (visual tracing) to the efficient end (recall gesturing) (Figure 3).

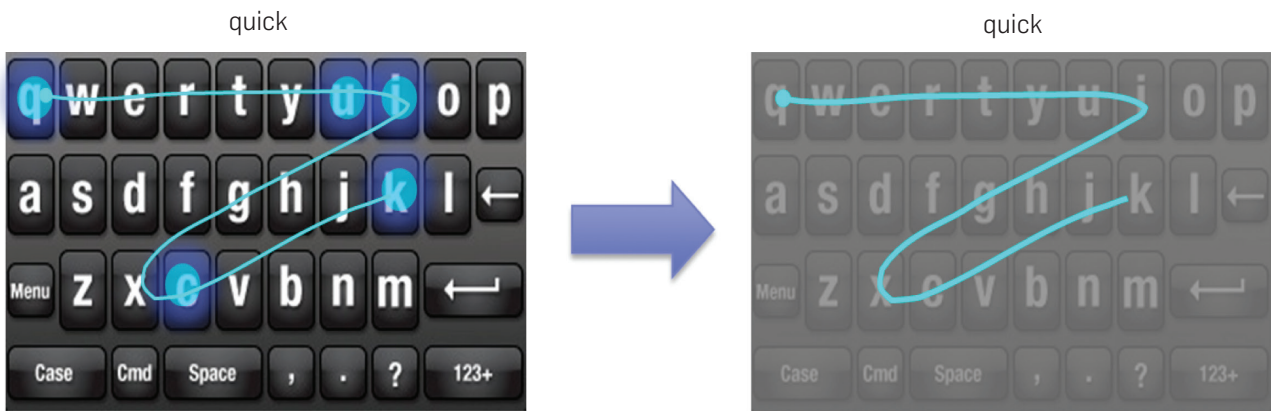
There are many factors facilitating such a shift. First, at both ends of the continuum or anywhere in between, the movement pattern is the same. The consistent movement pattern for the same word helps the shift from visual tracing to recall gesturing. On this point, we drew inspiration in Kurtenbach and Buxton's work on "marking menu" design,[24] although a direct application of marking menus to text input did not necessarily result in a successful text input method.[40] With marking menus the user can either wait for a visual radial menu to pop up, and then slide to the desired slice, or make a gesture in the same direction without the visual menu display if the angular gesture is remembered. As observed by Kurtenbach and Buxton,[24] the consistent movement patterns in the two distinct states of marking menus facilitate novice to expert mode transition in marking menu use. The basic psychology literature on automaticity in human behavior also shows that the key to developing skilled, low attention, automatic behavior lies in consistent mapping from stimuli to response.[35, 38]

In using a word-gesture keyboard, the production of movements increasingly changes from focusing on individual letters to connecting multiple letters into a word gesture. In other words, it shifts from smaller chunks to larger chunks in human performance.[4, 31] Chunking is another factor that facilitates the shift from tracing to gesturing.

In regular keyboard typing, users also develop mental word pattern representations.[44] This is evident from the fact that users type common words faster than random letters sequences. However, in a gesture keyboard the word pattern representation is a fluid continuous stroke and visually displayed, which plausibly ingrains the word patterns in users' memory much faster than learning common motor control schema for ten-finger typing.

Further research is required in understanding user performance and behavior in word-gesture keyboarding, particularly from the perspectives of two separate psychological research fields: human memory and human motor control. In general, human memory research distinguishes memory into declarative memory and procedural memory.[39] Declarative memory is about knowledge and facts and is explicit. Procedural memory on the other hand is about skills and how to do things, particularly body movements. Procedural memory is unconscious or implicit. The word-shorthands in gesture keyboarding are likely to involve both declarative and procedural memory, shifting in contribution from the declarative side to the procedural side and falling below conscious awareness. Anecdotally we observed that experienced users often were not explicitly aware the token paths on the keyboard of the words they gesture. Similarly, motor control and learning research suggests that voluntary actions are initiated by a conscious goal, but the perceptual-motor integration, sequencing, spatial representation and movement dynamics are outside of awareness.[41] Procedural memory and motor skills are typically long lasting. Skills

Figure 3. Illustration: word-shorthand gesture keyboarding is expected to shift from primarily visual-guidance driven letter-to-letter tracing to memory-recall driven gesturing.



such as bicycling or skiing, once learned, are hardly ever forgotten.[34] It is our experience that we could still remember and write word gestures proficiently on a unique layout (ATOMIK) that we had not seen or used for months or even a year.

