Reducing Visual Demand for Gestural Text Input on Touchscreen Devices

I. Scott MacKenzie
Dept. of Computer Science and Engineering
York University
4700 Keele Street
Toronto ON M3J 1P3 Canada
mack@cse.yorku.ca

Steven J. Castellucci
Dept. of Computer Science and Engineering
York University
4700 Keele Street
Toronto ON M3J 1P3 Canada
stevenc@cse.yorku.ca

CHI'12, May 5–10, 2012, Austin, Texas, USA.
ACM 978-1-4503-1016-1/12/05.

Abstract
We developed a text entry method for touchscreen devices using a *Graffiti*-like alphabet combined with automatic error correction. The method is novel in that the user does not receive the results of the recognition process, except at the end of a phrase. The method is justified over soft keyboards in terms of a Frame Model of Visual Attention, which reveals both the presence and advantage of reduced visual attention. With less on-going feedback to monitor, there is a tendency for the user to enter gestures more quickly. Preliminary testing reveals reasonably quick text entry speeds (>20 wpm) with low errors rates (<5%).

Keywords
Text entry; gestural input; Graffiti; unistrokes; automatic error correction; visual attention

ACM Classification
H.5.2. [**Information Interfaces and Presentation**]: User Interfaces - *Input devices and strategies (e.g., mouse, touchscreen)*

General Terms
Human factors, Performance, Measurement

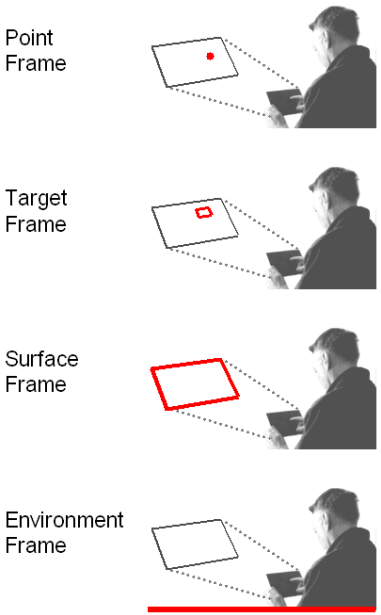

Figure 1. Frame model of visual attention. Progressing from bottom to top, increasing levels of visual attention are required.

Introduction

The recent proliferation of touchscreen mobile devices has generated considerable interest in gestural input via the fingers. Flicking, pinching, tapping and other gestures are now common in the repertoire of users' actions on these devices. Generally, such gestures serve to control the interface, typically by moving, sizing, or selecting a view or on-screen object. Another use of gestural input is text entry, where gestures produce textual symbols, such as letters, digits, or punctuation, or commands such as BACKSPACE, ENTER, or SHIFT.

This paper focuses on gestural input for text entry. We are interested in the distinct challenges presented by touchscreen devices as alternatives to devices with physical keyboards. Physical keyboards engage the user's tactile sense, providing critical information as fingers feel, engage, and press. The interaction is well known: Keys resist, then give way, and embark on their downward journey, dutifully informing the user that the intended *keypress* is complete. No similar experience exists with soft keys rendered on a touchscreen. And so, alternate sensory channels are engaged, typically visual, auditory, or both.

Touchscreens and Visual Attention

Besides the tactile sense, devices with physical keyboards create, through kinesthesia and proprioception, a sense of space and location. Users feel the tops and edges of keys, and groups of keys, and develop a sense of where their fingers are and the direction and distance to move to engage other keys. Again, no such experience exists for a soft keyboard on a touchscreen display. And there is little that visual or auditory feedback can offer to help. Users must look at the display to locate a destination key, then move the finger toward the key to select it. Visual attention is essential. This point has an interesting implication: Touchscreen mobile devices are, arguably, less mobile than their tactile counterparts. Mobility implies "on the move" and often multi-tasking. The added visual demand touchscreen devices impose on the user makes them less mobile in the sense that users are visually bound to the device while using it and, therefore, are less mobile.

In this paper we present a gestural text entry method that reduces the visual demand on users. The starting point is the *Unistrokes* handwriting recognition method [2], implemented using the *Graffiti* gesture set. Our approach is novel in that feedback about the on-going recognition process is not provided to the user, except at the end of a phrase. The rationale for this is explained below. We begin with a descriptive model for visual attention in terms of the frames of reference required for different types of interaction.

