SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-based Computers

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
Zhai and Kristensson (2003) presented a method of speed-writing for pen-based computing which utilizes gesturing on a stylus keyboard for familiar words and tapping for others. In SHARK^2, we eliminated the necessity to alternate between the two modes of writing, allowing any word in a large vocabulary (e.g. 10,000-20,000 words) to be entered as a shorthand gesture. This new paradigm supports a gradual and seamless transition from visually guided tracing to recall-based gesturing. Based on the use characteristics and human performance observations, we designed and implemented the architecture, algorithms and interfaces of a high-capacity multi-channel pen-gesture recognition system. The system's key components and performance are also reported.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Input Devices and Strategies, Interaction Styles; I.5.2 [**Design Methodology**]: Classifier Design and Evaluation, Feature Evaluation and Selection

Additional Keywords and Phrases: Text input, shorthand, stenography, shorthand recognition, gesture recognition

INTRODUCTION
Since the advent of pen-based computers, researchers have sought intuitive and efficient ways to input text on computers using a digital pen (see [14, 28] for two recent reviews). Two approaches have gained broad deployment in pen-based computing products – handwriting recognition systems and the stylus keyboard (SK). Taking advantage of the user's years of prior experience, handwriting recognition systems allow users to enter text either as cursive script or as individual letters. Decades of research and development effort have been invested in handwriting recognition technology [20, 24], resulting in increasingly practical commercial systems such as *Graffiti* for Palm OS, *Jot* for Pocket PC, and the handwriting recognition built into the Microsoft Windows Tablet PCs. However, handwriting recognition is limited by the speed of "longhand" writing which is about 15 wpm [5] as well as problems with recognition accuracy [7]. Gesture based text entry methods such as the Unistroke alphabet [9], Graffiti, Cirrin [16], Quikwriting [18] and Edgewrite [25] can be more efficient or more robust than natural alphabets but still require character level articulation.

The SK, also known as a graphical, soft, or on-screen keyboard, is a "virtual" keyboard drawn on the computer screen and tapped serially with a stylus. Much effort has been made to increase statistical movement efficiency in the SK (e.g. [8]). Two recent efforts in this regard are the OPTI [15] and ATOMIK keyboards [26], designed with heuristics and computer algorithms respectively. However an SK can be tedious to use and requires constant visual attention since every key tap must be exactly within the key boundaries.

Combining handwriting with SK, Zhai and Kristensson [27] developed a method that allows the user to draw patterns as a basic mode of entering common words (see Figure 1). Each pattern of a word is formed by the trajectory through all of the letters of the word on the SK, from the first to the last in order. We call such a pattern a *sokgraph*, since it is essentially a form of ***s***horthand defined ***o***n a ***k***eyboard as a ***graph***. An input system that uses the sokgraph approach was dubbed SHARK - ***SH***orthand ***A***ided ***R***apid ***K***eyboarding [27]. The original SHARK system uses dual modes of text entry. For an unfamiliar word, the user taps it as usual on a SK. For a familiar word, the user writes the gesture, and the system uses a matching algorithm to find the closest sokgraph. Due to the use of a lexicon, sokgraph recognition incorporates error tolerance afforded by the regularities in natural words composition. The feasibility of using and recognizing a small set of sokgraphs has been shown in [27]. Although artificial, the users could learn about 15 sokgraphs per hour of practice. (See [27, 28] for the related prior arts to SHARK).

While promising, substantial challenges have to be overcome to make SHARK a practical writing tool. This

UIST '04, October 24–27, 2004, Santa Fe, New Mexico, USA.

paper presents our innovations in designing the use behavior, architecture, algorithms, user interfaces and key implementation components of a complex interaction system to make SHARK a practical text writing approach for pen-based computing. We call the new system SHARK^2, since the capacity of the system in terms of recognizable sokgraphs has to rise by a power of two from the original prototypical system [27] in order to be practical.

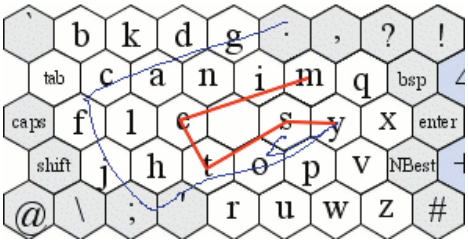

Figure 1. A user has written the word "system" in SHARK^2 on a 57K lexicon. As is evident in the figure, the system can handle gestures that are far away from the ideal sokgraph (marked in bold), both in terms of overall shape and location proximity. Shown is the ATOMIK layout. Sokgraphs can be defined on any layout, such as QWERTY.

CHALLENGES AND BEHAVIORAL DESIGN

The Problem

One of the design goals of the original SHARK method was to use the SK as a bridge for learning which assists the user to move from the tedious tapping mode to the more efficient gesturing mode. This is similar in spirit to the way a self revealing "marking menu" supports novice to expert transition for menu selections [11]. However, the distinct two modes of input is a significant impediment to a seamless migration from tapping to gesturing. Although tapping a word and gesturing its sokgraph constitute the same movement trajectory pattern, the discrete tapping motion and the continuous gesturing movements are different in many ways. Visually and cognitively, discrete tapping does not explicitly reveal the geometric pattern of the corresponding sokgraph to the user. The user's realization and memorization of the sokgraph is hence slow and covert. Kinematically, the actions of tapping individual letters of a word are distinctly different from producing the total gesture of a sokgraph of the word. The two are likely to require different "motor programs" or "motor schemas" [22], which also hinders the amount of learning and memory transfer from tapping to gesturing.

