Shorthand Writing on Stylus Keyboard

Shumin Zhai

IBM Almaden Research Center
650 Harry Road, San Jose, CA, USA
zhai@us.ibm.com

Per-Ola Kristensson

Department of Computer & Information Science
Linköping University, S-581 83, Sweden
perkr@ida.liu.se

ABSTRACT

We propose a method for computer-based speed writing, *SHARK* (shorthand aided rapid keyboarding), which augments stylus keyboarding with shorthand gesturing. *SHARK* defines a shorthand symbol for each word according to its movement pattern on an optimized stylus keyboard. The key principles for the *SHARK* design include high efficiency stemmed from layout optimization, duality of gesturing and stylus tapping, scale and location independent writing, Zipf's law, and skill transfer from tapping to shorthand writing due to pattern consistency. We developed a *SHARK* system based on a classic handwriting recognition algorithm. A user study demonstrated the feasibility of the *SHARK* method.

Keywords

Text input, shorthand, gesture, stylus keyboard, virtual keyboard, pervasive, mobile, handheld devices, human memory, learning, skill acquisition.

INTRODUCTION

Text input - ranging from writing emails, filling forms, typing commands, taking notes, to authoring articles and coding programs - constitutes one of the most frequent computer user tasks. The QWERTY keyboard, for various reasons, has been accepted as the standard tool to accomplish this task for desktop computing (see [25] for a brief review). The emergence of handheld and other forms of pervasive or mobile computing devices, however, calls for alternative solutions. Consequently text input has been revived as a critical HCI research topic in recent years. There have been various methods proposed, developed or studied (See [5, 14] for surveys). The two classes of solutions that have attracted most attention are handwriting and stylus based virtual keyboarding.

Handwriting is a rather "natural" and fluid mode of text entry, thanks to users' prior experience from writing on paper. Various handwriting recognition systems, such as *Graffiti* in the *Palm Pilot* and *Jot* in *Windows CE*, have been used in commercial products. The fundamental weakness of handwriting as a text entry method, however, is its limited speed, typically around 15 wpm [6]. Such a speed is good enough for entering names and phone numbers on a PDA, but too limited for writing longer text.

Virtual keyboards, tapped serially with a stylus, are also available in commercial products, typically in the familiar QWERTY layout. To improve movement efficiency, stylus keyboard layout can be optimized either by trial and error [15] or algorithmically [25]. Depending on the degree of optimization, the expert text entry speed with stylus keyboard can be more than 45 wpm on layouts such as *ATOMIK* (Fig. 1, see [26] also [15]). There are also weaknesses to stylus keyboarding. The simple tapping movement may feel tedious to repeat for prolonged use. Stylus keyboarding also requires intense visual attention, virtually at every key tap, which prevents the user from focusing attention on text output.

Our current work began with two observations in stylus keyboard research. First, it has been noted that some words or fragments of words are connected in stylus keyboards. With these words, such as *the* in Fig 1, it is possible to stroke through the keys rather than tapping on them individually, hence introducing a form of more fluid movement closer to drawing [25]. Second, for a well-practiced word, users tend to remember its *pattern*, the trajectory of stylus movement from key to key as an integrated chunk, rather than individual key taps [25].

These two observations led us to imagine letting the user directly write patterns as a basic mode of entering words on a stylus keyboard. Each pattern of a word is formed by the trajectory from the first to the last letter of the word on the keyboard. Fig. 2 shows a few examples of such patterns defined by the *ATOMIK* keyboard in Fig. 1.

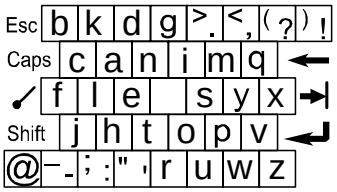


Fig. 1 The *ATOMIK* Keyboard (adapted from [25] by permission)

CHI 2003, April 5–10, 2003, Ft. Lauderdale, Florida, USA.

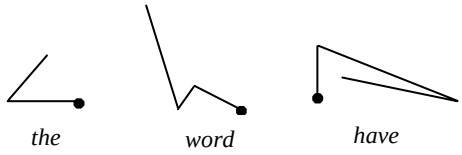


Fig. 2 Word patterns defined by ATOMIK Keyboard (a dot indicates the starting end)

MOTIVATING PRINCIPLES

Scale and location independency

Is there any advantage to pattern gesturing over individual letter tapping? The answer depends on how the gesture is produced and recognized. To precisely cross all letters defining a word would require just as much, if not more, visual attention as serially tapping all the letters. Furthermore, the time to perform such visually guided steering cannot be expected to be any shorter than tapping[1]. Therefore, for gesturing to be effective, patterns must be recognized at least partially independent of scale and location. As long as the user produces a pattern that matches the shape of the prototype of a word, the system should recognize and type the corresponding word for the user. If so, the users could produce these patterns with much less visual attention and presumably greater ease and comfort. Scale and location independency is the first principle in our current work.