Frame Model of Visual Attention

There is more to visual attention than simply needing it or not needing it. There is a scale along which the required level of visual attention varies. To illustrate, we present a *Frame Model of Visual Attention*. See Figure 1. Four levels are shown, but the model could be re-cast with different granularity depending on the interactions of interest. The intent is to show a progression in the amount of visual attention required for different classes of interaction tasks. High demand tasks are at one end, low demand tasks at the other. By "demand", we refer to the amount or precision in visual attention, not to the difficulty of the task.

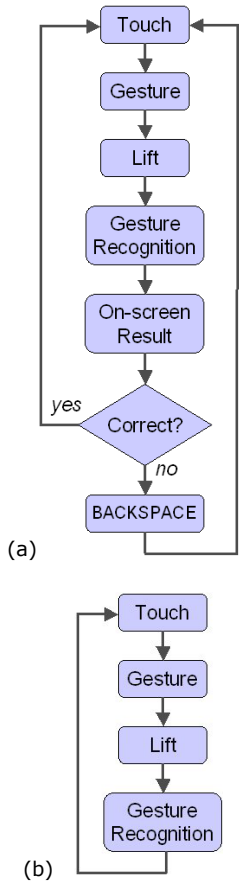

Figure 2. Gestural input (a) with inspection after each gesture (b) without inspection.

The *point frame*, at the top, requires the greatest demand in visual attention. Interactions in the point frame demand a high degree of accuracy and, consequently, require sharp central vision (aka foveal vision). The demand on visual attention is high. Examples in computing are tasks such as selecting a thin line or very small target, such as a pixel.

The *target frame* appears below the point frame in Figure 1. Interactions here involve selecting targets such as icons, toolbar buttons, or keys on a soft keyboard. Visual attention involving foveal vision is still needed, but with less demand than in the point frame. The targets are larger and, hence, slightly less precision and attention are needed.

The *surface frame* in Figure 1 applies to flicks, pinches, and most forms of gestural input on touchscreen devices. The user only needs to have a general spatial sense of the surface on which gestures are made. The visual demand is minimal; peripheral vision is sufficient.

The *environment frame*, at the bottom in Figure 1, includes the user's surroundings. Here, the frame of reference encompasses the user, the device, and the environment. Visual attention is not simply between user and device but with respect also to the user's surroundings. In most cases, the demand is low, and requires only peripheral vision. Some interactions involving a device's accelerometer or camera apply to the environment frame. Virtual environments may also apply here, depending on the task.

The frame model of visual attention offers insight into the problem of text entry on touchscreen devices, as explained the next section.

Reduced Feedback (= Reduced Visual Demand)
The underlying philosophy of *Unistrokes* or *Graffiti* is that each symbol is generated by a single gesture. Figure 2a illustrates. A gesture begins on touch and ends on lift. *Graffiti* gestures are spatially independent; thus, they may occur on top of one another or anywhere on the available surface. Although gestures fall within the surface frame of reference, as noted above, this is not necessarily true for gestures used for text entry. This is explained below.

A *Graffiti* gesture ends on finger lift. A recognition algorithm computes a set of features for the gesture based on the digitized sample points. The features are compared to features in a database and a symbol is produced as output. Hopefully, the result is correct. But, if the gesture was ill formed or incorrect in any way, the symbol may be wrong and may require a corrective action. This is shown in the last three steps in Figure 2a. The user visually inspects the recognized symbol and decides if the result is correct. If so, the next symbol may be entered. If not, BACKSPACE is entered to erase the errant symbol. Of course, entering BACKSPACE amounts to another full pass through the flowchart and does not contribute to the user's intended text.

With respect to the frame model of visual attention, the first three steps in Figure 2a fall within the surface frame. The last three fall within the target frame since the user's visual attention is on the small region of the display showing the recognizer output.

Figure 2b shows the same entry process, except the last three steps are omitted. Clearly there is less visual demand since the entire process falls within the surface frame. The result will be faster since there are fewer steps.

One final point: The decision box in Figure 2a is a two-choice reaction-time task and will take on the order of 250 ms [9, p. 62]. Thus, visual inspection adds about 250 ms to the time for each gesture. As a simple illustration of the effect, consider text entry at 24 words per minute. This is equivalent to 24 × 5 = 120 characters per minute or 2 characters per second or 1 character every 500 ms. Adding 250 ms to this yields an entry speed of 1 / (750/60000 × 5) = 16 words per minute. That's of 33% slower! Thus, the increased visual demand in monitoring the on-going recognition process comes a substantial performance cost.

