Relaxing Stylus Typing Precision by Geometric Pattern Matching

Per-Ola Kristensson [§♦]
[♦] Department of Computer and Information Science
Linköpings universitet
581 83, Linköping, Sweden
+46 (13) 284476
perkr@ida.liu.se

Shumin Zhai [§]
[§] IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
+1 (418) 927-1112
zhai@almaden.ibm.com

ABSTRACT

Fitts' law models the inherent speed-accuracy trade-off constraint in stylus typing. Users attempting to go beyond the Fitts' law speed ceiling will tend to land the stylus outside the targeted key, resulting in erroneous words and increasing users' frustration. We propose a geometric pattern matching technique to overcome this problem. Our solution can be used either as an enhanced spell checker or as a way to enable users to escape the Fitts' law constraint in stylus typing, potentially resulting in higher text entry speeds than what is currently theoretically modeled. 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 center positions of legitimate words in a lexicon. We present the development and evaluation of an "elastic" stylus keyboard capable of correcting words even if the user misses all the intended keys, as long as the user's tapping pattern is close enough to the intended word.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *input devices and strategies, interaction styles, theory and methods*; I.5.1 [**Pattern Recognition**]: Models – *geometric*; I.5.5 [**Pattern Recognition**] Implementation – *interactive systems*

General Terms

Algorithms, Design, Experimentation, Human Factors

Keywords

Text input, stylus keyboard, virtual keyboard, typing errors, typing correction, spell checker, Fitts' law

1. INTRODUCTION

The movement towards off-the-desktop computers, 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, also known as graphical, virtual, soft, or on-screen keyboard, is one class of them. Commercially, stylus keyboards are included in almost all hand-held and tablet computers. Academically, researchers have invested much time in this topic (see [4],[9],[11],[13],[22] for only a few examples).

One of the first and most prolonged research efforts has been on the optimization of the stylus keyboard layout. Getschow and colleagues [4] made an early attempt of minimizing the statistical movement distance based on character digraph distributions. Lewis and colleagues [9],[10] 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 [11] used such a model to manually design their OPTI layout. Zhai and colleagues [22] algorithmically optimized their ATOMIK layout based on the atomic interactions among all letters as defined by a Fitts-digraph energy function.

One weakness of 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 "mathematical theory of communication" [17]. 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. Similarly Green and colleagues have recently developed a reduced QWERTY layout that exploits language redundancy [6]. Another example is *Dasher* [20] 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 [13] has developed a stylus keyboard 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 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 [7]. In comparison, in order to type 60 words per minute (wpm) one has only 200 ms to process each character including all perceptual, cognitive, and motor control components.

Goodman and colleagues [5] proposed a method for stylus keyboard error correction that takes advantage of the language regularities without using prediction. 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' landing positions on the keys. A user study indicated that their model reduced the error rate as compared to verbatim tapping [5]. However it is not clear from their study to what degree the "tapping model" contributed to the error correction.

We propose a novel approach to exploit language regularities in stylus keyboards using geometric pattern matching. By mapping all words in a lexicon to the center positions of the keys in a stylus keyboard, the user's tapping pattern can be directly compared against a lexicon of words. The mismatch from such a comparison may serve as a basis for tapping-precision relaxation, as we shall see in the next section.

2. RELAXING FITTS' LAW

Fitts' law predicts that the mean time T to successfully hit a key of size W over distance D on a stylus keyboard is

$$T = a + bID \tag{1}$$

$$ID = \log_2\left(\frac{D+W}{W}\right) \tag{2}$$

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 constraint. 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:

1. 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.

2. The landing point of the stylus on a stylus keyboard 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 stylus keyboard. The center positions of all letter keys needed for inputting a word also form a pattern on the stylus keyboard. 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.

