RJ 10323 (A0408-008) August 10, 2004
Computer Science

IBM Research Report

Elastic Stylus Keyboards – Automatic Error Correction in Stylus Typing by Pattern Matching

Per-Ola Kristensson*, Shumin Zhai
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

*and Department of Computer and Information Science
Linköpings Universitet
581 83, Linköping, Sweden

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Elastic Stylus Keyboards – Automatic Error Correction in Stylus Typing by Pattern Matching

Per-Ola Kristensson [§♦]
♦ Department of Computer and Information Science
Linköpings universitet, 581 83, Linköping, Sweden
perkr@ida.liu.se

Shumin Zhai [§]
§ IBM Almaden Research Center
650 Harry Road, San Jose, CA, USA
zhai@almaden.ibm.com

ABSTRACT
Due to the inherent speed-accuracy trade-off constraint in stylus typing, users tend to miss the targeted keys, resulting in erroneous words. We present a pattern matching based error tolerance method for stylus keyboards. We view the hit points on a stylus keyboard as a high resolution geometric pattern. This pattern can be matched against patterns formed by the letter key positions of legitimate words from a lexicon. By finding the closest matching word, erroneous taps can be corrected. We present the rationale, a geometrical pattern matching algorithm, and the key aspects of our implementation of an Elastic Stylus Keyboard (ESK).

KEYWORDS: Text input, intelligent user interfaces.

INTRODUCTION
The movement towards off the desktop computing, including hand-held devices and tablet computers, has stimulated a wave of invention, design, and research on text entry methods that do not require a physical keyboard. The stylus keyboard (SK), also known as graphical, virtual, soft, or on screen keyboard, is one class of them. Commercially SKs are in almost all hand-held and tablet computers. Research wise much work has been invested in this topic (See [2, 5, 7, 8, 14] for only a few examples).

One of the first and most prolonged research efforts has been on the optimization of the SK layout. Getschow and colleagues [2] made an early attempt of minimizing the statistical movement distance based on character digraph distributions. Lewis and colleagues [5, 6] were probably the first to use the well known Fitts' law and digraph frequencies as bases to model and design stylus keyboard performance. MacKenzie and Zhang [7] used such a model to manually design their OPTI layout. Zhai and colleagues [14] algorithmically optimized their ATOMIK layout based on the atomic interactions among all letters as defined by a Fitts-digraph energy function.

One weakness of the existing stylus keyboards is the verbatim process — the user has to tap letter by letter with complete accuracy. It is well known that natural languages have a great deal of regularity and redundancy, as Shannon observed in the process of introducing his information theory [10]. From an information theory point of view, tapping all letters with 100% accuracy is over-specifying the amount of information needed. In contrast, other text input methods, such as the T9 method commonly used in mobile phones, exploits language redundancy to resolve ambiguous key strokes. Another example is Dasher [12] which uses language regularities to dynamically align the most likely next letter near the selection cursor so the user can visually react and steer through the intended characters. A tempting use of language regularity is typing prediction [1]. Masui [8] has developed a SK that presents a list of the most likely words for the user to select based on previous inputs. However, an often overlooked aspect of word prediction with choices, or utilizing language regularity in general, is the cognitive and visual reaction time and effort needed to choose from the multiple candidates. The human visual motor reaction is about 200 ms minimum, and increases linearly with the amount of information in multiple choices as predicted by Hick's law [4]. For reference, typing at the speed of 60 words per minute (wpm) means inputting each character in only 200 ms.

Goodman and colleagues [3] proposed a method for SK error correction. Inspired by speech recognition technology, they calculated the probability of the intended key based on a character-level

language model (letter sequence statistics) and a stylus tapping model derived from observations of users' tapping behavior. A user study indicated that their model reduced the error rate as compared to verbatim tapping [3].

We propose a novel approach to exploit the language regularity in SKs using elastic geometric pattern matching. The idea is inspired by the SHARK shorthand method proposed in [13]. SHARK matches a user's continuous stylus trace pattern on the keyboard with the patterns of all words in a lexicon mapped to the center positions of the keys in a SK. The same pattern matching approach could also be applied to stylus typing so that a correct word will still be given even if some taps fall outside the intended letter keys, spurious taps occur, or some or even all taps miss the keys altogether, as long as the user's tapping pattern is sufficiently close to the pattern of the desired word. We call stylus keyboards enhanced with this approach Elastic Stylus Keyboards (ESK).

