

A New Stylus-Based Method for Text Entry on Small Devices

Bryan Haggerty, Peter Tarasewich

HCI Laboratory, College of Computer & Information Science, Northeastern University
360 Huntington Avenue, 202 WVH, Boston, MA 02115 USA
{bhaggs, tarase}@ccs.neu.edu

Abstract

Ever-shrinking mobile device screens have made efficient text entry a substantial challenge. Many stylus-based methods have been proposed to tackle this problem, but most require a significant amount of already limited screen space. We present a novel stylus-based text entry method called *Touch-Point*, which does not require additional screen real estate or the use of complex gestures. The method features a variable-speed scrolling character set operated by the stylus. Characters are invoked by placing the stylus anywhere on a display, thereby allowing text entry in a space as small as a single character. In addition to the advantages *Touch-Point* can provide for small-screen devices, it can also be modified for use by disabled users who have limited motor capability and cannot operate a standard keyboard. Initial testing of *Touch-Point* against a gesture-based PDA method shows it to be a promising alternative to current stylus-based text entry methods.

1 Introduction

Technology is continually advancing in making devices smaller and more compact. This trend is very apparent in mobile devices, which range from personal digital assistants (PDAs) to phones to watches and even rings. Under circumstances that are often quite different from those where desktop computers are used, these ever-shrinking mobile devices require an efficient means of recording and accessing information. Since these devices can be taken anywhere, the user's situation can change rapidly from moment to moment. Given this "anytime/anywhere" environment, the ways in which humans interact with these devices are constantly changing.

People are now using mobile devices to speak with one another, make purchases, navigate roadways, listen to music, take photographs, and watch videos. But even with the abundance of multimedia content being transmitted with mobile devices, a great deal of information is still processed in the form of text. Examples include text messaging, maintaining a mobile calendar or address book, and mobile Web browsing (which can require keywords or a URL). Given this, appropriate and efficient methods for text entry on mobile devices must be available.

Using a stylus is a popular way of interacting with mobile devices such as PDAs. Methods such as gesture recognition (e.g., Graffiti or Jot), which require drawing representations of individual characters directly on the screen, have provided usable ways of entering text, but these methods still have drawbacks. The pitfalls of gesture recognition lie in the facts that 1) a user must learn which pen strokes represent a particular character, rather than the device interpreting the user's handwriting, 2) outside of the human interaction component, the device spends considerable processing time recognizing each character, which can hinder the efficiency of text input, 3) a special dedicated area where characters are drawn may be required on the device itself, claiming screen space that might be used for other purposes, and 4) the gestures themselves can require a great amount of precision to create, and may become more difficult to draw as screens shrink in size.

2 Background

Besides methods such as Jot and Graffiti, many other inspiring designs have been proposed for stylus-based text entry on small devices. One example of a stylus-based gesture recognition technique is EdgeWrite (Wobbrock, Myers, & Kembel, 2003). Users of this system enter text by traversing the edges and diagonals of a raised square overlay placed over the normal text input area of a PDA. Character recognition is accomplished not through pattern recognition but through the sequence of corners that are hit. Examples of EdgeWrite gestures are shown in Figure 1. EdgeWrite provides potential advantages to users with motor impairments that prevent them from using a stylus

with any degree of accuracy. Potential disadvantages include the need to learn the different gestures, and the room required to draw them.

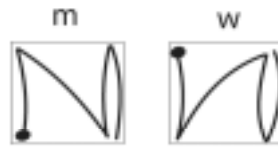


Figure 1: Example gestures For the EdgeWrite text entry system (Wobbrock et al., 1998).

Figure 2 illustrates the Cirrin System (Mankoff & Abowd, 1998). Cirrin is a word-level unistroke text input system. Users create a word by moving their stylus from letter to letter of the word without lifting the stylus from the input surface. Testing showed that Cirrin is about as fast as other existing pen entry systems. However, it still uses 26 cells to represent the English alphabet, which requires a significant amount of screen real estate to implement. Furthermore, given the small relative size of each cell, precise operation of the stylus is required to enter a word. As screen size shrinks, the movements between letters will become more difficult and potentially introduce more errors.

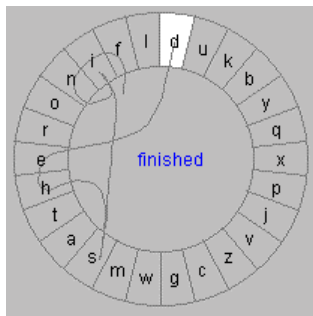


Figure 2: Using the Cirrin text entry system to enter the word “finished” (Mankoff & Abowd, 1998).