The approach is inspired by SHARK [21]. SHARK forms word patterns by tracing the letter keys comprising a word and matches the user's continuous gesture against them, essentially creating an efficient shorthand writing system. However there are important distinctions between self-correcting stylus keyboards and SHARK, both in terms of the amount of information in the task, pattern recognition technology and actual use behavior:

1. A user's tapping pattern contains more information than a user's continuous gesture. This is because a tap contains information about the user's intention – a tap with a pen on a stylus keyboard clearly means the user intended to hit a key in the proximity of the hit point. No corresponding information exists in SHARK where parts of a gesture can mean anything from a character, a chunk of characters to just being noise. Hence, a specialized system needs to be developed to maximize the utility of this extra information.

2. The recognition approaches in SHARK are tailored to recognizing continuous gestures. Using the SHARK approaches to recognizing word patterns (e.g. handwriting recognition [21] or a specialized multi-channel architecture [8]) would both wash out the extra information obtained and considerably complicate the recognition approach, adding additional performance overhead.

3. The interaction aspects of stylus typing correction are unknown. Can users take advantage of relaxation?

4. The UI aspects of the technique are unknown. For example, the choice of delimiter between words is crucial as we will show later in this paper.

The rest of this paper will be organized as follows. First we will describe the design and evaluation of a first-generation linear stylus keyboard correction system. Then we will describe the final "elastic" system we developed after gaining insights from user evaluations and practical use of the first system. Last we conclude by summarizing our experiences in developing and evaluating these kinds of correction systems and point to future work.

3. A LINEAR CORRECTION SYSTEM

We first experimented with a linear correction algorithm. In Figure 1 the user has tapped on the keys *r*, *j* and *w* on a QWERTY layout.

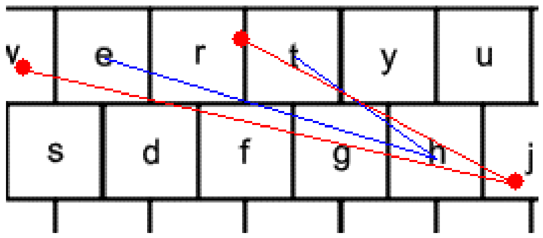

Figure 1. An example of accuracy relaxation.

Without any error correction "rjw" would be returned as the typed word. However, when looking at the hit points and comparing them with the corresponding points of the word "the" mapped on the keyboard layout (and all other words relevant in the lexicon), a close match is found. Hence we can return "the" despite the fact that the user missed all the targeted keys. We now present the first pattern matching algorithm we developed.

3.1 Similarity Measure

There are many pattern recognition and classification methods that can be used for the current problem; however most methods rely on large amounts of training data [2]. While such data could be collected, it would be a tremendous amount of work and be specific to a particular keyboard layout and language. We chose a simpler model-driven approach that matches the geometrical traces of the user's input and a word in the lexicon directly using an algorithm that has the following properties:

1. Scalable to a lexicon that practically includes all words needed by a user.

2. No training is required for the recognition algorithm.

First we need to decide on a delimiter method. Among other possible solutions such as a special-purpose physical button on the user's non-dominant hand, we currently use a set of delimiting characters. These characters are word delimiters in normal word processing, e.g. the tab-character, the space-character, semi-colon, etc.

Next, let X be an *unknown* pattern of n stylus hit points $\{x_i\}$, and let Y be a *template* pattern of n centers $\{y_i\}$ of keys in the keyboard corresponding to letters comprising a word w in the lexicon, $w \in \{w : w \text{ is a word with } n \text{ letters}\}$. The average spatial similarity between the patterns can then be computed with the following formula:

$$D(X,Y) = \frac{1}{n}\sum_{i=1}^{n}\left\|x_i - y_i\right\|_2 \qquad (3)$$

where $\|\cdot\|_2$ denotes the Euclidian norm. To avoid matching unlikely words we impose a threshold T on each point-to-point distance. If $D(X,Y) > T$, the distance to the word is set to ∞. Among all candidate words we then obtain a subset consisting of the words with a D value below the threshold T. These words are returned to the system as a ranked list. The system output is the word with the smallest D value.