Efficiency

The second principle of the current work lies in 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. In other words, this can be a form of *shorthand* writing.

In theory such form of shorthand can be defined on any keyboard layout. However if we define it on the familiar QWERTY layout, for example, it would involve frequent left-right zigzag strokes, because the commonly used consecutive keys are deliberately arranged on the opposite sides of QWERTY (See [24] for a long and [25] for a short review on this issue). We choose the *ATOMIK* layout for the current work. See Fig. 2 and Table 1 for examples of shorthand symbols defined on *ATOMIK* (Fig 1). *ATOMIK* (Alphabetically Tuned and Optimized Mobile Interface Keyboard) was optimized by a Metropolis algorithm in which the keyboard was treated as a "molecule" and each key as an "atom". The atomic interactions among all of the keys drove the movement efficiency - defined by the summation of all movement times between every pair of keys, weighted by the statistical frequency of the corresponding pair of letters - towards the minimum. This means that in any given scale, the average word gesturing length on *ATOMIK* is also minimized. *ATOMIK* is also alphabetically tuned, causing a general tendency that letters from A to Z run from the upper left corner to the lower right corner of the keyboard. This gives helps users look for letters that are not yet memorized. Furthermore, it maximizes, without sacrificing the first two features, the letter connectivity of the most common words [25].

Duality

Traditional shorthand writing systems, such as Pitman's, takes significant time and effort to master. With the exception of touch-typing on physical keyboards, which takes hundreds of hours of practice to be proficient [7], users are usually reluctant to invest time in learning a human computer interaction skill. A shorthand system defined on a stylus keyboard, however, does not have to contain a complete or even a large set of words, because one can use both tapping and shorthand gesturing. For familiar words whose patterns are well remembered, the user can write their shorthand. For the less familiar, one can use stylus tapping. Both modes of typing are conduced on the same input surface and the system distinguishes tapping from stroking and gives output accordingly. We call this combination method *SHARK* - shorthand aided rapid (stylus) keyboarding.

Zipf's law effect

Fourth, word frequency in a language tends to follow Zipf's law[2], with a highly skewed distribution (Fig. 3). For example, the 100 most common individual words make up 46% of the entire British National Corpus (http://www.bnc.org). The word *the* alone constitutes over 6% of the BNC. This means that a relatively small set of shorthand gestures can cover a large percentage of text input. The benefit of using shorthand for a small set of common words is disproportionably large.

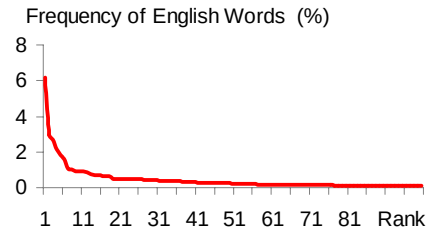


Fig. 3 Word frequency distribution in BNC

Transition from tapping to gesturing

Finally, a user's repertoire of shorthand symbols can be gradually expanded with practice. Gesturing and tapping a word with *SHARK* share a *common* movement pattern, which may facilitate skill transfer between the two modes.

---

[1] It is possible to use the law of steering [1] and Fitts' law of pointing to quantitatively analyze the difference between tapping and tracing.

2 Zipf's law models the observation that frequency of occurrence of event f, as a function of its rank i, is a power-law function $f \sim 1/i^a$ with the exponent *a* close to unity.

For a novice user, visually guided tapping is easier. When a word is tapped enough times, the user may switch to the more fluid "expert" mode of shorthand gesturing[3]. If a shorthand gesture is forgotten, one can fall back to taping, which reinforces the pattern and pushes the user back to expert mode. This gradual and smooth transition from novice to expert behavior can be very similar to the process of mastering *marking menus* [10, 11], and constitutes the fifth principle of *SHARK*.