Figure 3: 5-button watch top text entry system (Dunlop, 2004).

The text entry system shown in Figure 3, intended for use on a watch face, has four alphabetic buttons and a central space button (Dunlop, 2004). It functions as a dictionary based disambiguation text entry system, similar to that of T9. For each character of an intended word, the key that contains the character is pressed once. The sequence of key presses is then compared to those stored in a dictionary. The space key is used to cycle through all possibilities and to commit the intended word. User studies have shown encouraging results that demonstrate the feasibility of such a system, but clearly, much of the watch face is occupied by the 5 buttons, leaving little room for the text itself.

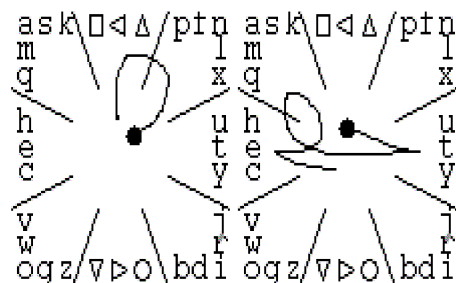


Figure 4: Using the Quikwriting text entry system to enter the letter “f” and the word “the” (Perlin, 1998).

Perlin developed a system called *Quikwriting* (1998), which combines the merits of the two previously mentioned systems. It functions similar to the watch interface, but instead of five physical keys, the system has nine virtual keys with multiple characters on a single key. A stylus is used to slide through the desired keys, similar to the Cirrin system. Every time the stylus slides across a virtual key, the key is recorded. The final key sequence is then matched to a pre-stored dictionary of words, as shown in Figure 4.

3 Touch-Point Stylus Method

We have invented a new stylus-based text entry method called *Touch-Point*. Touch-Point can be viewed as a variable-speed virtual wheel operated by the stylus and containing a scrolling character set. Continuous character sets have been used in other text entry methods such as the Date Stamp method's three-key input (MacKenzie, 2002) and thumbwheel methods (Tarasewich, 2003). The Touch-Point method is also a unistroke system. The stylus never leaves the surface of the input area until a character is entered.

With the Touch-Point system, the user can point to any location on the screen where text is desired. At that location, the first letter of a predefined character set appears on the screen (e.g., the letter "a"). A downwards motion of the stylus on the screen causes the character set to begin scrolling forwards (e.g., through "b, c, d, e") until the stylus is moved upwards to its original location. The further downwards the stylus is moved, the faster the characters scroll. Moving the stylus upwards will cause the character set to move backwards in a similar fashion (e.g., "z, y, x, w"). Characters are implemented as a continuous loop (e.g., moving past "z" might display "a"). The stylus is either lifted from the screen to commit the currently displayed character, or moved to the right to commit the character and start the next character. The stylus can also be placed on an existing character to modify it in the manner described above. Moving the stylus from right to left across an existing character will delete the character.

3.1 Implementing the Touch-Point Method

For the initial test of our Touch-Point method, we created a prototype with a text display area and a separate text input box. Because we were not sure how users would react to a continuously variable scroll speed, and did not yet know how best to implement the speed control, the scrolling motion was implemented in levels. The text input box is illustrated in Figure 5. Numbers appearing on the left of the box are for illustration purposes only and did not appear in the actual prototype.

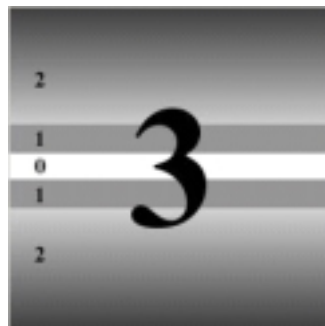


Figure 5: Prototype text entry area for Touch-Point system.

Scrolling control for the text box characters was divided into three zones. Zone 0 was the white stripe running through the middle of the box. Scrolling was stopped when the stylus was in this zone. When the stylus was moved into zone 1, designated by light grey stripes, the current character was shifted by exactly one character forwards or backwards in the character set. Zone 2 shifted characters forwards or backwards in a continuous motion, at a speed of between 4.8 and 7.1 characters per second (the further away from zone 1, the faster the speed). A character was committed when the stylus was lifted from the screen. Moving the stylus from right to left across the input box erased the last character in the text display.



Figure 6: The testing interface of the Touch-Point system.