The threshold T can be a fixed threshold, for example the diameter of a key on the keyboard layout, or be made more adaptive by, for example, looking at the distribution of the point-to-point distances. It is also possible to examine the actions of the user to determine the threshold dynamically. For instance, if many auto-corrections are followed by an immediate deletion by the user, the threshold can be increased by some amount.

The above matching method is simple and conservative. Specifically, if the user taps on all the correct keys of a word, no other word can be closer. Also, the above scheme is easy to implement and since we are comparing very few points, exhaustive (linear) search through the lexicon is very fast.

3.2 Evaluation

To assess the effectiveness of the linear correction system we conducted two experiments. Our main concern was if the simple linear correction algorithm was enough to correct the vast majority of the errors users make in stylus typing. A second concern was to determine if the input speed was increased as a result of relaxing the Fitts' law *W* constraint. For brevity we will omit some details of the experiments. The data presented should be interpreted as "lessons learned" which were fed into the next iteration of development of our stylus keyboard correction system.

3.3 Experiment I

3.3.1 Experimental Set-up

In the first experiment 14 participants of different gender and of various ages were recruited. We introduced two conditions: a verbatim condition where the user tapped on a stylus keyboard, and a "relaxed" condition employing the linear correction algorithm. Due to users' familiarity, we used the regular QWERTY layout for the experiment despite being non-optimal for stylus typing.

The experiment consisted of two tasks. Both used the pangram "The quick brown fox jumps over the lazy dog" as stimuli. This sentence is quite difficult for the linear correction algorithm on the QWERTY layout since the words "dog" and "fox" are close neighbors with "dig" and "fix". The lexicon used by the correction algorithm consisted of about 57,000 words supplied by Pitrelli and Roy [14].

The first task was performed in a *word* mode in which a single word at a time from the pangram was repeatedly presented to the user. The user was asked to type the word as fast and as accurately as possible. The user was not allowed to correct the input in this mode.

The second task was performed in *sentence* mode. The user was asked to copy the entire pangram "The quick brown …" and could not proceed until the whole sentence was completely correct, thus we forced users to either tap accurately or manually correct the input by re-positioning the caret and using the backspace key.

The order of the verbatim and relaxed conditions was alternated between the subjects to balance any learning effects. The number of word (task 1) and sentence (task 2) repetitions was 10 and 12 respectively. The input devices used in the experiments were a Wacom tablet or a touch-sensitive screen with a stylus. The conditions were balanced among the two different pen input devices.

3.3.2 Results

Taking the average of the last three sentences as the final speed, we analyzed the difference between the verbatim and relaxed stylus keyboard (Keyboard Type). Repeated measure variance analysis shows that the only significant factor is Keyboard Type × Order interaction: $F_{1,12} = 22.8$, $p < .001$. Neither order nor keyboard type alone was a significant factor in speed. This means that there was an asymmetrical skill transfer between the two types. As Poulton [15] argued, when there is an asymmetrical skill transfer, the only way to find the unbiased result is to take the data from the first condition presented, effectively turning the experiment to a between subject design, although the power of the experiment is much weakened. With such an approach, the difference between the two keyboard types was still insignificant, although the average speed of the relaxed stylus keyboard condition (29.5 wpm) was 24% higher than the verbatim condition (23.7). On average the participants made 8.7 errors in the verbatim condition and 5.3 in the relaxed condition. The difference was not statistically significant.

3.4 Experiment II

The first experiment revealed some but not clear advantage of using relaxation. To find out *if* there is any advantage with the technique after extensive learning we conducted a second experiment, designed to quickly get users up to speed with the technique. The users were only exposed to the *sentence* mode this time and were asked to write the word "computation" five times in sequence, ten times. We chose the word "computation" since the word is quite long and might be harder to reproduce quickly and accurately. Even though the task is artificial, it is appropriate for quickly simulating "expert" behavior.