New Use Behavioral Paradigm

What is needed is a mechanism that supports a smooth and gradual transition from novice to expert behavior without the distinct mode switch. The solution we have designed is to use continuous letter tracing, rather than discrete letter tapping, as the novice "mode". Tracing is a visually guided, closed-loop action. To trace a word on the SK the user is not required to have any prior knowledge of the word's sokgraph. However, the trajectory of the trace, explicitly displayed to the user by transient stylus ink, is the same as the sokgraph shape. The kinematics of tracing and gesturing are also the same. This means that each trial of tracing a word on SK is also a trial of learning its sokgraph. Over time, the pattern of the sokgraph builds up in the user's memory so the production of the trace becomes partly visually guided and partly memory recall driven. As the contribution of pattern recall, or open-loop action increases, the user's dependence on visual guidance will decrease. Eventually a user may completely remember the sokgraph and gesture it based primarily on memory recall, entering the expert "mode". Between visual tracing and memory recall the system is in fact modeless. The only difference is the degree of visual guidance reliance. A user's behavior is always somewhere between the two extremes but *gradually* shifts from closed-loop to open-loop performance with practice. The original tapping to gesturing binary switching paradigm now changes to a seamless and continuous transition from tracing to gesturing.

System Requirements

A challenge raised by this revision of the SHARK paradigm, however, is the large number of sokgraphs the system has to recognize and distinguish. To support such a paradigm change, a recognition-based text writing system has to be developed with the following necessary or desirable properties:

1. Every instance of word entry, except adding new words to the lexicon by tapping, has to be realized by sokgraph recognition. To the system there is no distinction between visually guided tracing and recall-based gesturing. The recognition system has to be able to recognize all words an individual user may use in regular writing*. The size of the lexicon should be in the order of 10,000 words.

2. The system has to be extensible to new words a user may adopt in writing.

3. The system has to be "real time". The recognition latency can only be a small fraction of the entire duration of writing a word.

4. The system should be compatible with the SHARK^2 paradigm, supporting gradual transition from visually guided tracing to recall driven gesturing.

5. The system should give the user the maximum amount of flexibility and least amount ambiguity.

*Word variations based on the same stem are considered different words in this context. For example "work", "worked", "working" and "works" are four separate words.

THE ARCHITECTURE AND ALGORITHMS OF THE GESTURE RECOGNITION ENGINE

A Multiple Channel Architecture of Recognition

Three characteristics set the SHARK^2 gesture system apart from other gesture recognition systems. First, the number of sokgraph gestures each individual user may need is much greater than the gesture repertoire of an alphanumeric recognizer such as *Graffiti* or *Jot*, or a typical command gesture recognizer based on a linear machine (e.g. [21]), often used in HCI research projects (e.g. [17]). Second, unlike natural (longhand) cursive handwriting recognition, a fraction of the sokgraph shorthand gestures, particularly those for short words, can be identical or similar in shape; therefore shape alone may not provide sufficient information to recognize the user's intent. Third, a set of sokgraphs defined on a keyboard layout constitutes a symbolic system novel to the user. This is both an advantage and a disadvantage. It is an advantage because each sokgraph has a unique ideal prototype, in contrast to natural hand writing in which even the same letter can be written in perfectly legitimate but very different styles. It is also a disadvantage because there is not a natural corpus of sokgraphs that can be collected, precluding many of the standard data-driven machine learning approaches to recognition (see [6] for a comprehensive overview).

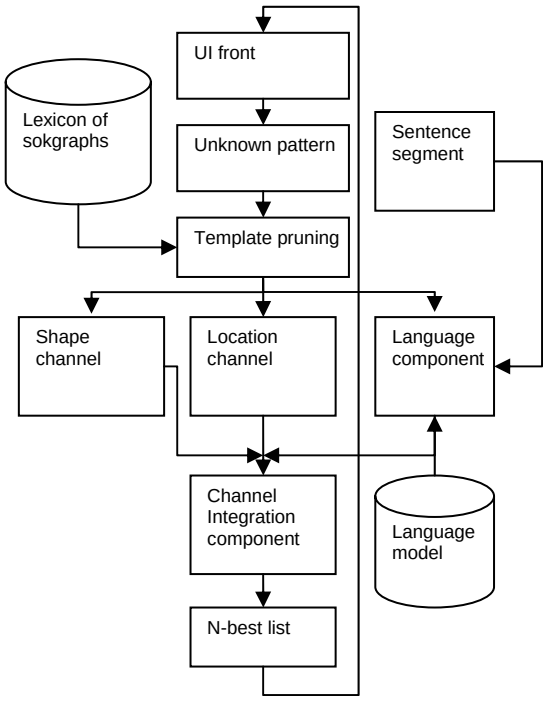

Figure 2. A multi-channel architecture for sokgraph recognition.

In consideration of these requirements we developed a multi-channel recognition system (see Figure 2). Each channel does not necessarily have enough discriminative power, but the collective information from the multiple channels can separate the sokgraphs sufficiently. The two core channels are a shape recognizer and a location recognizer. The former classifies a pen gesture according to the normalized (scale and translation invariant) shape of the pen gesture. The latter classifies a pen gesture according to the absolute location of the gesture on the keyboard.