Importantly, we do not expect the users to gesture every word without looking at the keyboard. Due to the Zipf's law effect, a small number of words are used disproportionally frequently and their stroke patterns are memorized early. Longer and less common words are typically made of common fragments whose shapes can be quickly remembered. Even proficient gesture keyboard users are likely to use a mixture of visual guidance from the keyboard and memory-driven production of gesture shapes. The degree of each depends on experience with the specific words. We will show empirical findings in word-gesture memory and learning later in the paper. An important word-gesture keyboard property is that it does not force the user into either "mode". The user gradually progresses from the easy end to the more efficient end in use. In this sense, a word-gesture keyboard is a "progressive user interface."[45]

3. GESTURE RECOGNITION

Conceptually, gesture recognition is done by identifying the word which has the highest probability given the user's gesture. This search problem can be formulated using Bayes' theorem:

$$\hat{W} = \arg\max_{W} \frac{P(G|W)P(W)}{P(G)}, \quad (3)$$

where $P(G|W)$ is the likelihood of W's word gesture matching a user's input gesture G, and $P(W)$ reflects the system's estimate of prior probability that the word W is the user's intended word. The denominator $P(G)$ only depends on the user's gesture and is invariant during the search. Satisfying Equation (1) is equivalent to:

$$\hat{W} = \arg\max_{W} P(G|W)P(W). \quad (4)$$

The search for the user's intended word is thus the product of two model estimates. The probability $P(G|W)$ reflects the gestural model and the probability $P(W)$ reflects the language model. Different methods can be used to compute the gesture likelihood $P(G|W)$ and the language model prior $P(W)$.

In order to estimate $P(G|W)$, we have used various techniques, such as dynamic time warping and template matching, to compute gesture keyboarding shape similarities.[21, 46] In principle, a user drawn gesture is compared with all word gesture representations for all words in the lexicon. In practice, the vast majority of words in the lexicon are highly unlikely to correspond to the user's intended word. Thus, to achieve real time performance the search is limited to the most likely candidates using various well-known search strategies, such as indexing and pruning.

To compute $P(W)$, various language modeling techniques, such as long-span language modeling with smoothing can be used.[8] In our experience, unigram frequencies in a lexicon alone provide significant power.[21, 46]

One of our special efforts in recognition algorithm design is making gesture keyboards friendly to both beginners and proficient users according to the ease-to-efficiency progression principle outlined earlier. In a version of our implementation,[21] the weight of gesture recognition shifts from local features (as determined by the location of various points of the gesture) to the global gesture shape according to the behavior of the user. Specifically, if the user is unfamiliar with the gesture shape of the word W therefore has to slide from one letter to another by visual tracing, the total time of writing W on the keyboard can be estimated according to the summation of Fitts' law time from one letter to the next (following Equations 1 and 2):

$$t_n(W) = Na + b\sum_{k=1}^{N-1} log_2\left(\frac{D_{k,k-1}}{S}+1\right) \quad (5)$$

where $t_n(W)$ is the normative time to trace the word W; N is the number of characters in W; $D_{k,k-1}$ is the distance from the kth character to the $(k+1)$th character in W on the keyboard; and S is the size of the $(k+1)$ key. a and b are two Fitts' law constants.

When an input stroke is compared against the word gesture of W, the ratio of the stroke's total time t_s and the normative tracing time $t_n(W)$ can be used to adjust the recognizer's relative weight on local features vs. the global shape features. If t_s is shorter than $t_n(W)$ and if the user is indeed intending to write the word W, the user must be demonstrating a degree of shape-memory driven gesturing. We therefore can place more recognition weight on the global shape feature and less on the visually dependent local features. Such a shift can be automatic and continuous (not binary) according to the degree of acceleration from Fitts' law prediction, and word candidate specific.[21]

4. FUNCTIONS AND SYSTEMS

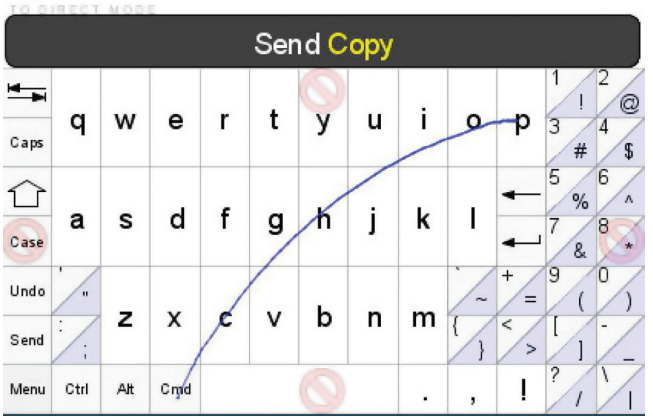
While a practical keyboard includes many functions and features, we highlight two particularly notable novel functions in some of the gesture keyboard systems we designed and developed—"command strokes" and the Case key.