ELASTIC STYLUS KEYBOARDS

Fitts' law predicts that the mean time T to successfully hit a key of size W over distance D on a SK is $T = a + b\log_2((D+W)/W)$. This means that relative tapping accuracy imposes a certain speed ceiling. If the user attempts to go beyond the ceiling, the landing points of the stylus will tend to fall outside of a targeted key, resulting in a letter different from the intended one. In other words, the user will tend to break the W constraints. This adds to the user's frustration since it takes additional time and effort to correct these errors. Accuracy constraints are particularly problematic for users with certain motor control disabilities and for expert users who push their text entry speed limit. In the case of small mobile devices, the accuracy problem will be more acute.

Our goal is therefore to relax the accuracy requirement of precisely tapping on each letter, effectively widening the constraints of *W*. This is possible based on two observations. One is that not all letter combinations are legitimate words, as discussed in the introduction. We can therefore exploit these inherent constraints in legitimate words. The simplest implementation of this constraint is a lexicon which can easily be customized for each individual user. Other implementations of letter constraints may include a collection of n-grams, syllables, phonemes etc. The second observation is that the landing point of the stylus on a SK is a continuous variable recorded by the tablet or the touch screen, in contrast to a physical typewriter keyboard which can only record discrete key positions. A series of these landing points implicitly form a high resolution *pattern* (a sequence of points) on the SK. The center positions of all letter keys needed for inputting a word also form a pattern on the SK. The distance between these two patterns can be computed by various algorithms. By analyzing such distances to all words in a lexicon the most likely word can be found, even if one or more letters are mis-tapped, as long as the match passes a certain threshold. Otherwise the verbatim letter sequence can be returned.

For instance, in Figure 1 the user has tapped on the keys *r, j, n* and *w* (hit points indicated as solid circles) on a zoomed-in part of the QWERTY layout. Without any error correction "rjnw" would be returned as the typed word. However, when looking at the shape formed by r-j-n-w and comparing it with the shapes of all words in a lexicon on the SK, "the" is the closest match. A pattern matching system should be able to correct the input despite the fact that the user missed all the targeted keys and accidentally had one spurious stylus tap. We call such an accuracy relaxation approach to ESK "geometrical matching".

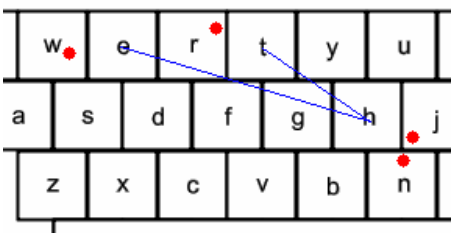

Figure 1. An example of error correction: the user tapped on r-j-n-w, but the intended t-h-e is returned

The advantages of an ESK in comparison to a statistical letter sequence approach [3] are numerous. ESK's matching effect works on the word level, and the words explicitly belong to an individual user's lexicon. New words can be added and removed in a customized dictionary; different languages such as Chinese *pinyin*, English and Swedish can be mixed without affecting the performance or behavior of the ESK. Depending on the size of the lexicon, the error tolerance of the ESK can be adjusted, either by the user or automatically by the system. Note that if the user aims at the correct letters in a word, the resulting shape will tend to approximate the ideal word pattern and be correctly matched. A user's input pattern can still be successfully matched to the intended word even if some of the hit points are far away from the correct keys, as long as the word patterns in the lexicon are sufficiently separated. Furthermore, the intuitive spatial interpretation of the matching method may enable expert users to consciously take advantage of the error correction scheme.

PATTERN RECOGNITION ALGORITHM AND SYSTEM

There are many pattern recognition methods that can be used for the current problem. We first used a simple linear matching algorithm, which computed the sum of all distances between the tapping points and their corresponding keys in a word in sequence (First tap to first letter in a word, second tap to second letter, and so on). User testing revealed various problems with this approach. For example, it could only find words with the same number of letters as the number of taps. If a tap is missed (deletion error) or an extra tap is inserted (insertion error), a correct word will never be found. We have also observed transposition errors in stylus typing in which the sequence of two letters was reversed. The linear matching approach was also poor at correcting this type of error.

Clearly a certain amount of *elasticity* is needed in the matching algorithm. We use a pattern matching algorithm that has the following properties: 1. Scalable to a lexicon that practically includes all words needed by a user. 2. Can match sequences of different lengths and cope with transposition errors. 3. No training of the classifier is necessary. The algorithm finds the minimum distance between two patterns by searching for the closest corresponding points between them through dynamic programming.

Similarity measure

Let X denote an *unknown* pattern consisting of an ordered sequence of n stylus hit points x_i on a SK, and let Y denote a *template* pattern consisting of m points y_i that are the centers of the corresponding keys for any word $w \in \{w : w \text{ is a word in the lexicon}\}$.