Both the shape channel and the location channel draw their recognition templates from a lexicon. The SHARK^2 paradigm requires the lexicon to include all (but just enough) words a particular user needs in regular writing. This lexicon 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. In practice it is a combination of all. We currently use a base lexicon adapted from [19].

Template Pruning

Due to the massive vocabulary required for the SHARK^2 system, an initial pruning component first filters out a large number of the sokgraph templates from entering later stage recognition channels. An effective filtering mechanism we found is based on the start and end positions of the sokgraph templates, normalized in scale and translation. We compute the start-to-start and end-to-end distances between a sokgraph template and the normalized unknown input gesture. If either of the two distances is greater than a set threshold, the template will be discarded. This pruning process was implemented efficiently by only storing the coordinates of the start and end points of a template pattern in a linked list. Traversing the list and collecting the templates that pass this filter is thus an inexpensive operation, even for large datasets.

Shape Channel Recognition

The most basic means of sokgraph classification is based on the shape information contained in the user's input gesture. There are many approaches to on-line shape similarity measurements. Most relevant to the current work are those methods used in pen-gesture recognition [13, 17] or handwriting recognition [24, 20].

Pen-gesture recognition commonly uses trained classifiers based on the classic linear machine [6] (pp. 36-39). A particular popular variant thereof often cited in the HCI literature is the Rubine pen-gesture recognizer [21].

Handwriting recognition systems use a multitude of techniques and approaches, including neural networks, hidden Markov models, and model matching [24, 20]. An early model based approach to cursive script recognition system by Tappert [23] pioneered the use of so-called *elastic matching* in handwriting recognition. While being outperformed in cursive script recognition by statistical classifiers nowadays [3], elastic matching retains the valuable property of not requiring any training at all. Elastic matching in cursive script recognition measures the spatial similarity of two patterns by comparing the point-to-point

correspondences in sequence, allowing certain elasticity if a nearby point has a shorter spatial distance than the corresponding point (see Figure 3 left).

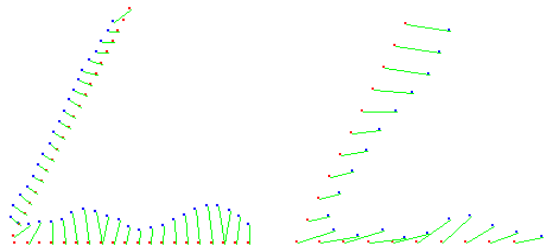

Figure 3. Elastic matching of two patterns with a look-ahead threshold of 2 points (left). Proportional matching of two patterns (right).

We initially experimented with using elastic matching for the SHARK[2] system, but in early testing we found that elasticity in the matcher is an undesirable classification property when the template space is crowded with nearby competing patterns. We found that we could significantly reduce the error rate by introducing a linear matcher, which we call (for reasons that will soon become apparent) a *proportional matcher*.

Proportional Shape Matching The proportional shape matching distance between an unknown pattern u and a template pattern t is defined as

$$x_s = \frac{1}{N}\sum_{i=1}^{N}\left\|u_i - t_i\right\|_2 \qquad (1)$$

where N is the total number of sampling points of the patterns. It is easy to see that the elastic matching algorithm presented by Tappert [23] with zero look-ahead reduces to Equation 1. Before applying Equation 1 the patterns are re-sampled into N equidistant points and normalized in scale and location.

Normalization is achieved by scaling the largest side of the bounding box of a pattern to a pre-determined length L:

$$s = L / \max(W, H) \qquad (2)$$

where W and H are the original width and height of the bounding box. Finally we translate the pattern's geometric centroid to the origin in the coordinate system.

The final result of the shape channel is an approximate scale and translation invariant distance measure of the similarity between the patterns, based on the average sum of the equidistant sample points' spatial distance (see Figure 3 right).

Location Recognition Channel

The second core channel of sokgraph recognition in our current architecture examines the absolute location of the user's gesture trace on the keyboard. The rationale for having such a channel is twofold: 1. Location information provides an increased recognition resolution of sokgraphs; 2. Location is part of the user's memory of a sokgraph and therefore will be reproduced during gesture production.

Shape Confusion and Location Information By sokgraph shape alone some words may be near or in complete confusion with each other. To gain a quantitative baseline understanding of sokgraph confusion, here we present a brief analysis of the confusion pairs of sokgraphs in a 20K lexicon [19] on an ATOMIK layout. As shown in Table 1, there are a small fraction of words that have identical sokgraphs. Considering normalized sokgraph shape only (independent of scale and location), there are 1117 pairs of words that have identical sokgraphs (confused pairs), for example *root* vs. *heel*, *mend* vs. *shea*, *abe* vs. *ids*, *can* vs. *cam*, and *ben* vs *buy*. Many of these conflicts are not natural or complete English words. For example, in a 20K lexicon, the word *at* conflict with *du*; *as* conflicts with *lo*, *oz*, *by*, *ny* and *ft*; *rjr* conflicts with *sas* (See Figure 1 for the ATOMIK layout).