Although users can choose any strategy they like (e.g., not looking at the display), our experience with text entry systems using gesture recognition (and, as well, word prediction) is that users generally choose to monitor the on-going recognition process. An idea we are pursuing in this research is to take this choice away — remove or hide the gesture-by-gesture feedback that users monitor during input. With this feedback removed, users should proceed more expeditiously since (a) there is no reaction-time task at the end of each gesture and (b) the interaction is fully in the surface frame. Of course, users inspect the results of gesture recognition for a reason – to observe if an error occurred. So, an additional idea we are pursuing is to handle recognition errors using automatic error correction. Users only see the result of their gestures at the end of a phrase. Hopefully, the result is satisfactory. Our automatic error correction algorithm is described next.

Automatic Error Correction

Automatic error correction is not new. At a simplistic level, the auto-correct feature on word processors qualifies: Type *adn* and the system converts to *and*. Specific algorithms have also been proposed and tested. Clawson et al. [1] developed *Automatic Whiteout++* for mobile phones. Their system corrects common errors during entry, such as hitting a neighboring key, character substitution, or transposition (*the* instead of *teh*). When tested on data from a mini-QWERTY experiment, the system corrected 32% of the errors automatically.

Kristensson and Zhai proposed an error correction technique using spatial pattern matching [3]. For example, entering *the* on a QWERTY keyboard forms a spatial pattern. If the user enters *rgw*, it is converted to *the* because the patterns are geometrically similar (and *rgw* is not in the dictionary). Pattern recognition was performed at the word level, when SPACE was entered. Overall, their system had a success rate of 83%.

Our automatic error correction method is intended for gesture input, so keyboard geometry is not a consideration. The algorithm works as follows.

Figure 3. Samsung *Galaxy Tab 10.1* running *Android 3.1*.

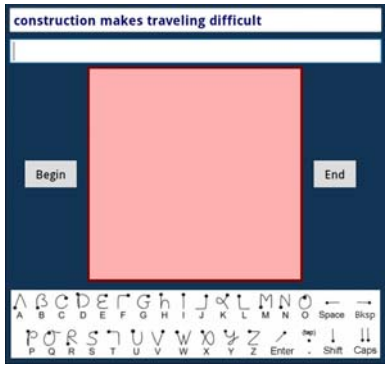

Figure 4. Experiment software.

Figure 5. Gestural input with the experiment software.

The incoming stream of gestures is processed as a series of inputted words (SPACE-delimited gestures). Each gesture in a word is deemed correct, incorrect (for a variety of reasons), or unrecognized. Unrecognized gestures remain in the inputted word as placeholders ("#"). This is important: For an unrecognized gesture, a character is intended, but the actual character is unknown. The inputted word is looked up in a dictionary containing words and their frequencies from a corpus. If the word is in the dictionary, it is left as-is. If the word is not in the dictionary, then the algorithm tries to correct the word as follows. The minimum string distance (MSD) [7] is computed between the inputted word (which is not in the dictionary) and all words in the dictionary. Three lists are produced: words with MSD = 1, MSD = 2, and MSD = 3. Each list is ordered by decreasing frequency. The three lists are concatenated. Then, all words of the same size as the inputted word are removed and put at the front of the list (again, ordered by decreasing frequency). The word at the front of the new list is selected to replace the inputted word. Note that depending on the word and the errors, the lists may be large, small, or empty. If all lists are empty, the inputted word is left as-is.

The dictionary used for testing is based on the British National Corpus and contains about 10,000 words [6].

The algorithm described above is similar to the automatic error correction algorithm described by Tinwala and MacKenzie [8]. There are a few differences, however. Their system provided "click" auditory feedback following each stroke and also spoke words through a text-to-speech service as each word was entered. Our system provides no on-going feedback (except for the digital ink tracking the finger path). Furthermore, as described below, our implementation defers feedback to the end of a phrase, rather than providing word-by-word feedback. The rationale for this is twofold: (a) to avoid the timely reaction-time task that occurs when recognizer results are inspected and (b) to reduce visual demand by limiting input to the surface frame.

Although the algorithm above is simple, its utility in use remains to be tested. Our initial testing is explained in the next section.