We define $D(X,Y)$ between X and Y as the minimum stretching cost needed to transform X into Y. Let $D(X,Y) = K(n,m)$ be the minimum stretching cost of matching $x_1 \ldots x_n$ against $y_1 \ldots y_m$. Wang and Pavlidis [11] showed that $K(i, j)$ for the subsequences $x_1 \ldots x_i$ against $y_1 \ldots y_j$ can be computed using a recurrent equation of the form:

$$K(i, j) = \min\begin{Bmatrix} K(i-1, j-1) + p(x_i, y_j), \\ K(i-1, j) + \tau \times p(x_i, y_j), \\ K(i, j-1) + \tau \times p(x_i, y_j) \end{Bmatrix} \qquad (1)$$

where $K(0,0) = 0$, $p(x_i, y_i)$ is the stretching cost x_i to y_j, and τ is an empirically determined parameter specifying the cost of either ignoring or inserting a single point. We currently set $\tau = 2.0$. To avoid extreme stretching of a single point we currently define $p(x, y)$ as

$$p(x, y) = \begin{cases} \|x - y\|_2 & \text{if } \|x - y\|_2 \le r \\ \infty & \text{otherwise} \end{cases} \qquad (2)$$

where r is the maximum distance a point may be stretched.

To be able to make fair comparisons between patterns of unequal length we compute a pseudo-

normalized minimum stretching cost: $D_N(X,Y) = D(X,Y)/(n+m)$.

It is well known that the computation of $K(n,m)$ for matching X against Y in equation (1) can be solved efficiently using dynamic programming in $O(nm)$ time [11]. The best matching word is the word whose pattern has the lowest normalized stretching cost D_N against the user's tapping pattern.

Indexing

To avoid an exhaustive search of a large ($\approx 57{,}000$ words) lexicon we implemented a simple indexing technique. Since equation (2) constrains point-to-point distances to be shorter than r, a reasonable constraint is to force the point-to-point distances between the first and last points in the patterns to also be shorter than r. If r is sufficiently conservative, e.g. 1.5 times the radius of an alphabetical key on the keyboard, it directly leads to a simple yet effective indexing method.

We construct an ordered *k*-ary tree data structure of depth 2 where *k* is the number of alphabetical keys on the keyboard layout. Each node at index i at depth m represents a circular cluster C_{im} where the ith key center is the cluster center, and r is the radius of the cluster. At index i, $1 \le i < k$, C_{i1} represents a *start* position cluster and C_{i2} represent an *end* position cluster. A pattern Y of length m is *indexed* by a pointer in a cluster C_{j2} on depth 2 iff $y_1 \in C_{i1}$, $y_{m-1} \in C_{j2}$ and C_{j2} is a child node of C_{i1}. Set membership here is used to denote that a point is contained in the circular cluster. When querying the index with an unknown pattern X we walk the tree in breadth-first order and collect the set of all patterns in the lexicon indexed to the same depth 2 clusters as X. This set is then searched exhaustively.

If r is too large or all words in the lexicon have patterns mapped to the same start and end point clusters, this procedure would still result in an exhaustive search. In practice the character frequencies are distributed unevenly but with enough spread for this indexing procedure to significantly reduce an exhaustive search. For our 57,000 words large lexicon the largest possible set that needs to be searched exhaustively is about 4,000 words large when $r = 1.5$ in keyboard key radius units.

Threshold

After pattern classification we obtain a subset consisting of the words in the lexicon with a similarity distance D_N to the user's tapping pattern below a set threshold T. These words are then returned to the system as a ranked list. The system outputs the word with the shortest D_N.

The threshold T can be a fixed, e.g. to the diameter of a key on the keyboard layout, or a more adaptive value, e.g. by looking at the distribution of the point-to-point distances. We currently set $T = 1.0$ in keyboard key radius units.

Lexicon

The lexicon used can be constructed with various methods. It can be a preloaded standard dictionary, or a list of words extracted from the user's previously written documents, including emails and articles, or words added by the user to the list, or a combination of all. We have tested our system with a lexicon containing about 57,000 words created for handwriting recognition applications [9], as well as with a custom lexicon extracted from a user's 7 years of emails sent and received (about 7,000 words).

Using indexing an ESK can handle large lexicons, however it is important that the lexicon is just large enough (but not larger than necessary) to include all words a particular user needs so the probability of unwanted corrections is minimized and the capacity of correct mapping for "sloppy" stylus typing is maximized.

Delimiter

In the process of developing the elastic stylus keyboard, we conducted various user studies, some formal and other informal. The user studies results were fed back to the next iteration of design and development. Delimitation between different words was one of the most critical issues found in our user studies.