We have outlined five motivations and rationales of using *SHARK* as a new approach to text entry. In what follows we first briefly point out previous work related to *SHARK*. We then describe the design and implementation of a *SHARK* system to demonstrate that *SHARK* is practically feasible based on current pattern recognition technology. We then switch to the human side by presenting an experiment that tested users' ability to learn and recall *SHARK* gestures. We will conclude the paper with a discussion of future work.

RELATED WORK

The need of entering text on mobile devices 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 continuous gesture-based text input. For stylus keyboard optimization, which is an important foundation of *SHARK*, readers are referred to [25] and also [15]. For other input methods, see [5, 14, 25].

The idea of optimizing gesture for speed is embodied in the *Unistrokes* alphabet designed by Goldberg and Richardson [9]. 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 Roman alphabet. The fundamental limitation of *Unistrokes*, however, is the nature of writing one letter at a time.

Quikwriting designed by Perlin [17] 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*cula*r In*put), designed by Mankoff and Abowd [16], 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 bases of *SHARK* such as open-loop *pattern* production rather than crossing individual keys, location and scale independency, combination with and transition from stylus tapping, are not in *Cirrin*.

The idea of bridging novice and expert modes of use by common movement pattern in *SHARK* was inspired by Kurtenbach, Buxton and colleagues' work on *marking menu* [10, 11]. Instead of having pull-down menus and shortcut keys - two distinct modes of operation for novice and expert respectively, a marking menu uses the same directional gesture on a pie menu for both types of users. Particularly innovative in marking menu is the use of delayed feedback. For a novice whose action is slow and needs visual guidance, marking menu "reveals" itself by displaying the menu layout. For an expert whose action is fast, their system does not display visual guidance at all so the user's actions become open loop marks. The same movement gesture affords smooth transition from novice to expert.

A self-revealing menu approach has also been explored in text entry, in the *T-Cube* method designed by Venolia and Neiberg [22]. *T-Cube* defines an alphabet set by cascaded pie menus. A novice enters characters by following the visual guidance of menus, while an expert user could enter the individual characters by making menu gestures without visual display. A weakness of the T-Cube is that it works at alphabet level and hence could not be very fast, as was shown by the authors [22].

Dasher, by Ward, Blackwell and MacKay [23], is another approach of continuous gesture input. *Dasher* dynamically arranges letters in multiple columns, with likely target letters closer to user's cursor based on the proceeding context. A letter is selected when it passes through the cursor, whose movement is hence minimized. This minimization, however, is at the expense of visual attention. Because the letter arrangement constantly changes, *Dasher* demands user's visual attention to dynamically react to the changing layout.

SHARK GESTURE RECOGNITION

This section deals with *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 [21] or [4] for surveys of the common techniques). We have developed a *SHARK* recognition system based on the classic *elastic matching* algorithm [20] which computes the minimum distance

[3] The *SHARK* system may count the frequency of each word used. When beyond certain threshold, a *SHARK* shorthand pattern can be optionally displayed on the keyboard (by connecting the letters with dotted line, for example) suggesting that the user switch to shorthand for that word.

between two sets of points by dynamic programming. One set of points is from the shape that a user produces (a *unknown* shape). The other is from a *prototype* – the ideal shape defined by the letter key positions of a word. After preprocessing, filtering and normalization in scale, the distance between the unknown shape and the prototypes are computed by elastic matching. The word corresponding to the prototype that has the shortest distance to the sample is returned as the recognized word. Based on these operations, we have written a *SHARK* notepad application with dual modes of input - shorthand gesturing and stylus tapping. The algorithm and techniques used in our system can all be found in the previous literature [2, 20, 21].

It is beyond the space limit and the scope of this paper to explore the optimal recognition technology for *SHARK*. However, it is necessary to discuss a special aspect of *SHARK* recognition – ambiguity handling. The shape of a *SHARK* gesture is not always unique and hence creates ambiguity. Although in low percentage, there are words that share the same *SHARK* gesture, particularly for some short words. For example, the word *can*, *an*, and *to* are completely identical if we do not consider scale and location. The same is true to *do* and *no* (see Fig. 1). We have designed two approaches to resolve ambiguities. One is the use of transient pie menus. As shown in Fig. 4, when the recognition system found more than one match to a sample, it pops up a pie menu with all the candidates (usually only two or three) in a consistent order. A user inexperienced with this particular ambiguous word would look at the menu and make the second stroke in the *direction* of the candidate intended, independent of location. With experience, the user does not have to look at the menu, because the candidates are presented in a consistent location of the pie menu and the selection of choice depends on direction only. An experienced user may simply remember the second stroke as part of the shorthand for that word. As shown in Fig. 4, for example, a right horizontal stroke followed by a stroke to the upper-left direction will always be *can*. Similarly left and down is always *to* and so on.