3.4.1 Pilot Performance

A user with a total of a few hours of experience (both conditions) did a pilot test. The user typed the word "computation" 50 times (in 5 groups) in verbatim and relaxed conditions respectively. The user's total number of errors was 18 with the verbatim condition and 4 with the relaxed condition, suggesting the auto-correction in the relaxed condition had a positive effect. Counting the speed (converted to wpm) of the last 20 correctly typed words (the early words served as a buffer for the learning effect in this within subject test) the average was 50.1 wpm for the relaxed condition and 35.5 wpm for the verbatim. The difference was statistically significant ($F_{1,38} = 53$, $p < .0001$, within the subject). This suggests that, for advanced users, there could be a significant advantage to relaxation.

3.4.2 Experimental Set-up

We recruited 26 unpaid volunteers. Some of these had participated in the earlier experiment, and these were balanced among the two conditions. Each participant entered the word "computation" 50 times in 5 groups. The user had to correct errors remaining in each group before proceeding.

3.4.3 Results

The mean speed of the participants was in favor of the relaxed condition. Their average speed was 33 wpm in the relaxed condition and 30 wpm in the verbatim condition. However the difference was not statistically significant. There were large individual differences in speed. Figure 2 plots the speed against participant number (ranked by performance, highest rank corresponds to the fastest subject). At each corresponding rank, the participant in the relaxed condition almost always outperforms his or her counterpart in the verbatim condition, but the performance is more pronounced with the top performers. It appears that some users could take advantage of the relaxation and exploit it more than others.

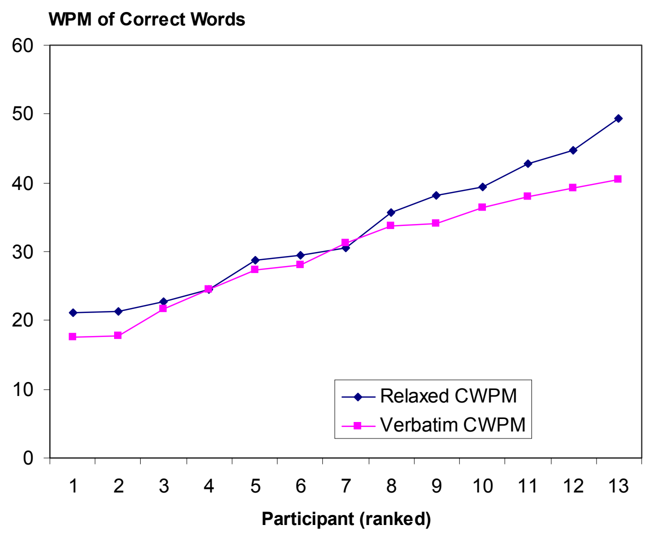

Figure 2. Ranked participant wpm scores for both conditions.

The total numbers of errors that had to be corrected by the user in all five sentences were 38 in the verbatim condition and 22 in the relaxed condition. The latter does not include errors automatically corrected into the correct word by the linear correction system. For the relaxed condition, we performed an analysis of the corrections on the individual words for the last three sentences of each subject. A total of 46 errors were found (including errors automatically corrected into the correct word). Table 1 shows a classification of these errors.

Errors automatically corrected	24	52%
Error corrected by the user before the matcher	17	37%
Deletions causing erroneous correction	2	4.5%
Missed the space key	2	4.5%
Other	1	2%

Table 1. Errors in the user study.

There were 24 instances where an error was auto-corrected to the correct word. Hence the error correction method captured most mistakes. The majority of the remaining errors were corrected by the user before auto-correction was applied. Among all the cases where automatic error correction took place the majority was successful: 24 instances were corrected correctly, which amounts to a success rate of 83%. In two cases deletions (the user omitted a character) caused erroneous corrections ("compuation" and "computtion" caused the system to correct them into "completion"). In two other cases the space key was missed, causing concatenation (of two words into one so the user had to move the caret between the words and insert a space character afterwards). These two categories of errors were very detrimental to users' perception of and performance with the relaxed keyboard. They felt they could not really trust the system since it may give them implausible results when they (unconsciously) missed a key, particularly when the missed key was the space key, which acted as the delimiter. This also made them more conservative in taking advantage of relaxation algorithm and pushing for higher speed.