We first used the most obvious choice of a delimiter — the space key tap. This was proven problematic, because the user was just as likely to mis-tap the space key as any other key, even if we doubled the height of the space key in a QWERTY stylus keyboard. If the stylus fell outside of the space key while aiming for it, two words would not be delimited and an unintended word that was close to the combination of the two would most likely be given. Conversely, if the user hit the space key while aiming for some other key, an unintended delimitation might cause the system to map a half finished word to a different word.

There are a number of possible solutions. One is to use a special-purpose physical button. Its drawback is that the user either has to use two hands, with the non-dominant hand dedicated to entering a space (and hence delimitation), or switch between tapping with a stylus and pressing a button.

We eventually decided to use a pen gesture as a way of entering space (and delimitation). Although many other gestures could be used alternatively or concurrently, we decided to use a left-to-right pen stroke anywhere on or near the ESK as a word delimiter. This novel method proves to be quite effective — gesturing is distinctly different from a tapping action and yet easy to evoke between tapping actions.

DISCUSSION AND CONCLUSIONS

The current ESK system (written in Java) can search a lexicon containing about 57,000 words in real time with no perceptible delay on a Tablet PC with a 1 GHz processor. Informal testing shows that one could type faster with less effort on an ESK than on a regular stylus keyboard, due to the reduced need of correcting frequent errors and the more relaxed requirement for precision tapping. We also observed that it is important to choose an appropriate correction threshold and a suitable lexicon, so that neither too many errors are left uncorrected nor many input strings are changed to unintended words. In general we observed that users were more unforgiving for receiving unintended words than appreciating correct error corrections. It is hence necessary to be conservative.

An ESK works with any stylus layout, either QWERTY or an optimized layout. An ESK is a practical and easy-to-implement solution to improve the verbatim and error-prone input method of today's stylus keyboards; requiring little, if any, training from the end-user's part.

ACKNOWLEDGMENTS

We thank John Karidis for contributing to the initial invention of ESK. Part of this work was conducted when Per-Ola Kristensson interned at IBM in 2003.

REFERENCES

1. Darragh, J.J. and Witten, I.H. Adaptive Predictive Text Generation and the Reactive Keyboard. *Interacting with Computers, 3* (1), 1991, 27-50.
2. Getschow, C.O., Rosen, M.J. and Goodenough-Trepagnier, C., A systematic approach to design a minimum distance alphabetical keyboard. *Proc. RESNA (Rehabilitation Engineering Society of North America) 9th Annual Conference*, 1986, 396-398.
3. Goodman, J., Venolia, G., Steury, K. and Parker, C., Language modeling for soft keyboards. *Proc. AAAI 2002*, 2002, 419-424.
4. Keele, S.W. Motor Control. in Boff, K.R., Kaufman, L. and Thomas, J.P. eds. *Handbook of Perception and Human Performance*, John Wiley & Sons, New York, 1986, 30.31-30.60.
5. Lewis, J.R., Kennedy, P.J. and LaLomia, M.J. Improved typing-key layouts for single-finger or stylus input, 1992, IBM Technical Report TR 54.692.
6. Lewis, J.R., Potosnak, K.M. and Magyar, R.L. Keys and Keyboards. in Helander, M.G., Landauer, T.K. and Prabhu, P.V. *Handbook of human-computer interaction*, Elsevier Science, Amsterdam, 1997, 1285-1315.
7. MacKenzie, I.S. and Zhang, S.X., The design and evaluation of a high-performance soft keyboard. *Proc. CHI 1999*, ACM, 25-31.
8. Masui, T., An Efficient Text Input Method for Pen-based Computers. *Proc CHI 1998*, ACM, 328-335.
9. Petrelli, J.F. and Roy, A. Creating Word-Level Language Models for Handwriting Recognition. *International Journal on Document Analysis and Recognition, 5* (2&3), 2003, 126-137.
10. Shannon, C.E. A mathematical theory of communication. *The Bell System Technical Journal, 27*. 1948, 379-423, 623-656.

11. Wang, Y.P. and Pavlidis, T. Optimal Correspondence of String Subsequences. *IEEE Trans. Pattern Analysis and Machine Intelligence.*, *12* (11). 1990, 1080-1087.
12. Ward, D., Blackwell, A. and MacKay, D., Dasher - A data entry interface using continuous gesture and language models. *Proc. UIST 2000*, ACM, 129-136.
13. Zhai, S. and Kristensson, P.-O., Shorthand Writing on Stylus Keyboard. *Proc CHI 2003, CHI Letters 5(1)*, ACM, 97-104.
14. Zhai, S., Smith, B.A. and Hunter, M. Performance Optimization of Virtual Keyboards. *Human-Computer Interaction*, *17* (2,3). 2002, 89-129.