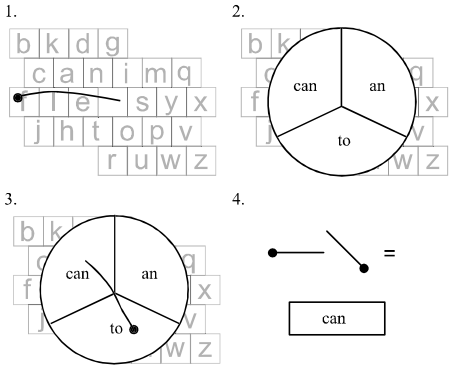


Fig. 4. Resolving ambiguity by a transient pie menu

One disadvantage to this approach is the loss of efficiency due to the added second stroke. The second approach we designed maintains efficiency but sacrifices the location independence principle for the ambiguous words. The system checks the start position, or the geometric center of the sample, relative to the letters defining the multiple ambiguous prototypes, to determine which word the user intends to write. For example, if the user makes a right horizontal stroke closer to the upper left region of the keyboard where *c-a-n* are (Fig. 1), the system gives the user *can*; if further to the right, *an*; if closer to the lower middle region, *to*, and so on.

The weakness of the partial location dependency method is that it requires more visual attention when writing the words with ambiguity in order to make sure the location is closer to the intended word on the keyboard, although this is still less stringent than tapping. Further study is needed to find the best ambiguity resolution method for *SHARK*.

Note that since the recognition is *online* (real time), the system takes the direction of the *SHARK* gestures from pen down to pen up into account. Some words that appear ambiguous offline are not in fact ambiguous. The word *in*, for example, is also a horizontal stroke but it will never be confused with the word *can*.

A USER STUDY

Issues to be explored

The *SHARK* method raises many human performance questions. To completely address these questions requires extensive long-term research. At this early stage, we focus on the most basic question - can users learn, remember or discriminate (see [13] for a recent study on gesture similarity), and produce shorthand gestures defined on stylus keyboard at all? Can they learn a useful number of *SHARK* gestures in a relatively short period?

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 [8]. A literate Chinese person must learn two to five thousands of unique characters. In sum, there are ample evidence that people can master a large number of symbols.

However, even Chinese characters are made of common radicals. It is likely that in people's memory Chinese characters are reduced to these radicals and the two dimensional relationships among the radicals. *SHARK* gestures (See Table 1 for examples), on the other hand, are formed in a novel approach that does not break a continuous shape down to elements, although experienced users may transfer common traces between some words, such as *-ing* and *-tion*. Clearly, to understand people's ability to learn *SHARK*, an empirical study is necessary.

How well people can learn new skills partly depends on the methods they learn and practice the new skill with. The secondary goal of this user study is to design an effective method to help users learn *SHARK* gestures.

Learning method

The literature on skill acquisition (e.g. [18]) and human memory (e.g. [3]) presents a variety of theories, models and insights on how people learn and remember skills. A compelling method from that literature that may be relevant and useful in helping people to learn *SHARK* is practicing with expanding rehearsal interval (ERI) [12]. 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 [19].

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. Recently it has been shown that such a method could be effective to learning stylus keyboarding [26], although the rate of expansion and the unit of training in stylus keyboarding needed further investigation. Different from the original ERI method which schedules rehearsal at increasing but preprogrammed intervals, we decided to make ERI adaptive based on the learner's performance. The interval of rehearsal for a particular *SHARK* gesture expands only if the learner could recall the gesture correctly. Otherwise, the interval stays the same. We could also make the interval shrink if the learner repeatedly fails to recall a gesture, but our experience shows that this is not necessary in practice and may lead to frustration.

In each cycle of rehearsal of a particular word by our ERI method, the participant was asked to write the *SHARK* shorthand for that word on the stylus keyboard area that did not show the layout (Fig. 5). This forced the user to actively retrieve *SHARK* movement patterns from memory. Research has shown that active retrieval is a key to memory retention [19]. The word that matched the user's *SHARK* gesture was then displayed in the *last entry* area (Fig. 6). If correct, the word would be rescheduled to appear at an interval twice the current value. The user may go on to the next word or practice the current word a few more times with the choice of seeing the keyboard layout by clicking *show keyboard*, before moving on.