Initial Test

In-house *Graffiti* recognition software was integrated into a test program written in Java. The software was tailored to run on a Samsung *Galaxy Tab 10.1* running *Android 3.1*. See Figure 3. The interface presents phrases for input selected at random from a set of 500 phrases [5]. See Figure 4. User input is seen in Figure 5. Gesture recognition is not revealed to the user until the END button is tapped at the end of a phrase, whereupon a popup results dialog appears. See Figure 6. The example shows an entry speed of 21.4 wpm with an error rate of 12.0% in the (raw) transcribed text. The corrected text has an error rate of 0%. It appears three errors were committed: two strokes were unrecognized, one was misrecognized. All three errors were corrected by the algorithm. Of course the algorithm doesn't work as well in all cases.

As a further test, a user experienced with *Graffiti* entered 25 consecutive phrases with the test software. The mean entry speed was 21.7 wpm. The mean error rate in the transcribed text was 12.8% (min = 0%, max = 25.0%). For the corrected text, the mean error rate was 4.3% (min = 0%, max = 21.9%). See

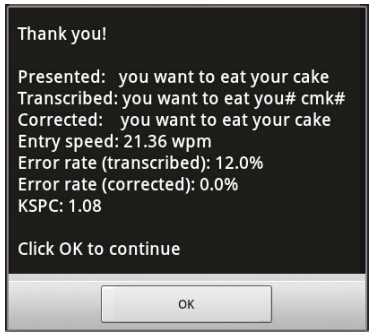

Figure 6. Results dialog for an example phrase. The recognizer produced the transcribed phrase. The automatic error correction algorithm produced the Corrected phrase.

Figure 7. Clearly, the automatic error correction algorithm worked quite well overall, fully correcting 12 of 24 phrases containing errors. The few cases were the algorithm did not do so well occurred where the SPACE stroke was misrecognized; this caused two words to be interpreted as one.

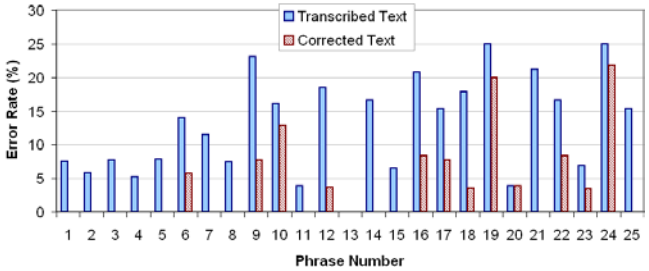

Figure 7. Transcribed and corrected error rates (%) for 25 consecutive phrases of entry.

Some examples of errant and corrected words are i#reeular → irregular, sa# → say, yu#ta → quota, custdo# → customs, t##k → took, tod → too, tog#ther → together.

One frequently-cited complaint about soft keyboards is that they consume screen space and obscure the underlying GUI [4]. This is unavoidable. Gestural text entry, as presented here, operates in the surface frame of reference. There is less visual demand. Although not implemented in our prototype software, the gestural input surface may be translucent, covering the entire display but without obscuring the underlying GUI. This will be the subject of further work.

References

[1] Clawson, J., Lyons, K., Rudnick, A., Iannucci, R. A., and Starner, T., Automatic Whiteout++: Correcting mini-QWERTY typing errors using keypress timing, *Proc CHI 2008*, (New York: ACM, 2008), 573-582.

[2] Goldberg, D. and Richardson, C., Touch-typing with a stylus, *Proc INTERCHI '93*, (New York: ACM, 1993), 80-87.

[3] Kristensson, P.-O. and Zhai, S., Relaxing stylus typing precision by geometric pattern matching, *Proc CHI 2005*, (New York: ACM, 2005), 151-158.

[4] Li, F. C. Y., Guy, R. T., Yatani, K., and Truong, K. N., The 1Line keyboard: A QWERTY layout in a single line, *Proc UIST 2011*, (New York: ACM, 2011), 461-470.

[5] MacKenzie, I. S. and Soukoreff, R. W., Phrase sets for evaluating text entry techniques, *Ext Abs CHI 2003*, (New York: ACM, 2003), 754-755.

[6] Silfverberg, M., MacKenzie, I. S., and Korhonen, P., Predicting text entry speed on mobile phones, *Proc CHI 2000*, (New York: ACM, 2000), 9-16.

[7] Soukoreff, R. W. and MacKenzie, I. S., Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic, *Ext Abs CHI 2001*, (New York: ACM, 2001), 319-320.

[8] Tinwala, H. and MacKenzie, I. S., Eyes-free text entry with error correction on touchscreen mobile devices, *Proc NordiCHI 2010*, (New York: ACM, 2010), 511-520.

[9] Welford, A. T., *Fundamentals of skill*. London: Methuen, 1968.