4.1. Command strokes

The concepts and paradigm of text input outlined above can also be applied more broadly to commands and user interfaces in general. Commands are often interleaved with text input. For example, users may need to edit text (copy/paste), activate application functions (such as *Save*), or switch the language from one to another (such as from English to French). We extended the gesture keyboard paradigm so that it could support both text entry and command activation in one system.

With our systems, the user may issue commands (such as "Copy" and "Paste") by tracing out the command names on the keyboard starting from a designated key (e.g. a Cmd key). For example, *Cmd-c-o-p-y* copies selected text and *Cmd-p-a-s* pastes the text. Command recognition was made incremental so *Cmd-c, Cmd-c-o, Cmd-c-o-p* and *Cmd-c-o-p-y* all issue the same command. The system suggests the

Figure 4. Command strokes: gesture *Cmd-c-o-p* could send *Copy* command to the OS.



command effect as soon as the command stroke is unambiguous[15] (Figure 4).

4.2. Case key

Most of the time the case of a word (lower, upper or title) can be determined automatically in modern text input systems, particularly word-based systems. For example, in English the first word in a sentence and proper nouns are typically capitalized. However, there are exceptions to these normal rules. Automatic casing makes the use of the legacy Shift key unnecessary most of the time, but not all the time. This situation makes it difficult for the user to decide if to press the Shift key before entering a word. To correct the case of a word afterward with the Shift key is even more cumbersome because the user has to first select the text to be modified, delete it, and then use the Shift or CapsLock keys to trigger a mode change, and finally retype the text.

We introduced a new key on the keyboard, the Case key (see the lower left corner of Figure 1). This key cycles through the different word case alternatives for the word just entered or preceding the text caret. The Case key uses dictionary information to intelligently support nonstandard casing convention for some words, such as "iPhone". Since the Case key modifies the word preceding the current text caret position ("reverse Polish") it enables users to perform case corrections after the word is entered and only when they are actually needed.

4.3. Systems

We have designed and implemented many versions of experimental gestures keyboard systems, variably named HSK,[16] SHARK,[46] and SHARK2[21] which was publicly released from the IBM AlphaWorks site in 2004 (Figure 5). Until recently both CPU and memory were limited on mobile devices. But with indexing and aggressive pruning it was still possible to achieve real-time performance. For example, one of the first mobile versions of gesture keyboards we implemented could store both the gesture and the language model for 50,000 words in 450K memory and return recognition results with less than 20ms average latency on a phone equipped with a 32-bit 168 MHz Texas Instruments OMAP1510 CPU (Figure 6). We also led the design and development of a commercial version of word-gesture keyboard, ShapeWriter, released on the iPhone, Android and Window Mobile platforms in many languages.[37] These systems reflected increased maturity and practicality, as well as the mobile platform hardware and software constraints at the time. Working with platform and technical constraints was a part of a journey of research and innovation.

5. EMPIRICAL RESEARCH

One would imagine it is simple to determine a new text input method's efficacy by measuring the average user's average speed. An example to the contrary is the decades' old debate of QWERTY versus the Dvorak simplified keyboard that spilled over even into economic theories.[10, 26] It is difficult to design and execute decisive tests for text entry. There are many reasons for this challenge, including learning, speed-accuracy trade-off, and the multifaceted nature of use quality.

Figure 5. Shorthand aided rapid keyboarding (SHARK). The first publicly released fully functioning word-gesture keyboard (October 2004).

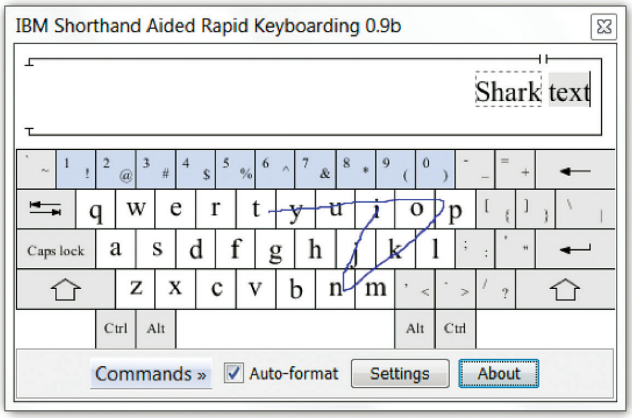
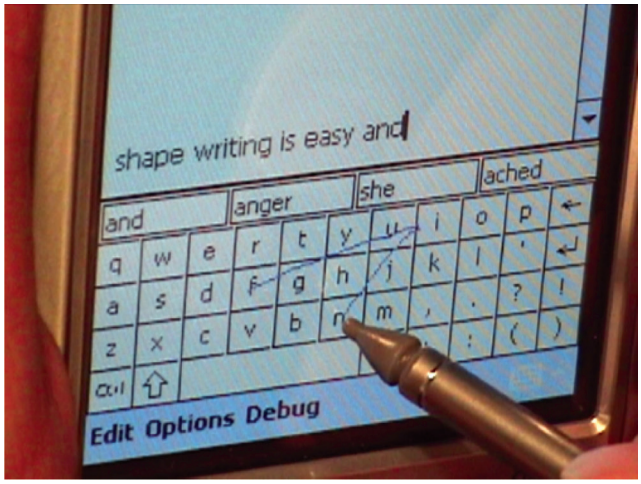


Figure 6. One of the first implementations of mobile word-gesture keyboard running on a mobile device with a 168MHz processor in real time (2006).