If the participant could not recall the *SHARK* gesture of a practiced word or failed to write it correctly, the rehearsal interval will keep its current value. The user could choose to learn (for a new word) or relearn (for a forgotten word) the *SHARK* gesture by displaying the *ATOMIK* keyboard, together with the gesture prototype drawn in dotted lines connecting the letters in the word (Fig. 6). The user could write the *SHARK* gesture anywhere on the keyboard in any scale, and practice the gesture as many times as wanted to explore the tolerance of acceptable shapes for the target word.

Fig. 5. Screenshot of the experimental set-up: the user was asked to write a word ("inside") by *SHARK* without visual reference

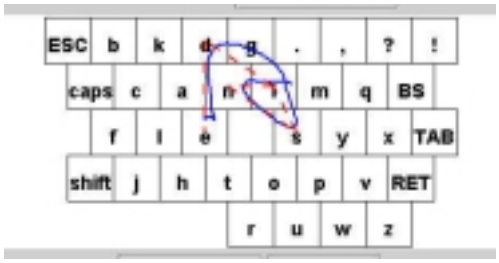


Fig. 6. The user may learn a SHARK gesture by displaying the *ATOMIK* keyboard layout

The learning program maintained two lists of words: a *word list* containing the maximum number of words to be learned in the study and a *rehearse list* keeping all words being actively rehearsed at various intervals. Each word in the *rehearse list* had its own timer, counting down from its current rehearsal interval value. The algorithm that managed the ERI scheduling worked as follows:

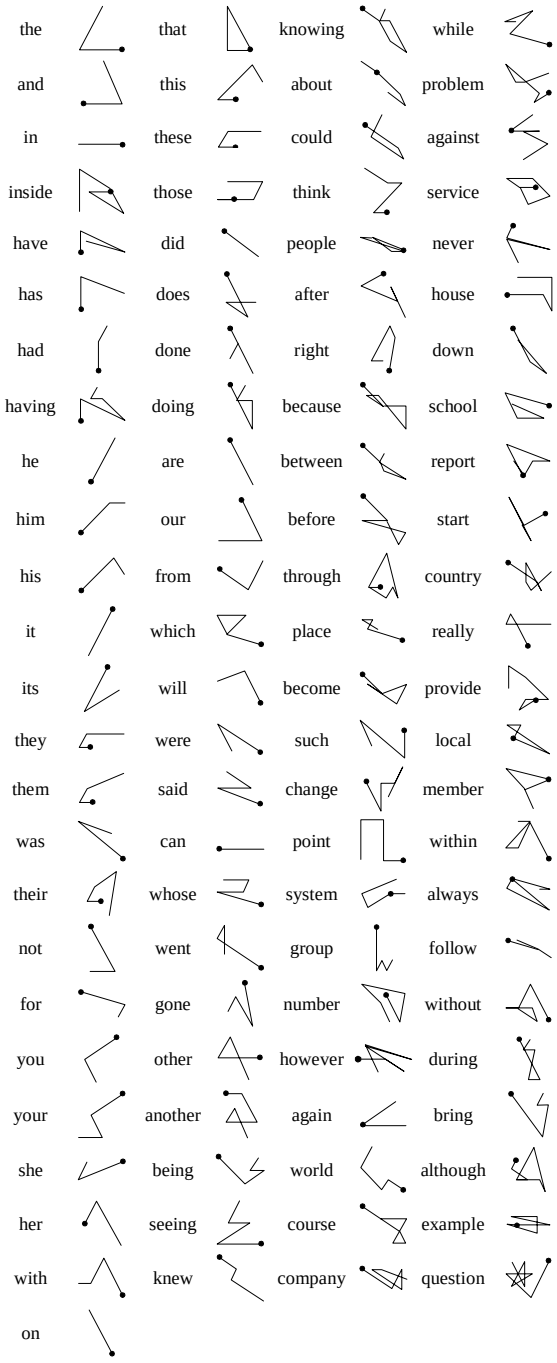
1. If the *rehearse list* is empty, pick a new word from the *word list* and initialize it with a rehearse interval of 30 sec and put it in the *rehearse list*.
2. Pick the word from the *rehearse list* that has the earliest rehearse time according to the value left in its timer.
3. If the timer is below 30 sec present the word to the user. Otherwise pick a new word from the *word list* and present it to the user.
4. If the user writes the correct *SHARK* gesture of the target word, return it to the *rehearse list* with a doubled rehearsal interval, else return it with an unaltered rehearsal interval, Go to 2.