	QWERTY	ATOMIK
Shape	1461	**1117**
Shape & start key	609	**519**
Shape & end key	589	**522**
Shape & both ends	537	**493** (**284** Roman Numerals)

Table 1. The number of confusion sokgraph pairs in a 20K lexicon, considering shape only, shape and start key position, shape and end key position, and shape and both start and end key positions

If we consider shape plus the starting key position, the number of confusion pairs reduces to 519. If we consider shape plus ending key position, the number is 522. If we consider shape plus both the start and ending key positions, the number of confusion pairs reduces to 493. Examples of confusion pairs with identical start and ending positions include *refuge* vs. *refugee*, *webb* vs. *web*, and *traveled* vs. *travelled*. 284 pairs (58%) of these remaining confusions pairs are Roman numerals, such as "lxvi" vs. "lxxxvi", "xci" vs. "xcii", and "mmxvii" vs "mmxviii". Table 1 also shows the numbers of confusion pairs for the sokgraphs defined on QWERTY layout. Note that we have only considered identical conflicts, which give us a baseline understanding of the percentage and relative contribution factor of conflict. In conclusion, location cue helps to reduce sokgraph ambiguity.

Location Memory Since in the SHARK[2] paradigm, users initially use and learn a sokgraph by tracing the letters, the sokgraph location on the SK plays an important role in the

memory of the each individual sokgraph. Even for a completely memorized sokgraph, the user is likely to remember its approximate location on the SK together with its shape, particularly the beginning and ending positions.

Location Channel Algorithms Therefore we use a location algorithm that computes the distance of the unknown input trace u to the template (ideal) trace t of word w on the SK (now both u and t are absolute). t is defined by the lines connecting the centers of the letters that constitute w. Both t and u are re-sampled to a fixed number N of equidistant points. The location channel distance is defined as:

$$x_L = \sum_{i=1}^{N} \alpha(i)\delta(i) \tag{3}$$

where N is the number of points in the patterns. δ is defined as:

$$\delta(i) = \begin{cases} 0, & D(u,t) = 0 \wedge D(t,u) = 0 \\ \|u_i - t_i\|_2, & \text{otherwise} \end{cases} \tag{4}$$

where u_i is the i th point of u. D is in turn defined as:

$$D(p,q) = \sum_{i=1}^{N} \max\left(d(p_i,q) - r, 0\right) \tag{5}$$

where d is

$$d(p_i,q) = \min\left(\|p_i - q_1\|_2, \|p_i - q_2\|_2, \ldots, \|p_i - q_N\|_2\right) \tag{6}$$

where N denotes the number of sampling points in the patterns. r is the radius of an alphabetical key. This means that we form an invisible "tunnel" of one key width that contains all letter keys in w. A perfect distance score of zero is given when the entire gesture input trace u is within this tunnel of t. Otherwise, the sum of the spatial point-to-point distances is used. In other words, Equations (3-6) give special weight to traces that are contained within the tunnel of radius *r* whose path is formed by serially connecting all the individual keys used in a word. This was based on the observation from actual use that when all letters in a word is traced ("tunneling"), one would expect the word to be recognized no matter what the shape of the trace is.

$\alpha(i), i \in (1, N)$ are weights for different point-to-point distances ($\sum_{i=1}^{N} \alpha(i) = 1$). The shape of $\alpha(i)$ can be set in various ways. For example it could be dynamically trained through a large amount of data when available. It can also be prescriptively set. Currently we use a function that gives the lowest weight to the middle point, and the rest of the points' weights increase linearly towards the two ends. This is because when producing a gesture it is easier for the user to pay visual attention to the beginning and ending points than the rest of the locations.

Channel Integration

The shape and location channels output distance scores between an unknown gesture and templates drawn from the lexicon. These distances in the two channels are not on a common scale and cannot be directly compared. The issue of multiple classifier integration of distances or scores is not new in pattern classification. Several methods are proposed using such methods as voting or distance to rank conversion [6]. Bouchaffra et al. [4] present a method of deriving probabilities from handwriting recognizer scores based on training. Lacking training data, we devised our own method to convert channel distances to a common integration scale.

As is common in engineering, a reasonable assumption is that the distance from an input gesture to the template of the intended word (in either channel) follows a Gaussian distribution. In other words, if an input gesture has distance x to a template y, the probability of y being the targeted word can be calculated using the Gaussian probability density function:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{7}$$

where $\mu = 0$ and σ can usually be obtained through training from large amount of data. σ reflects how sensitive a channel is. For example if σ equals to one key radius, those templates whose distance to the input gesture are greater than one key width (2σ) have practically zero probability of being the intended sokgraph. In our current system we prescriptively use σ as a parameter to adjust the weight of the contribution of each channel. The greater σ is, the more flat the $p(x)$ distribution will be, and hence the less discriminatory the channel is when it is integrated with the other channel (hence less weight). As a pruning measure all candidates (templates) with $x > 2\sigma$ are discarded without further processing. Among the remaining candidates $w \in W$, the marginalized probability of a word w with distance x being the user intended target word is:

$$p'(w) = \frac{p(x)}{\sum_{i \in W} p(i)} \tag{8}$$

Finally we integrate the probabilities from the two channels using Bayes' rule and obtain a confidence score for the word:

$$c(w) = \frac{p'_S(w)\,p'_L(w)}{\sum_{i \in W_S \cap W_L} p'_S(i)\,p'_L(i)} \tag{9}$$

where $p'_S(w)$ and $p'_L(w)$ are the probability scores from the shape and location channel respectively; W_S and W_L are the sets of the remaining candidates that passed the 2σ threshold pruning stage in the shape and location channel respectively. The result of the integration process is a ranked list of the templates ordered by confidence.