Instead of aiming for one decisive study, we have evaluated gesture keyboards in two approaches. First, we have conducted a series of lab-based experiments in order to understand different aspects and the fundamental potential of word-gesture keyboards. Second, we have implemented gesture keyboard systems and released them to the public, starting in 2004. This has enabled us to study how users in the real world perceive the technology.

5.1. Gesture memory and learning

The experiment in Zhai and Kristensson[46] tested a core premise of word-gesture keyboarding that users are able to recall and reproduce the gesture strokes of well-practiced words with little visual guidance of the keyboard. The ATOMIK keyboard layout,[49] illustrated in Figure 2 and used in that experiment, was previously unfamiliar to the participants. In each session with the visual keyboard blanked, the participants were first asked to recall and reproduce the gesture strokes of the words practiced in previous sessions. They could make a second attempt if the first attempt of drawing the gesture on the blank interface did not match the target word by the system's shape-based gesture recognizer. For 40min after the test, they practiced word gestures they had not mastered through a spaced-repetition schedule. The results showed that each participant learned on average 15 word gestures per session. In the final test after a total of four sessions, the participants correctly produced on average about 50 (between 39 and 62, mean 48.8) words in their first attempt, and about 60 (between 49 and 77, mean 58.7) words including the second attempt when the first failed (Figure 7).

While the experiment is an artificial lab study that may not exactly correspond to users' practical experience of learning word-gesture keyboarding, it nonetheless shows that it is possible to memorize the shape aspects of a gesture as defined by a keyboard and reproduce them without relying on the keyboard's visual display. Fifty to sixty does not seem to be a very large number of words, but the most common 50 words in English cover 40% to 50% of word occurrences in common English. The less common and longer words typically consist of common word fragments whose shapes may be mastered first hence still help the user to rely less on the visual guidance of the keyboard.

5.2. Initial user performance

In another lab experiment, we measured users' gesture keyboarding performance in their first 40min of use. On a familiar Qwerty layout, participants' average speed reached 15, 20, and 25 words per minute (wpm) after 5, 20, and 40min of practice, respectively, at a 1.1% error rate (Figure 8). There were considerable individual performance differences in word-gesture keyboarding. The fastest participants surpassed 40 wpm by the end of the 40 minute experiment.[17]

5.3. Ceiling performance

Typing competition was a common method of demonstrating typewriter quality in the mechanical typewriter days. Typing competition's results are often affected by the rules and context of the competition, but nonetheless the record

Figure 7. The number of word gestures successfully reproduced without looking at a keyboard after each session of practice.

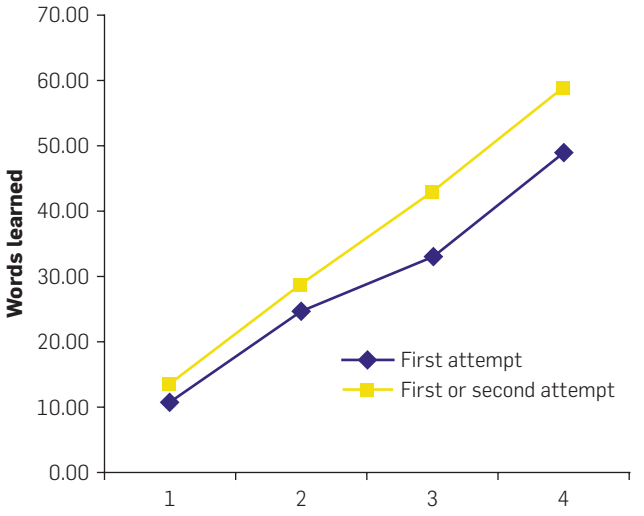
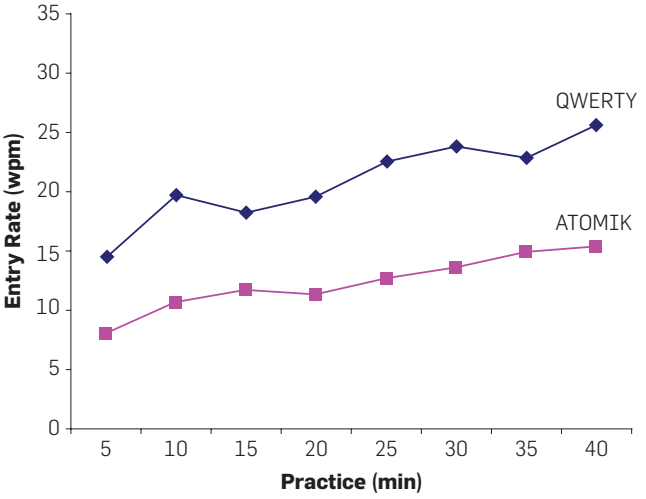