Experimental set up and procedure

The experiment was carried out on a PC attached with a Wacom tablet model ET-0405-U as the pen input device.

Six paid volunteers, two male four female, between 20 and 30 years old, participated in a five-session experiment. None of them had any prior experience with *SHARK* or the

Table 1. Words in the study and their *SHARK* shorthand[4]



[4] Due to space constraint these *SHARK* symbols can only be listed in small size. The static shapes do not reveal the dynamic stroke directions, which make them more distinct. A solid dot signifies the start point of a *SHARK* symbol. Note also symbols listed in the table are *SHARK* prototypes. Actual user's sample shapes always deviate from the prototypes, with rounded corners for example.

ATOMIK keyboard. Session 1 was 40 minutes practice with ERI only. Session 2, 3, and 4 consisted of two parts. Part one was a test session of words a participant had practiced in previous sessions, lasting from 6 to 20 minutes. There was a period of minimum one day (night), and sometimes a full weekend between a test and the previous practice session. Part two was 40 minutes practice with ERI. Participants were asked to take a 5-minute break in the middle of the practice session, and were suggested to take short breaks anytime they liked to. Session 5, the last session, was a final test session only.

In test sessions, words were presented in a random order. The participants were given at most two chances to recall and write the *SHARK* gesture correctly.

In the experiment a total of 100 words were in the *word list*. We focused on the baseline of *SHARK* shorthand memory in this experiment and made sure no ambiguity existed in the list. 50 of the 100 words used were selected from the top 100 most common words in the British National Corpus. The next 50 words were selected from the top 100 to top 300 most common words. These words and their corresponding *SHARK* shorthand prototypes are listed in Table 1.

Results and discussion

The results of the user study shows that all of the participants could learn to write correctly recognizable *SHARK* gesture for any word presented to them, if practiced enough times (typically 7 to 15 ERI cycles). Fig. 7 shows the ERI traces of a few sample words by one participant. Note that all these traces ran across different days (sessions). Some participants felt that new words disappeared too quickly, suggesting the expansion rate (factor of 2) too aggressive, whereas others found the frequency of practicing adequate. Some participants could keep up with the ERI expansion and correctly write the *SHARK* gesture every time (See *other* in Fig. 7 for example). In sum, the ERI method appeared effective, but it could certainly be further tuned. For example, it is possible to make the expansion rate adaptive to individuals based on their performance.

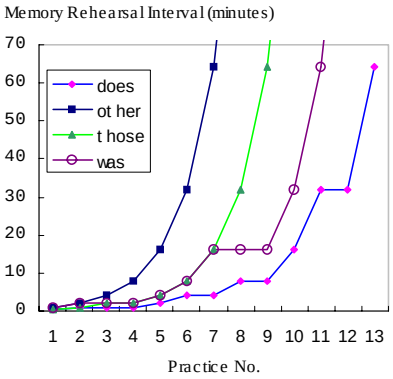


Fig. 7. Sample ERI traces

As shown in Fig. 8 and 9, participants were able to correctly write more words in each learning session, 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, or slightly higher toward later sessions (Fig. 9).

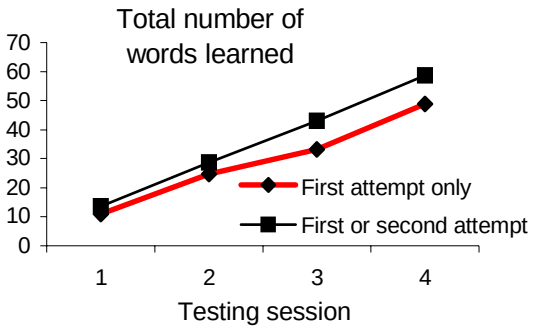


Fig. 8. Total number of words correctly written in test sessions, averaged across participants