Dynamic Channel Weighting by Gesturing Speed

The shape and location channels extract different types of information from the user's input gesture. As discussed earlier, the user could base the production of a sokgraph gesture on either visual guidance from the SK or recall from memory, depending on the degree of familiarity with the particular sokgraph. Visual guidance results in greater location dependency than memory recall. If the user visually traces a sokgraph letter by letter on the keyboard, the location channel should yield a high score for the traced word. On the other hand, if the user produces a gesture primarily based on fast open-loop memory sokgraph recall, the location channel is likely to yield a poor score since the user could only aim approximately at the initial landing position on the keyboard and the rest of production would be more focused on the shape with little reference to the absolute location of each of the letter keys.

This analysis suggests a dynamic weighting scheme of the two channels; namely, adjusting the relative weight of the two channels according to the speed of the input gesture production. In general, open-loop recall movements are faster than closed-loop feedback-based movements. The gesture completion time should therefore be informative on how strongly the location channel should participate in the final selection of the target word.

However, the time to complete a pen gesture also depends on the length and complexity of the pen gesture. This issue can be addressed by understanding human action laws. The visual feedback based movement (tracing) can be viewed as a series of either pointing or crossing actions, both can be modeled using robust quantitative laws [1, 2]. The total normative writing time for a sokgraph gesture for a word w in this mode can be modeled using Fitts' law

$$t_n(w) = Na + b\sum_{k=1}^{N-1}\log_2\left(\frac{D_{k,k+1}}{W}+1\right) \qquad (10)$$

where $D_{k,k+1}$ is the distance between the k th and $k+1$th letters of word w on the keyboard, W is the key width, N is the number of letters in the word; a and b are two constants in Fitts' law. In the context of SKs, their values were estimated at $a = 83$ ms, $b = 127$ [29].

Once we have obtained $t_n(w)$ for each word and the total time of the actual sokgraph production, t_a, we can use them to modify the probability calculated from the location channel scores. In equation 7 we can use this transformation to substitute σ with σ_1 as follows:

$$\sigma_1 = \begin{cases} \sigma, \text{ if } t_a \geq t_n(w) \\ \sigma\left(1+\gamma\log_2\left(\frac{t_n(i)}{t_a}\right)\right), \text{ otherwise} \end{cases} \qquad (11)$$

For example, if t_a is 50% of $t_n(i)$, σ increases with 100γ percent. γ is an empirically adjustable parameter, expected to be between 1 and 10. In our current system we set $\gamma = 2.0$.

Note that this approach is more than simply adjusting the relative weight between the shape and location channels. It modifies the location channel probability of each *individual* word according to its path on the keyboard.

USING LANGUAGE INFORMATION

We have achieved quite satisfactory performance with the two core channels of recognition. However there may still be conflicting words even if both shape and location are considered, as suggested in Table 1. Most of these can be resolved by the language context (previous or surrounding words). It is also possible to give the user greater flexibility of sokgraph gesturing by taking advantage of language context. Using language models to improve performance is common, and in fact a key technology, in both speech and handwriting recognition.

Language Model and Integration

SHARK^2 uses a basic language model by default and can modify such a model by mining the user's email or other documents. We have explored several approaches to utilize a language model to improve the performance of the system. We currently use smoothed bigrams as the language model. A future extension of this model would be to use the trigram backoff method commonly used in speech recognition systems [10].

Once we have constructed the language model, we can use it to re-arrange the *N*-best list obtained from the classifier. Recall that the classifier outputs a confidence score $c(w)$ for a word w in the *N*-best list. Let w_p be the previous word entered by the user. We compute a language model score by transforming the bigram probabilities with a Gaussian function:

$$l(w) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{(1-P(w\mid w_p))-\mu}{\sigma}\right)^2\right] \qquad (12)$$

where $P(w \mid w_p)$ is the probability of w occurring after w_p. σ is a constant weighting the contribution of the language model. The motivation for using a Gaussian function is the heavily skewed bigram probability

distribution, and the desire to bring classifier confidence scores and bigram probabilities on an approximate common scale. By adjusting σ we can also tune the language context influence to be at the appropriate level.

The integration of the confidence scores and the transformed bigram probabilities is computed using Bayes' rule:

$$c'(w) = \frac{c(w)l(w)}{\sum_{i \in W} c(i)l(i)} \qquad (13)$$

where W is the set of words contained in the *N*-best list.

Statistical Text Stream

It is possible to evaluate the probability of the word stream entered by the user for a sentence segment (i.e. a sequence of words delimited by punctuation, *enter* key presses, etc.). The system may find and give the user the most likely stream among all possible paths formed by the *N*-best outputs for each word from other channels (shape, location, and bigram modeling). In other words, the system can take advantage of more global language regularities.

The search for this path can be efficiently performed using dynamic programming based on the products of the bigram probabilities. We adapted a version of the well-known Viterbi algorithm from continuous speech recognition [10] (pp. 244-251) to achieve this function.

A potential drawback in using global constraints is that the user may or may not like the fact that words already committed (displayed) by the system can change as the user writes new words. This is an effect similar to using T9 in mobile phones where the current part of a word may change after each subsequent key press.

Alternatively, we can take a more conservative approach. Rather than changing the committed stream to the most likely one, the system can highlight a stream when it deviates from the most likely path. The user may click on the highlight and choose other phrases. In other words, this may serve as a high level "spelling checker".

FEEDBACK AND OUTPUT INTERFACES

Designing effective feedback and output interfaces is critical to the overall usability of SHARK^2. This section presents some of the key components of the current SHARK^2 system in this regard.