Figure 8. Ten novice users' average speed of writing random common phrases on a gesture keyboard in the first 40 min, at 1.1% error rate.



setting method reveals the top range of performance possible with a given text input method. Similarly, the peak error-free one sentence speed that can be achieved with a given input method reveals one aspect of the method's potential. To measure the top speed possible, we had ten participants practicing five common phrases such as "Thanks for taking care of this" and "Look forward to seeing you soon" on the SHARK gesture keyboard for 15min and tested their peak performance for 10min. The recorded error-free peak phrase speed averaged across the ten participants was 57.5 wpm while the top performer achieved 99 wpm.[17]

It is important to note that all of the above performance measures should be understood with the conditions they were collected in mind. These performance measures also depend on the underlying recognition algorithms implemented, the lexicon size and language model embedded

in the gesture keyboard, and the layout of the keyboard. A better optimized layout, smaller lexicon, and more error tolerant algorithms would afford higher performance. Many more theoretical and empirical questions regarding word-gesture keyboard's learning, initial performance, and ceiling speed are to be answered in the future.

6. SUBJECTIVE EVALUATION AND USER ACCEPTANCE

We have also collected subjective ratings in small-scale and short-term lab studies. In comparison with physical thumb keyboard (the dominant mobile input method at the time of the study), the word-gesture keyboard was on average considered more preferred, more fun, and less physically but more visually demanding. It is difficult to measure the total user experience of a technology in the lab. Fortunately, the emergence of a new generation of touchscreen devices such as the iPhone and Android devices and the fact that we had developed our research into a practical product via a start-up company, ShapeWriter Inc, enabled us to collect valuable and rarely available data on real users' perceptions of a new technology.

We gathered this information by examining user reviews for our publicly released ShapeWriter application on the iOS platform. It was submitted on July 7, 2008 to Apple's AppStore and released to the public on July 14, 2008. Since iOS prevented apps from replacing the built-in keyboard, we built a note-taking app initially named "WritingPad" (Figure 1). In addition to note-taking, the app allowed users to send their notes via email and SMS.

After the release, we analyzed the first 556 public user reviews on Apple's AppStore and reported the analysis in Zhai et al.[51] Of all of the 556 reviews, 81.6% were completely positive, 12.5% were somewhat positive, and 5.9% were completely negative.

Some of the comments were highly enthusiastic. For instance, *"Game changing app"* by jhudge05: *"Typing on the iPhone used to [sic] tedious and frustrating for me, but now that I use WritingPad I am actually writing faster on the iPhone than I was on my Blackberry",* and *"Holy $41t"* by Corso123: *"'revolutionized typing' is the understatement of the year. This technology should be part of every keyboard on all touchscreens. Someone nominate these software developers for a Nobel. No Joke. Thank you so much for this software... –brian."*

There were individual differences in the reviews. Some users stated that they quickly become proficient with the technique (*"It's super accurate and super easy to use and I'm still in awe of how genius it is."*), while others had trouble getting used to it (*"It took me a few days of use to get used to it"*).

Interestingly, users' opinions were also split on the impact of the so-called "fat finger problem." For some users, the gesture keyboard was an enabler: *"I have 'fat finger symdrome' and cannot type on the Iphone. Thank goodness for this program! Now, I can actually write emails!",* and *"Works great for people with large fingers like myself. Very liquid and intuitive. Brilliant Application."* However, other users had the opposite experience: *"ShapeWriter's on screen key pad, when used with a stylus, works great. But with my big fat finger, its more like sewing on a button while wearing boxing gloves."*

Other comments pointed out bugs and deficiencies, which helped us refine the software. Many reviewers wrote affectionate responses with words like *love, omg, fun, great, rocks, awesome, amazing, exciting, pleasant, cool, addictive, stunning, astounding,* and *fantastic*.[51]

Since the initial iPhone release on July 14, 2008, ShapeWriter has also been released for Google Android and Windows Mobile devices. In addition to user comments, our publicly released gesture keyboard systems (called SHARK Shorthand in 2004 and ShapeWriter in 2007–2010) were positively reviewed in newspapers and blogs. The first press mention was by *San Jose Mercury News* and *Seattle Times* in April 2003 and later by *The New York Times, CNET, BBC World News,* and *Die Zeit* in 2004–2007. Before the acquisition by Nuance Communications Inc, ShapeWriter Inc as a company also won a number of awards and recognition including Google's Android Developer Challenge Award, *Time.com's* top 11 iPhone must have applications, and Razorfish's top 10 mobile technologies to watch.