We concluded that the 83% successful error correction rate was still too low to be practical. Furthermore, the remaining errors were problematic from a user experience point of view since they caused the users to trust the system less. Trust is also a general challenge to intelligent user interfaces. This led us to develop a much improved solution to pattern-based precision relaxation.

4. AN ELASTIC CORRECTION SYSTEM

There were a few lessons learned after evaluating the simple linear correction system:

1. The stylus keyboard correction scheme worked in that it did correct the majority of the users' errors. However it will probably take a long time before users are proficient enough to take advantage of the correction mechanism to gain higher input speeds.

2. The choice of delimiter is crucial. The space key is a bad delimiter since it is the most likely key being hit. Erroneous delimitations must be avoided since they will result in incorrect replacements by the system or no correction at all if the user fails to delimit two long words.

3. The algorithm is too simplistic in that it can only handle patterns of the same length. The study revealed that users made deletion errors that the algorithm could not correct. Furthermore, users requested that the algorithm should handle insertion and transposition errors too. In general, the success rate of the algorithm must be very high, since erroneous corrections results in much more labor for the user than if the algorithm didn't do anything at all.

Based on the empirical data gathered in the experiments, improving delimitation and handling patterns of different lengths would directly lead to a 97% error correction success rate.

4.1 Elasticity in the Matching Process

In Figure 3 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. With the linear matcher no correction at all or an erroneous correction would be returned. We will now present a system capable of correcting the input despite the fact that the user missed all the targeted keys and accidentally had one spurious stylus tap. We call a keyboard employing such a system an Elastic Stylus Keyboard (ESK).

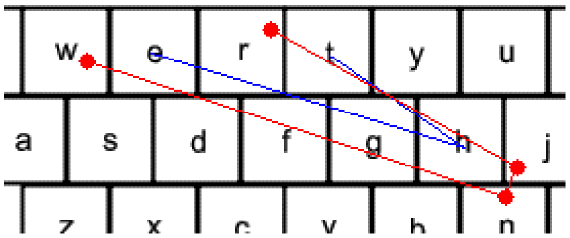

Figure 3. An example of error correction: the user tapped on r-j-n-w but the intended t-h-e is returned.

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.

4.2 Similarity Measure

Let X denote an *unknown* pattern consisting of an ordered sequence of n stylus hit points $\{x_i\}$ on a stylus keyboard, 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 $x_1 \dots x_n$ against $y_1 \dots y_m$. Wang and Pavlidis [19] showed that $K(i,j)$ for the subsequences $x_1 \dots x_i$ against $y_1 \dots y_j$ can be computed using a recurrent equation of similar form as the traditional edit-distance between two strings over a finite alphabet. However this form was difficult to use since it involved a constant penalty in inserting or ignoring a single point. Intuitively the cost of inserting or ignoring a point should depend on the distance between the points matched. For this reason, we modified the recurrence given by Wang and Pavlidis [19] by multiplying the constant penalty τ for inserting or ignoring a point, with the actual distance between the points compared:

$$K(i,j) = \min \begin{Bmatrix} K(i-1,j-1) + \delta(x_i,y_j), \\ K(i-1,j) + \tau \times \delta(x_i,y_j), \\ K(i,j-1) + \tau \times \delta(x_i,y_j) \end{Bmatrix} \qquad (4)$$

where $K(0,0) = 0$, $\delta(x_i,y_j)$ is the stretching cost x_i to y_j, and τ is an empirically determined parameter weighting 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 $\delta(x,y)$ as

$$\delta(x,y) = \begin{cases} \infty & \|x-y\|_2 > r \\ \|x-y\|_2 & \text{otherwise} \end{cases} \qquad (5)$$

where r is the maximum distance a point may be stretched. It is important to constrain the algorithm in this manner since it is unclear whether the above algorithm satisfies the triangle inequality [3]. Unintuitive matches may result if we do not clearly define the point-to-point constraints. See [3] for a comprehensive discussion of the subtleties in defining "elastic matching" algorithms.