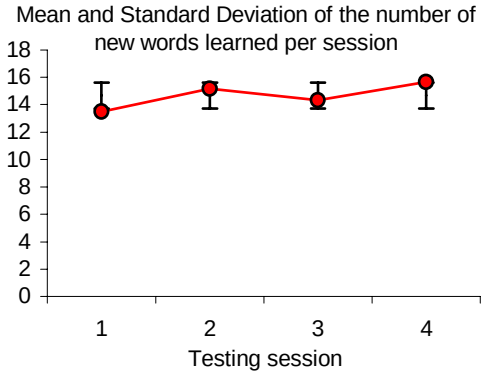


Fig. 9. More words learned in each session

During the study participants were encouraged to write down comments on *SHARK* and the learning method used each day. Most of them found the *SHARK* method exciting, 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). User's initial reaction is often the most critical to the adoption of an interface style. None of the participants found the method tedious throughout the study. In the final questionnaire, participants rated 0.6 on the scale of (–3 frustrating, +3 satisfying); 1.83 on (–3 dull, +3 stimulating); 0.6 on (–3 difficult, +3 easy) to use; 0.6 on (–3 difficult, +3 easy) to learn. To the question "If such a method is made available for practical use, would you learn it?", their answer averaged 1.6 on the scale of (–3 definitely no, +3 definitely yes). When asked if they would use it in a list of situations, 0/6 participant choose "not at all", 2/6 answered "Yes, when a physical keyboard is not available" and 4/6 "Yes, to replace keyboard typing sometimes", and 0/6 "Yes, to replace keyboard typing all the time". It is particularly encouraging that the majority could imagine using *SHARK* even to "replace keyboard typing sometimes". Since there is no direct comparison to methods with similar amount of experience and the study was short and necessarily artificial, these subjective data can only be taken as a reference, not conclusions.

Some of the anecdotal comments were also informative. Two participants attempted to memorize the *ATOMIK* layout and tried to guess the *SHARK* gesture based on their memory of the layout when asked to write a word the first time. The participants were usually displeased when they wrote a word correctly in their mind, but the system was "unfair" and did not recognize it. It suggests that our recognizer based on a standard elastic matching algorithm, which emphasizes the degree of *proportion*, did not necessarily accurately reflect user's cognitive model of the *SHARK* gestures, which is probably more *topological*. Two participants mentioned "it was easier to remember complex shapes than simple ones", contrary to our concern that long words will be difficult to handle with *SHARK*. Participants occasionally drew mirror image of the correct *SHARK* shape, although practice eventually overcame such types of error.

50 to 60 shorthand gestures learned in four sessions can already be very useful in text input, given the Zipf's law effect. The study showed no sign of slowing down in user's ability to learn more *SHARK* shorthand symbols. How many such symbols can eventually be learned, if limited at all, is unknown. Because these symbols are constructed differently from other large character sets such as Chinese, it is difficult to find precedence of human capability in this regard. In general, human long-term memory capacity is only limited by acquisition speed.

Given the prospect of creating a whole new system of efficient writing, a great deal more research is needed in the future to understand, validate/invalidate, improve, change, and innovate many aspects of it. We have only done the most basic work to show the feasibility of *SHARK*. Other compelling research issues include its expert speed limit, long term study of users' learning and usage of *SHARK*, particularly in real use of text production; the usability of various ambiguity resolution methods; theoretical quantification, modeling and empirical measurement of *SHARK* gestures-based input in terms of speed, accuracy, and capacity, from both human memory and motor control, as well as machine recognition point of view; and improvement of learning method, preferably integrated in the *SHARK* system and combined with the use of it. In terms of developing more advanced recognition algorithm that matches human perception of shapes more closely in the future, much can be done in making the recognition more sophisticated, with situation specific "work-around" and probably borrowing techniques from modern complex speech recognition systems. High-level constraints, such as the context of proceeding words, can also potentially be used to make *SHARK* recognition more tolerant to "sloppy" writing.

CONCLUSIONS

Speed writing is an old research topic with renewed interest due to the need of using mobile computing devices. To write a significant amount of text, natural writing is too slow and stylus keyboarding is tedious and visually demanding. We have proposed a new approach that combines the two at a shorthand level - *SHARK*. *SHARK* defines a shorthand symbol for each word according to its movement pattern on a stylus keyboard layout. The key rationale for designing *SHARK* rests on five principles: efficiency, location and scale independency, duality of tapping and gesturing, Zipf's law effect, and transition from tapping to gesturing. We have developed a *SHARK* recognizer based on an elastic matching recognition algorithm. A user study indicated the feasibility of *SHARK*. Participants could learn *SHARK* shorthand symbols, at a average rate of 15 words per (45 to 50 minutes) session, by practicing with the ERI paradigm. Most participants found *SHARK* shorthand fun to use, and considered even to use it to replace physical typing sometimes. It was evident that people could learn a useful number of *SHARK* shorthand symbols in a rather small amount of time. To understand all issues involved and the full potential *SHARK*, possibly even beyond mobile computing as a form of speed writing, requires a great deal more research in the future.