Since our first public release of a word-gesture keyboard, SHARK Text, in 2004, many other similar offerings have followed suit. Notable products include ShapeWriter, Swype, SlideIT, T9 Trace, FlexT9, and TouchPal. Together, these products have created popular awareness of an alternative paradigm for touchscreen text input, and today many people use them for their daily communication activities.

7. FUTURE DIRECTIONS

The word-shorthand gesture keyboard project has produced a wide range of results from which we attempt to piece together a coherent but simplified account in this article. Throughout the project, we tried to bridge invention with science, practical product design and development with theory-driven research, and application of modern computing techniques with human performance insights and modeling. We drew inspirations from theoretical HCI thoughts in, for example, Buxton's work on user learning.[3, 24] We frequently applied methods, models or at least the spirit of a school of thought in HCI spearheaded by the classic monograph of Card, Moran and Newell.[6] This school of thought bases human-computer interaction design on psychological insights embodied in approximate human behavior and performance regularities, rules, equations and models. We also exploited to a degree we could the power of statistical approaches to information processing rooted in classic information theory,[36] but enabled and modernized as computational power increases to a level on mobile devices impossible only a few years ago.

Although a new paradigm of information input has been established and embedded in many mainstream products, we believe this paradigm is still in its first generation of evolution. Significant advances in research and innovations can be expected in the years to come.

First of all, we only have an incomplete understanding of the user performance of word-gesture keyboards. Deeper perceptual-motor and cognitive studies are needed. For example,

we still do not have an accurate predictive model of users' transition from recognition-based tracing to recall-based gesturing. Modern human motor control and learning theories have made great progress in the last decades.[34, 41, 42] Leveraging findings and insight from that literature to make specific gesture keyboard design and analysis decisions offer opportunities for deeper research.

Particular lacking to date is a rigorous quantification of gesture space density as a function of the keyboard layout and the size of the lexicon. Without such a model it is difficult to fully understand error rate as a function of speed-accuracy trade-off. "Sloppy" gestures tended to be faster but also more error-prone. Exact or statistical modeling of gesture keyboard's speed-accuracy trade-off incorporating human control behavior is another important future research topic.

Also critically lacking in the literature to date is large-scale data logging and analysis of word-gesture keyboards in everyday use, which may provide not only a more complete understanding of user behavior but also data for large-scale machine learning of gesture keyboard algorithms and their parameters. Such work of course requires significant infrastructure and privacy preservation efforts.

The core technology of a word-gesture keyboard can conceivably be improved by using larger and long-span language models that take into account several previous words of context when they compute the language model's prior belief in a word candidate. However, the trade-off between the language model's size and efficacy remains an open question in the case of word-gesture keyboards. The spatial model of gesture keyboards should also be more broadly explored and tested. We have only explored a certain type of simple and efficient local (location) and global (shape) features for gesture keyboard recognition, but a variety of features can be invented in the future, particularly given the non-stop improvements in processing speed and memory capacity of mobile devices.

Gesture keyboards can also be used with other modalities. For example, if gestures can be effectively delimited they may be incorporated into eye-tracking systems or 3D full-body motion tracking systems, such as those used in Microsoft game products. Gesture keyboards can also be potentially integrated with speech input. In fact, there is already an experimental system that simulates the effects of a word-gesture keyboard combined with speech.[19]

We have alluded to the keyboard layout issue several times in this paper. For ease of adoption, Qwerty is a necessary default layout. It is very clear that the efficiency of word-gesture keyboards can be significantly improved if the keyboard layout is optimized. Qwerty is inefficient for word-gesture keyboarding because the gesture strokes frequently zigzag between the left and right over a relatively long distance. For this reason, we would want the keyboard to be arranged so that frequent letter-key pairs tend to be closer to each other. The layout of a gesture keyboard can also be optimized toward ambiguity minimization, so that word gestures are more distinct from one another. Not only would this make gesture keyboards more error-tolerant, but also facilitate ease to efficiency progression since gestures defined on such a layout should be more distinguishable. How to optimize the layout toward multiple objectives is another open question.[2] Even more challenging is how to get users realize the benefits of an optimized layout and quickly learn them in perhaps a playful fashion.[23]

Acknowledgments

This article is a synthesis of a set of previous publications.[15–17,21–23,45–47,49–51] We thank IBM Research, Linköping University, and many friends and colleagues for their years of support and contribution. Without their appreciation of innovation for long-term impact this sustained research program would not have been possible. We thank Stu Card and Bill Buxton for their insightful comments and suggestions that have greatly improved this article. We also thank Kelly Tierney of IBM Corporate Design who rendered the illustration in Figure 2.