To be able to make fair comparisons between patterns of unequal length we normalize the similarity measure. Since correct normalization of Equation 4 is non-trivial [12] we compute a pseudo-normalized minimum stretching cost:

$$D_N(X,Y) = \frac{D(X,Y)}{n+m} \qquad (6)$$

It is well known that the computation of $K(n,m)$ for matching X against Y in Equation 4 can be solved efficiently using dynamic programming in $O(nm)$ time [19]. The best matching word is the word whose pattern has the lowest normalized stretching cost D_N against the user's tapping pattern.

The algorithm presented can be seen as a generalization of a minimum edit-distance algorithm where we replace the character equivalence function (which in the most simplistic case is either 0 if the characters match or 1 otherwise) with a Euclidean distance between the points.

4.3 Indexing

Unlike the linear correction algorithm, the algorithm presented in Equation 4 has quadratic complexity. To avoid a resource-intensive exhaustive search of a large ($\approx 57{,}000$ words) lexicon we implemented an indexing technique that considerably narrows down the search space. Since Equation 5 constrains corresponding point-to-point distances between the unknown input and the template pattern to be shorter than r, by eliminating template patterns whose first and last point do not meet this constraint we may reduce the search space significantly. If r is sufficiently conservative, e.g. 1.5 times the radius of an alphabetical key on the keyboard, it directly leads to an effective indexing method.

We construct an ordered *k*-ary tree data structure of height 2 where *k* is the number of alphabetical keys on the keyboard layout. Each node at index i at depth d represents a circular cluster C_{id} where the ith key center is the cluster center, and r is the radius of the cluster. At index i, $1 \le i \le k$, C_{i1} represents a *start* position cluster and C_{i2} represents an *end* position cluster. A pattern Y of length m is *indexed* by a pointer in a cluster C_{j2} at depth 2 iff $y_1 \in C_{i1}$, $y_m \in C_{j2}$ 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 at 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 lexicon, containing 57,000 words, the largest possible set that needs to be searched exhaustively is about 4,000 words when $r = 1.5$ in keyboard key radius units.

4.4 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 $T = 1.0 \times r_k$, where r_k is the radius of a key.

4.5 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 [14], 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.

Such an "optimized lexicon" can be retrieved by for instance supplying a "core lexicon" with the system containing the 5,000-7,000 most frequent words in the language, and a function to mine the user's documents and email to "fill in" the majority of the words missing from the user's vocabulary.

4.6 Delimiter

As mentioned earlier, for the linear correction system we first used the most obvious choice of a delimiter — the space key tap which proved problematic.

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 method proves to be quite effective — gesturing is distinctly different from a tapping action and yet easy to evoke between tapping actions.

4.7 Advantages

The advantages of an ESK in comparison to a statistical letter sequence approach [5] 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. Since the system is using the geometrical tapping trace, some amazing corrections can be done. It is for example possible to correct the user's input even if the user missed all the intended letter keys (see Figure 3 and Figure 4), something that is virtually impossible to correct without taking the spatial hit point information into account. Furthermore, the intuitive spatial interpretation of the matching method may enable expert users to consciously take advantage of the error correction scheme.

4.8 Evaluation

Proficiency in a text entry system such as the ESK is a function of practice. The closed-loop action of typing on a regular stylus keyboard is limited by the human motor control system and can be reliably modeled using letter digraph statistics and Fitts' law [22]. Since an ESK relaxes or "breaks" Fitts' law we cannot, as of today, model expert performance using any known human performance law. Also note that the recognition precision of an ESK varies with the size and contents of the lexicon.