ACKNOWLEDGMENTS

Part of this work was conducted when Shumin Zhai was a guest professor at Linköping University. We thank Santa Anna IT Research Institute AB and Sture Hägglund for support. We also thank Jingtao Wang, Paul Maglio, Maria Holmqvist and Pernilla Qvarfordt for their assistance.

REFERENCES

1. Accot, J. and S. Zhai. Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. *Proc. CHI'97*. p. 295-302.
2. Arakawa, H., Odaka, K., Masuda, I. On-line recognition of handwritten characters – Alphanumerics, Hiragana, Katakana, Kanji. *Proc. 4th International Joint Conference on Pattern Recognition*, Nov 1978, pp. 810 – 812.
3. Baddeley, *Human Memory - Theory and Practice*. Revised Edition. 1998, Boston: Allyn and Bacon.
4. Beigi, H.S.M. An overview of handwriting recognition. *Proc. The 1st Annual Conference on Technological Advancements in Developing countries*. 1993. Columbia University. p. 30-46.
5. Buxton, W. *Human Input to Computer Systems: Theories, Techniques and Technology* (book manuscript). 1994/2002: available at http://www.billbuxton.com/inputManuscript.html
6. Card, S.K., T.P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. 1983, Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.
7. Cooper, W.E., ed. *Cognitive aspects of skilled typewriting*. . 1983, Springer-Verlag: New York.
8. Gardiner, A., *Egyptian Grammar: Being an Introduction to the Study of Hieroglyphs (3rd edition)*. 1978: Aris & Phillips.
9. Goldberg, D. and C. Richardson. Touching-typing with a stylus. *Proc. INTERCHI*. 1993. Amsterdam, p. 80-87.
10. Kurtenbach, G. and W. Buxton. User Learning and Performance with Marking Menus. *Proc. CHI*. 1994. p. 258-264.
11. Kurtenbach, G., A. Sellen, and W. Buxton, An empirical evaluation of some articulatory and cognitive aspects of "marking menus". *Human Computer Interaction*, 1993. **8**(1): p. 1-23.
12. Landauer, T.K. and R.A. Bjork, *Optimum rehearsal patterns and name learning*, in *Practical Aspects of Memory*, 1978, Academic Press: London. p. 625-632.
13. Long, A.C., J.A. Landay, L.A. Rowe, and J. Michiels. Visual similarity of pen gestures. *Proc. CHI2000*.
14. MacKenzie, I.S. and R.W. Soukoreff, Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 2002. **17**(2&3), p.147-198.
15. MacKenzie, I.S. and S.X. Zhang. The design and evaluation of a high-performance soft keyboard. *Proc. CHI*. 1999. p. 25-31.
16. Mankoff, J. and G.D. Abowd. Cirrin: a word-level unistroke keyboard for pen input. *Proc. ACM UIST, Tech. Note*. 1998. p. 213 - 214.
17. Perlin, K. Quikwriting: Continuous Stylus-based Text Entry. *Proc. ACM UIST, Tech. Note*. 1998: p. 215 - 216.
18. Proctor, R.W. and A. Dutta, *Skill acquisition and human performance*. 1995, Thousand Oaks, CA: Sage.
19. Schmidt, R.A. and R.A. Bjork, The conceptualizations of practice: common principles in three paradigms suggest new concepts for training. *Psychological Science*, 1992. **3**(4): p. 207-217.
20. Tappert, C.C., Cursive Script Recognition by Elastic Matching. *IBM Journal of Research & Development*, 1982. **26**(6): p. 756-771.
21. Tappert, C.C., C.Y. Suen, and T. Wakahara, The State of the Art in On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990. **12**(8).
22. Venolia, D. and F. Neiberg. T-Cube: A Fast, Self-Disclosing Pen-Based Alphabet. *Proc. CHI*. 1994. p. 265 - 270.
23. Ward, D., A. Blackwell, and D. MacKay. Dasher - A data entry interface using continuous gesture and language models. *Proc. ACM UIST*. 2000. p. 129-136.
24. Yamada, H., A historical study of typewriters and typing methods: from the position of planning Japanese parallels. *Journal of Information Processing*, 1980. **2**(4): p. 175-202.
25. Zhai, S., B.A. Smith, and M. Hunter, Performance Optimization of Virtual Keyboards. *Human-Computer Interaction*, 2002. **17**(2&3), p.229-270.
26. Zhai, S., A. Sue, and J. Accot. Movement Model, Hits Distribution and Learning in Virtual Keyboarding. *Proc. CHI*2002. p.17-24.