Managing Visual Attention

Unlike ten finger touch typing on a typewriter keyboard, SHARK^2 (and all pen-based methods) requires visual attention. Because of this, the user would not be able to keep his attention on the application the text is written to and may not notice errors in the text output. It is therefore important to counter such a limitation.

A simple measure is to pronounce the word output through a speaker or earphone. We have experimented with such an option and found it quite effective. The privacy and disturbance drawbacks of this approach are obvious. The second option we explored was to overlay the word that was recognized directly on the SK, at the point of the last pen up position, i.e. where the user is most likely to be looking.

The third measure we explored is the use of a "text stream editor" above the SHARK^2 keyboard (see Figure 4). As the user writes new words, text continuously flows into the stream editor from the right and out to the left, and also goes to the application window with focus. Since the text stream editor is in close vicinity to the keyboard, one does not have to switch visual attention far away to confirm if the intended words are written. The user can also use this text stream editor area to edit text output by various means, including the "spelling highlight" function mentioned earlier, and selecting candidate words from an *N*-best list.

Figure 4. The stream editor showing a fragment of the sentence "Text editing made easy despite recognition errors". Gray words indicate that the user can press the stylus on them and reveal an *N*-best list of other possible candidate words.

Dynamic Visualization of Gesture Recognition

As a general rule of user interface design, a good UI always projects a clear conceptual model to the user on how the underlying system works. This is particularly a challenge for recognition based user interfaces. Users of speech and handwriting recognition systems are often perplexed by unexpected results. We made two attempts to expose the high level logic in sokgraph recognition, particularly as an option to novice users who are still learning to use the system.

The first is to communicate which sokgraph was selected by the system as matching the user's input gesture. This is performed in two different ways depending on whether the shape or location channel contributed the most to the recognition result. If the location channel had the most contribution the sokgraph is projected over the corresponding keys of the word on the keyboard layout, thereby emphasizing that location was the primary cue to the input. If the shape channel was the main contributor, the pattern is projected onto the bounding box of the user's pen trace.

Second, a visualization technique based on animating patterns or *morphing* was used to communicate how the user's gesture input is different from the ideal gesture. We observed that merely presenting the ideal sokgraph pattern does not expose the actual recognition process. Since SHARK^2 matches models (sokgraphs) using spatial similarity, a natural feedback mechanism is to visualize the spatial distance between a user's pen trace and the matched sokgraph. Using morphing, the system gradually transforms the user's gesture to the ideal sokgraph the system identified by linear interpolation. The segments of a user's pen trace that are the most far away from the ideal sokgraph has the largest and most noticeable motion, hereby communicating to the user where the major deviations from the ideal pattern lies (Figure 5). If an unintended word is recognized, the user can see how it happened. Both the on-keyboard word and the morphing trace disappear as soon as the user is no longer interested in the feedback information and puts down the pen for the next gesture.

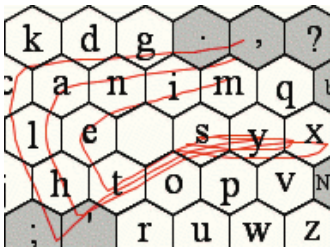

Figure 5. Three steps in the morphing feedback process overlaid on a single bitmap. The outer trace is the user's original pen trace. The inner trace is at the end of the morphing process where the user's trace has almost completely morphed into the ideal sokgraph.

N-best List Output

As shown earlier, there are occasionally words sharing the same sokgraph (e.g. *to* and *too*, *refuge* and *refugee*). Most of these are corrected automatically by the language model, but one cannot completely exclude ambiguity in a recognition system. Fortunately, a recognition system always has multiple candidates for each user input, ranked by confidence (*N*-best list). We have designed interfaces in SHARK^2 that give the user access to the next best choices. The primary interface is a linear menu containing the next best choices for a particular word. By holding down the stylus on a particular word, a translucent linear menu pops up. The user may select another candidate from the list, finalizing the decision by lifting the stylus. We implemented a linear menu instead of a pie menu, due to the fact that the hand tends to obscure half of the pie menu when the system is used on a touch-sensitive screen. A second technique is to simply let the user delete the word by a copy-mark gesture in the stream editor and write a new sokgraph.

SYSTEM IMPLEMENTATION

We have implemented all parts of the SHARK^2 system in Java (version 1.4.2). Our system has been tested on Microsoft Windows, including the Tablet PC version, Mac OS X, and various UNIX flavors running an X server such as Linux and Solaris. Naturally our system is most useful on a Tablet PC or in a PDA environment. A subset of the system (SK and an early version of the sokgraph classifier) has been ported to the Sharp Zaurus Linux PDA.

SYSTEM EVALUATION

Effective text entry is a non-trivial user interface problem. The evaluation of novel text entry systems is also complex due to the many dimensions on which text entry performance can be measured. These include not only speed and accuracy, but also the particular user's learning experience (see [28] for a review). Comprehensive human performance and learning evaluations, based on both real use and laboratory experiments, is beyond the scope of the current paper which focuses more on technology. Thus, this section presents only initial indications of SHARK^2's performance, both as a method and as a system.

In an early experimental study Zhai and Kristensson [27] demonstrated that users could learn about 15 sokgraphs per hour of practice, showing that although novel and artificial, users can learn, remember, recall, and produce sokgraph gestures. Equally important, users found the system interesting and fun to use, which is clearly important for a new writing system to be successful. These conclusions should still hold with SHARK^2.