Instead of performing a theoretical modeling of the performance of an ESK, we simulated expert performance by letting two users repeatedly write selected sentences correctly (errors were not allowed). The test was carried out on a Tablet PC with an ESK using the QWERTY keyboard layout and a lexicon containing around 57,000 words. The results are shown in Table 2. Note that these numbers are "record" speeds that do not reveal the true average and expert typing performance of an ESK, and should only be considered a demonstration of the potential of the technique in "breaking" Fitts' law in stylus typing. Also note that we used a very large lexicon as a stress test of the technique and our implemented system. A smaller "optimized" lexicon (as discussed in section 4.5) would increase the probability of desired automatic correction. As a reference, the theoretical average expert typing speed on a QWERTY stylus keyboard has been estimated to around 34.2 wpm [23].

Testing phrase	User 1	User 2
the quick brown fox jumps over the lazy dog	46.3	37.7
ask not what your country can do for you	45.4	40.1
intelligent user interfaces	51.3	51.8

Table 2. "Expert" speed estimates (wpm) of two users.

5. FEEDBACK

We have implemented and experimented with three feedback methods for the ESK. First, for each key tapped the corresponding character is outputted, just as in a normal stylus keyboard. To help users get an understanding of the geometrical pattern approach we experimented with outputting the hit points of the stylus taps (slowly fading away by time) as a way to "hint" to the user that the proximity information is taken into account.

Second, since the system replaces the input with something else, it is important to inform novice users about the replacements. We currently achieve this by drawing the word replaced with a distinct background color (Figure 4).

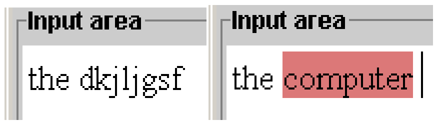

Figure 4. Example of correction indication.

Third, since the matching algorithm outputs normalized scores it is possible to collect a ranked list of the best matches (*N*-best list). By pressing on a corrected word with the stylus the user can bring up a selection widget allowing direct access to the *N*-best list.

6. 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 when receiving unintended words than appreciating correct error corrections. It is hence necessary to be conservative.

We have explored using a geometric pattern recognition approach to relax the precision requirements in stylus typing. The first implementation of this approach, the linear correction system, was tested in two experiments. The data and experiences from these experiments guided our development of the ESK. Although we have not evaluated the ESK as formally as the first linear correction system, informal testing indicates that the ESK performs much better. The two key weaknesses observed with the first version of the relaxed stylus keyboard (the linear correction system) were solved. First, since we now use a gesture distinctively different from tapping as the word delimiter, the delimitation problem caused by missing the space key found in the experiment is eliminated. Second, due to the introduction of elasticity in the matching algorithm, the precision relaxation system now returns correct results even when a required key tap was missed or an extra tap accidentally added, as long as the overall shape of the input pattern matches the desired target word better than any other alternative. As a result of these improvements, a user can trust the system much more and be more comfortable in taking advantage of the precision relaxation.

An ESK is essentially letting users take advantage of the regularities in the languages in a novel way – relaxing precision constraints by pattern matching, hence "breaking" Fitts' law. Prior techniques have either relied on users consciously "compressing" the input [16] or using prediction that demands cognitive and visual reaction time. In contrast an ESK does not require a user to learn a compression technique or hope for the system to predict the desired input. An expert ESK user simply taps the word as quickly as possible and relies on the redundancy in the language to make sure that word patterns are sufficiently separated.

We see future improvements in primarily two domains. First, we noted from the evaluation of the first linear correction system that the user corrected the input before the system in 37% of the cases.

An extension would be to make the system "eager" – recognizing the user's input before delimitation. Such a feature would signal to the user that the system is automatically correcting the input right away. Second, an improvement would be to use context or common sense information [18] about the text the user is writing to constrain the lexicon dynamically, thus increasing the probability of correcting the user's input. In short there are many opportunities for intelligent user interfaces in this domain.

An ESK works with any stylus keyboard layout, either QWERTY or an optimized layout. Potentially the ESK technique could also be used to correct eye-typing and other text entry interfaces. 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.