References

1. Accot, J., Zhai, S. More than dotting the i's – foundations for crossing-based interfaces. In *Proceedings of CHI 2002: ACM Conference on Human Factors in Computing Systems, CHI Letters* 4,1 (2002), ACM, 73–80.
2. Bi, X., Smith, B.A., Zhai, S. Multilingual touchscreen keyboard design and optimization. *Hum. Comput. Interact.* (2012), To appear (available online at http://www.tandfonline.com).
3. Buxton, W., *Human Input to Computer Systems: Theories, Techniques and Technology*, book manusript, available at http://www.billbuxton.com/inputManuscript.html
4. Buxton, W. Chunking and phrasing and the design of human-computer dislogues. In *Proceedings of IFIP World Computer Congress* (Dublin, Ireland, 1986), 475–480.
5. Buxton, William (2005). Piloting Through the Maze. *Interactions Magazine*. 12(6), November + December, 10
6. Card, S., Moran, T., Newell, A. *The Psychology of Human-Computer Interaction*, Lawrence, Erlbaum Associates, Hillsdale, NJ, 1983.
7. Cao, X., Zhai, S. Modeling human performance of pen stroke gestures. In *Proceedings the ACM CHI conference on Human factors in computing systems* (2007), ACM, 1495–1504.
8. Chen, S.F., Goodman, J. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics* (1996), Association for Computational Linguistics, Santa Cruz, CA, 310–318.
9. Cooper, W.E., ed. *Cognitive Aspects of Skilled Typewriting,* Springer-Verlag, New York, 1983.
10. David, P.A. Clio and the economics of QWERTY. *Am. Econ. Rev. 75* (1985) 332–337.
11. Fitts, P.M. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psychol. 47*, 6 (1954) 381–391.
12. Getschow, C.O., Rosen, M.J., Goodenough-Trepagnier, C. A systematic approach to design a minimum distance alphabetical keyboard. In *Proceedings of RESNA (Rehabilitation Engineering Society of North America) 9th Annual Conference* (Minneapolis, MN, 1986), 396–398.
13. Goldberg, D., Richardson, C. Touching-typing with a stylus. In *Proceedings of INTERCHI, ACM Conference on Human Factors in Computing Systems* (Amsterdam, The Netherlands, 1993), ACM, 80–87.
14. Goodman, J., Venolia, G., Steury, K., Parker, C. Language modeling for soft keyboards. In *Proceedings of International Conference on Intelligent User Interfaces (IUI'02)* (2002), ACM, 194–195.
15. Kristensson, P.O., Zhai, S. Command strokes with and without preview: Using pen gestures on keyboard for command selection. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems* (San Jose, CA, 2007), ACM, 1137–1146.
16. Kristensson, P.O. Design and evaluation of a shorthand-aided soft keyboard. Master's thesis, Linköping University, Sweden (2002), p. 96.
17. Kristensson, P.O. Discrete and continuous shape writing for text entry and control. PhD Thesis, Linköping University, Sweden (2007).
18. Kristensson, P.O. Five challenges for intelligent text entry methods, In *AI Magazine* (2009), 30(4) 85–94.
19. Kristensson, P.O., Vertanen, K. Asynchronous multimodal text entry using speech and gesture keyboards. In *Proceedings of 12th Annual Conference of the International Speech Communication Association* (2011), ISCA, 581–584.
20. Kristensson, P.O., Vertanen, K. Performance comparisons of phrase sets and presentation styles for text entry evaluations. In *Proceedings of the 17th ACM International Conference on Intelligent User Interfaces* (2012), ACM, 29–32.
21. Kristensson, P.O., Zhai, S. SHARK2: A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST 2004)* (2004), 43–52.
22. Kristensson, P.O., Zhai, S. Relaxing stylus typing precision by geometric pattern matching. In *Proceedings of ACM International Conference on*