Text entry speed with a new method is a function of practice and a matter of skill acquisition. Since the mean time to move a stylus from one key to another can be reliably modeled by Fitts' law, the eventual input speed with a SK can be confidently predicted based on letter digraph frequencies and Fitts' law [29]. Since SHARK^2 allows a degree of open-loop performance and uses a more fluid mode of motor movement, its writing speed can be potentially faster. Unfortunately there is not a human movement law that can reliably model sokgraph like gestures, despite the recent progresses in the study of "laws of action" as applied to user interfaces [1, 2]. Furthermore, with SHARK^2 the ease of entering text also depends on its lexicon size. The smaller the lexicon size, the more flexibility the system offers to the user for writing words within the lexicon.

Short of a theoretical prediction, we did some informal trials to *estimate* how fast a user could eventually write text with SHARK^2 after sufficient amount of practice. One simple method to simulate the eventual performance is to let users repeat phrases, so their learning of these specific words and sentences are saturated. Table 2 shows the "record" speed of the authors in such a scenario. Note that these are not typical average speed, but only indications of what SHARK^2 could potentially achieve as limits or possibilities. The lexicon used in the test, containing 7777 words, was mined from one of the authors' email, both sent

and received over a period of seven years. The numbers show the potential of the SHARK approach as a speed writing method, which are much higher than any existing pen-based method. As a reference, the expert performance using an optimized stylus keyboard has been theoretically estimated to about 45 wpm [29].

Testing phrase	User A	User B
The quick brown fox jumps over the lazy dog	69.0	70.3
Ask not what your country can do for you	51.6	60.0
East west north south	74.4	72.9
Up down left right	74.1	85.6

Table 2. Sample "record" speeds (wpm) with SHARK^2

Referring to the system design goals outlined in the introduction of this paper, the following results were achieved:

1. SHARK^2 can indeed handle all instances of word entry as sokgraph gestures or traces.

2. If a word is not in SHARK^2's lexicon, one can enter it by tapping. The system is extensible.

3. The system is real time. Due to the pruning techniques used, SHARK^2 can search 20,000 sokgraphs on an 800MHz PIII IBM Thinkpad in 20 to 40 ms.

4. The system is designed in accordance with the SHARK^2 paradigm, supporting gradual transition from visually guided tracing to recall driven gesturing. The system shifts its weight between the shape and location channel automatically according to the user's gesturing speed.

5. By incorporating a lexicon and a language model, the system maximizes input flexibility for the user. The system also has various feedback and output interfaces to allow the user to understand its mechanism and correct unintended text.

DISCUSSIONS AND CONCLUSIONS

We have made substantial progress towards making SHARK a practical writing approach. Zhai and Kristensson [27] demonstrated the feasibility of the SHARK approach by developing and experimenting with a small prototype system with dual modes of tapping and gesturing. In SHARK^2 we revised the original dual mode paradigm to a modeless and seamless skill transfer paradigm, in which the users' actions gradually shift from visually guided tracing to recall-based gesturing. We addressed the large vocabulary gesture recognition challenge by designing and developing a gesture recognition system that uses multiple channels of information—particularly shape and location to classify gesture input. In addition we have shown how to integrate a language model into the shorthand writing system to further increase recognition accuracy and relax the precision requirement.

Although we believe every aspect the SHARK^2 system can be further improved, the current system is the first complete system that makes writing with the sokgraph approach practical. Due to the novelty of the paradigm, we have relied on "prescriptive" algorithms based on use behavior analysis. In the future when a large user base is formed and training data become available, "data-driven" approaches such as neural networks can possibly make the system more adaptive.

It is interesting to compare SHARK^2 with traditional pen-based stenography with respect to speed and overall ease of use. SHARK^2 creates partially scale and translation invariant sokgraphs in a large vocabulary, similar to the word level shorthand symbols common in classic pen-based stenography for the most common words. However, in classic stenography a large part of the labor in high speed text writing is in transcribing the symbols to longhand text. In SHARK^2 transcription is achieved automatically. Leedham and Downton [12] created a computer system to automatically recognize and transcribe Pitman shorthand strokes. They note that one key problem, aside from the general difficulty in recognizing stenographs, was that the users may write too fast and too inaccurate to contain enough information for any recognition system. SHARK^2 has a clear definition of a sokgraph shape and a visualization method to inform the user on just how close or far away the gesture was from the recognized template, thus advising users not to push the system too far. In classic stenography the user has to invest large amount of time in learning even to begin using it. In SHARK^2 all words can be traced on a SK and the results are immediately displayed on the screen. A novice user only needs to find the letter keys on the keyboard in order to begin text writing. In this sense the SK works as a "training wheel" towards a new form of shorthand (sokgraph). The keyboard is also a mnemonic device that helps the user remember the sokgraphs.

From an information theory point of view SHARK^2 takes advantage of the information redundancy in a lexicon and a language model to relax the requirement of precisely specifying words verbatim. To write a word the user only has to make enough an effort to express the intention by the approximate shape and location of the word's sokgraph. The error tolerance in the system is inversely proportional to the size of the lexicon used. As a reference point, five consecutive letters have 26^5 or nearly 12 million permutations. Even 57,000 words, a large vocabulary size for any individual, is only a small fraction of all letter permutations in the usual word length. In practice, a SHARK^2 lexicon only has to contain enough words for an individual. In some sense, SHARK^2 is a step towards an optimal graphical coding scheme for text writing that interfaces the user's intention and the computer.