7. ACKNOWLEDGMENTS

We thank John Karidis for contributing to the initial invention of ESK. Part of this work was conducted when Per-Ola Kristensson was a graduate intern at IBM Almaden Research Center.

8. 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, Elsevier Science.

[2] Duda, R. O., Hart, P.E. and Stork, D. G. *Pattern Classification (2nd ed.)*, 2001, John Wiley & Sons.

[3] Fagin, R. and Stockmeyer, L., Relaxing the Triangle Inequality in Pattern Matching. *Intl. J. Computer Vision,* 28(3), 1998, 219-231, Kluwer.

[4] 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, RESNA.

[5] Goodman, J., Venolia, G., Steury, K. and Parker, C., Language modeling for soft keyboards. *Proc. AAAI 2002*, 2002, 419-424, AAAI Press.

[6] Green, N., Kruger, J., Faldu, C. and St. Amant, R. A reduced QWERTY keyboard for mobile text entry. *Extended Abstracts CHI 2004*, 1429-1432, ACM Press.

[7] Keele, S.W. Motor Control. in Boff, K.R., Kaufman, L. and Thomas, J.P. eds. *Handbook of Perception and Human Performance*, 1986, 30.31-30.60, John Wiley & Sons.

[8] Kristensson, P-O. and Zhai, S. SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. *Proc. UIST 2004, CHI Letters* 6(2), 43-52, 2004, ACM Press.

[9] 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, IBM.

[10] 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*, 1997, 1285-1315, Elsevier Science.

[11] MacKenzie, I.S. and Zhang, S.X., The design and evaluation of a high-performance soft keyboard. *Proc. CHI 1999*, 1999, 25-31, ACM Press.

[12] Marzal, A. and Vidal, E. Computation of normalized edit distance and applications. *IEEE Trans. Pattern Analysis and Machine Intelligence,* 15(9), 1993, 926-932, IEEE Press.

[13] Masui, T., An Efficient Text Input Method for Pen-based Computers. *Proc. CHI 1998*, 1998, 328-335, ACM Press.

[14] Pitrelli, J.F. and Roy, A. Creating Word-Level Language Models for Handwriting Recognition. *lntl. J. Document Analysis and Recognition*, 5(2&3), 2003, 126-137, Springer Verlag.

[15] Poulton, E. C. Unwanted asymmetrical transfer effects with balanced experimental designs. *Psychological Bulletin*, 66(1), 1966, 1-8, APA.

[16] Schieber, S.M. and Baker, E. Abbreviated text input. *Proc. IUI 2003*, 2003, 293-296, ACM Press.

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

[18] Stocky, T., Faaborg, A. and Lieberman, H. A commonsense approach to predictive text entry. *Extended Abstracts CHI 2004*, 1163-1166, ACM Press.

[19] Wang, Y.P. and Pavlidis, T. Optimal Correspondence of String Subsequences. *IEEE Trans. Pattern Analysis and Machine Intelligence.*, 12(11), 1990, 1080-1087.

[20] Ward, D., Blackwell, A. and MacKay, D., Dasher - A data entry interface using continuous gesture and language models. *Proc. UIST 2000 , CHI Letters* 2(2), 2000, 129-136, ACM Press.

[21] Zhai, S. and Kristensson, P.-O., Shorthand Writing on Stylus Keyboard. *Proc. CHI 2003, CHI Letters* 5(1), 2003, 97-104. ACM Press.

[22] Zhai, S., Smith, B.A. and Hunter, M. Performance Optimization of Virtual Keyboards. *Human-Computer Interaction*, 17(2,3). 2002, 89-129, Lawrence Erlbaum Associates.

[23] Zhai, S., Sue, A. and Accot, J. Movement Model, Hits Distribution and Learning in Virtual Keyboarding. *Proc. CHI 2002, CHI Letters* (4)1, 2002, 17-24, ACM Press.