Intelligent User Interfaces (2005), ACM, 151–158.
23. Kristensson, P.O., Zhai, S. Learning shape writing by game playing. In *Proceedings of CHI '07 extended abstracts on Human factors in computing systems* (San Jose, CA, USA, 2007), ACM. 1971–1976.
24. Kurtenbach, G., Buxton, W. Issues in combining marking and direct manipulation techniques. In *Proceedings of ACM symposium on User Interface Software and Technology* (1991), 137–144.
25. Lewis, J.R., Kennedy, P.J., LaLomia, M.J. Improved Typing-Key Layouts for Single-Finger or Stylus Input, IBM Technical Report TR 54.692, 1992.
26. Liebowitz, S.J., Margolis, S.E. The Fable of the Keys. *J. Law Econ. 33*, 1 (1990), 1–25.
27. MacKenzie, I.S., ed. *Special issue on text entry for mobile devices. Hum. Comput. Interact. 17* (2002).
28. MacKenzie, I.S., Soukoreff, R.W. Text entry for mobile computing: Models and methods, theory and practice. *Hum. Comput. Interact. 17*, 1 (2002).
29. MacKenzie, I.S., Zhang, S.X. The design and evaluation of a high-performance soft keyboard. In *Proceedings of CHI'99: ACM Conference on Human Factors in Computing Systems* (1999), 25–31.
30. Melin, O.W. *Stenografiens historia.* P.A. Norstedt & Söner, Stockholm, Sweden, 1927/1929.
31. Miller, G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychol. Rev. 63* (1956), 81–97.
32. Montgomery, E.B. Bringing manual input into the 20th century, *Computer* (1982), 15(3) 11–18.
33. Rick, J. Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop. In *Proceedings of 23nd Annual ACM Symposium on User Interface Software and Technology* (2010), ACM, 77–86.
34. Schmidt, R.A., Lee, T.D. *Motor Control & Learning: A Behavioral Emphasis*, 5th edn, Human Kinetics, 2011.
35. Schneider, W., Shiffrin, R.M. Controlled and automatic human information processing: I. Detection, search, and attention. *Psychol. Rev. 84*, 1 (1977), 1–66.
36. Shannon, C.E. A mathematical theory of communication. *Bell System Technical J. 27* (1948), 379–423, 623–656.
37. ShapeWriterInc. Press Release: ShapeWriter Publishes Award Winning Software On Multiple Mobile and PC Platforms, 2008 [retrieved from www.reuters.com on 14 May 2012].
38. Shiffrin, R.M., Schneider, W. Controlled and automatic human information processing: II. Perceptual learning, automatic attending and a general theory. *Psychol. Rev. 84*, 2 (1977), 127–190.
39. Tulving, E. Introduction to memory. In *The New Cognitive Neurosciences*, M.S. Gazzaniga, ed. MIT Press, Cambridge, MA, 2000, 727–732.
40. Venolia, D., Neiberg, F. T-cube: a fast, self-disclosing pen-based alphabet. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems* (1994), 265–270.
41. Willingham, D.B. A neuropsychological theory of motor skill learning. *Psychol. Rev. 105* (1998), 558–584.
42. Wulf, G., Shea, C.H. Principles derived from the study of simple skills do not generalize to complex skill learning. *Psychonomics Bull. Rev. 9*, 2 (2003), 185–211.
43. Yaeger, L., Webb, B., Lyon, R. Combining neural networks and context-driven search for on-line, printed handwriting recognition in the Newton. *AI Magazine 19*, 1 (1998).
44. Yamada, H. A historical study of typewriters and typing methods: from the position of planning Japanese parallels. *J. Inform. Process. 2*, 4 (1980) 175–202.
45. Zhai, S. On the ease and efficiency of human-computer interfaces. In *Proceedings of ACM ETRA 2008: ACM Eye Tracking Research & Applications Symposium* (2008), 9–10.
46. Zhai, S., Kristensson, P.O. Shorthand writing on stylus keyboard. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems, CHI Letters 5(1)* (Fort Lauderdale, FL, 2003), ACM, 97–104.
47. Zhai, S., Kristensson, P.O. Introduction to shape writing, IBM Research Report RJ10393 (also as a Chapter 7 of *Text Entry Systems: Mobility, Accessibility, Universality* MacKenzie, I. S., Tanaka-Ishii, K., eds Morgan Kaufmann Publishers, 2006, 139–158).
48. Zhai, S., Hunter, M., Smith., B. A. The Metropolis keyboard – an exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the 13th annual ACM symposium on User Interface Software and Technology (UIST)* (2000), 119–128.
49. Zhai, S., Hunter, M., Smith, B.A. Performance optimization of virtual keyboards. *Hum. Comput. Interact. 17*(2–3) (2002), 89–129.
50. Zhai, S., Kristensson, P.O., Smith, B.A. In Search of effective text input interfaces for off the desktop computing. *Interacting with Computers 17*, 3 (2005), 229–250.
51. Zhai, S., Kristensson, P.O., Gong, P., Greiner, M., Peng, S.A., Liu, L.M., Dunnigan, A. Shapewriter on the iphone: from the laboratory to the real world. In *Extended Abstracts of ACM CHI Conference on Human Factors in Computing Systems (Design Practice)* (2009), ACM, 2667–2670.

Shumin Zhai is currently a senior staff research scientist at Google and Editor-in-Chief of *ACM Transactions on Computer-Human Interaction.*

Per Ola Kristensson is a lecturer in Human Computer Interaction and an EPSRC Research Fellow at the University of St Andrews.