ACKNOWLEDGMENTS

The authors would like to acknowledge Sreeram Balakrishnan, Alan Cypher, Ronald Fagin, Maria Holmqvist, Thomas Moran, Wayne Niblack, John Pitrelli, Barton Smith, Jayashree Subrahmonia and Jingtao Wang for

their contributions to this work. Per-Ola Kristensson was a graduate intern at IBM Almaden Research Center (1/2003–9/2003, 6/2004-8/2004).

REFERENCES

1. Accot, J. and Zhai, S., Beyond Fitts' law: models for trajectory-based HCI tasks. *Proc. CHI 1997: ACM Conference on Human Factors in Computing Systems, 1997*, ACM, 295-302.
2. Accot, J. and Zhai, S., More than dotting the i's - foundations for crossing-based interfaces. *Proc. CHI 2002: ACM Conference on Human Factors in Computing Systems, CHI Letters 4(1), 2002*, ACM, 73 - 80.
3. Bellegarda, E.J., Bellegarda, J.R., Nahamoo, D. and Nathan, K.S. A Fast Statistical Mixture Algorithm for On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16* (2). 1994,1227-1233.
4. Bouchaffra, D., Govindaraju, V. and Srihari, S. A methodology for mapping scores to probabilities. *IEEE Transactions on Pattern Analysis and Machine Intellgence*, *21* (9). 1999,923-927.
5. Card, S.K., Moran, T.P. and Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey, 1983.
6. Duda, R.O., Hart, P.E. and Stork, D.G. *Pattern Classification*. John Wiley & Sons, New York, 2001.
7. Frankish, C., Hull, R. and Morgan, P., Recognition accuracy and user acceptance of pen interfaces. *Proc. CHI 1995, ACM Conference on Human factors in Computing systems*, *1995*, 503-510.
8. 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.
9. Goldberg, D. and Richardson, C., Touching-typing with a stylus. *Proc. INTERCHI, ACM Conference on Human Factors in Computing Systems, 1993*, ACM, 80-87.
10. Jurafsky, D. and Martin, J.H. *Speech and Language Processing*. Prentice Hall, New Jersey, 2000.
11. Kurtenbach, G., Sellen, A. and Buxton, W. An empirical evaluation of some articulatory and cognitive aspects of "marking menus". *Human Computer Interaction*, *8* (1). 1993,1-23.
12. Leedham, C.G. and Downton, A.C. Automatic recognition and transcription of Pitman's handwritten shorthand: an approach to shortforms. *Pattern Recognition*, *20* (3). 1987,341-348.
13. Long, A.C., Landay, J.A. and Rowe, L.A., Implications for a Gesture Design Tool. *Proc. CHI 1999, ACM Conference on Human factors in Computing systems, 1999*, 40-47.
14. MacKenzie, I.S. and Soukoreff, R.W. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, *17* (1). 2002.
15. MacKenzie, I.S. and Zhang, S.X., The design and evaluation of a high-performance soft keyboard. *Proc. CHI'99: ACM Conference on Human Factors in Computing Systems, 1999*, 25-31.
16. Mankoff, J. and Abowd, G.D., Cirrin: a word-level unistroke keyboard for pen input. *Proc. ACM Symposium on User Interface Software and Technology (UIST), Technical Note, 1998*, 213 - 214.
17. Newman, M.W., Jason, J.L., Hong, J.I. and Landay, J.A. DENIM: An informal web site design tool inspired by observations of practice. *Human-Computer Interaction*, *18* (3). 2003,259-324.
18. Perlin, K., Quikwriting: Continuous Stylus-based Text Entry. *Proc. UIST - ACM Symposium on User Interface Software and Technology, Technical Note, 1998*, 215 - 216.
19. Pitrelli, 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.
20. Plamondon, R. and Srihari, S.N. On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22* (1). 2000,63-84.
21. Rubine, D., Specifying gestures by example. *Proc. SIGGRAPH 1991: ACM Conference on Computer Graphics, 1991*, 329-337.
22. Schmidt, R.A. *Motor control and learning - A Behavioral Emphasis*. Human Kinetics Publishers, Inc., 1988.
23. Tappert, C.C. Cursive Script Recognition by Elastic Matching. *IBM Journal of Research & Development*, *26* (6). 1982,756-771.
24. Tappert, C.C., Suen, C.Y. and Wakahara, T. The State of the Art in On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12* (8). 1990.
25. Wobbrock, J.O., Myers, B.A. and Kembel, J., A High-Accuracy Stylus Text Entry Method. *Proc. UIST - ACM Symposium on User Interface Software and Technology, CHI Letters, 2003*, 61-70.
26. Zhai, S., Hunter, M. and Smith, B.A. Performance optimization of virtual keyboards. *Human-Computer Interaction*, *17* (2,3). 2002,89-129.
27. Zhai, S. and Kristensson, P.-O., Shorthand Writing on Stylus Keyboard. *Proc. CHI 2003, ACM Conference on Human Factors in Computing Systems, CHI Letters 5(1), 2003*, ACM, 97-104.
28. Zhai, S., Kristensson, P.-O. and Smith, B.A. In Search of Effective Text Input Interfaces for Off the Desktop Computing. *Interacting with Computers*, *16* (3). 2004,to appear.
29. Zhai, S., Sue, A. and Accot, J., Movement model, hits distribution and learning in virtual Keyboarding. *Proc. CHI 2002: ACM Conference on Human Factors in Computing Systems, CHI Letters 4(1), 2002*, ACM, 17-24.