The method was implemented on an HP H5550 iPAQ PDA, which is illustrated in Figure 6. The text display area was at the top of the screen, and the text input box was at the bottom. The Touch-Point input area was implemented in comparable proportions to that of a Jot character input area to control for possible effects resulting from input area size differences. This configuration is similar to the PDA's built-in Jot text entry software, which allowed us to perform a comparison test between the two methods. For our initial testing, we used a character set consisting only of the numbers 0 through 9. This was done to minimize the number of jot gestures that needed to be memorized in order to minimize the effect of any initial learning curve. The method was also implemented such that when a number was committed, the number remained in the test input box as the starting point for the next number.

3.2 Testing the Touch-Point Method

A study was conducted with the prototype in order to test the viability of the Touch-Point method and to compare text entry speeds and error rates against the PDA's existing Jot gesture based method. Each participant received an introduction on how to use the PDA and training on the Jot and Touch-Point methods. Participants were then asked to enter a sequence of 29 numbers by using both the Touch-Point and Jot methods. The same set of numbers was used for each participant. The numbers, shown in Table 1, were randomly generated and varied in length from 1 to 9 digits (with an average length of 4.1 digits). Half of the participants began with the Touch-Point method and the other half with Jot. The Jot gestures for numbers are shown in Figure 7.

Table 1: The set of numbers used as test input.

629882	724	94910848
518	242	95723933
27	84	73344294
76	9	59100739
540	595	845421090
304740	93	67400403
142397	135	4
2082	83851	9
44	27228	2
85	5501	











Numbers	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Figure 7: Jot gestures for numbers 0 through 9.

Participants were asked to correct all noticed errors, and to input the numbers as quickly and accurately as possible. Upon completion of the two tasks, participants were asked for their opinions on both text entry methods.

4 Results

Eight participants, all students from the research laboratories at our college, took part in the study. Only one participant used a PDA regularly, but had no experience with gesture recognition methods. Therefore, all participants were considered novice users for our purposes.

Table 2: Text entry and error rates for Touch-Point and Jot techniques.

		Touch-Point		Jot
		Overall	Final 9	
Text entry rate (WPM)		2.84	3.47	7.19
Error Rate (Errors per character)	Corrected	0.26	0.15	N/A
	Non-corrected	0.01	0.01	0.01
	Total	0.27	0.16	N/A

Table 2 shows the results of this testing. The first row shows the average text entry rate in words per minute (WPM), with a “word” being defined as five characters (in this case, digits) (MacKenzie, 2002). The first column is the overall text entry rate for the Touch-Point method, and the second column is the average for the last nine numbers only. The analysis of the final nine numbers is provided to give insight into how participants performed after becoming somewhat familiar with the method. The third column is the average entry rate for Jot. Row two gives the average number of errors corrected by each user, row 3 shows the errors that were not corrected, and row four is the sum of these rates. All error rates are reported in errors per character entered. Since Jot is a proprietary software package, it could not be modified to record the number of corrected errors. Participants stated that they preferred Jot to Touch-Point when entering text on the PDA, but said that under certain circumstances (e.g., on smaller screens), the Touch-Point method may be more practical.

5 Discussion

While non-corrected error rates are equal in both cases, the text entry rates for Jot were faster than those for the Touch-Point method. What is encouraging, however, is the fact that entry times increased and error rates decreased as subjects became familiar with the Touch-Point system. The rate for Jot is consistent with the 7.37 WPM reported by a study done by Sears and Arora (2001), although their rate reflects the entire Jot character set. The rate for the Touch-Point method is reasonably close to the 4.95 WPM for Graffiti reported by the same study (Sears & Arora, 2001). We did observe a fair amount of gesture recognition error occurring while participants used Jot during our study, but since this was not recorded we cannot compare it to the corrected errors rates obtained for the Touch-Point method.

6 Conclusions and Future Work

Our motivation for developing Touch-Point was creating a text-entry method that is suitable for very small devices, those that may not have screens large enough to support Jot, Graffiti, or any of the other current stylus-based methods. Our prototyping and testing was designed as a first step in testing the feasibility of the method. While the results are not particularly impressive from an input speed point of view, the Touch-Point method is still a viable text entry alternative for small devices.

The prototype and testing had definite limitations in terms of its shortened character set and relatively small number of participants, but was designed as an intermediate step towards the final implementation of the Touch-Point method. Next steps in this research include implementing a full character and punctuation set, adjusting the speeds of the character scrolling, and conducting large, long-term tests to get a more complete picture of how well the method works. The method might also be modified using linguistic models to achieve dynamic character reordering based on the user's input (e.g., predicting which character a user might choose next based on previously input characters, and making it the starting point of the next character selection).

On a device such as a PDA, with its relatively large screen, a gesture method such as Jot may always outperform Touch-Point, especially for experienced users. However, our intentions were not to create a method that beats Jot or Graffiti, but to create a method that 1) can be implemented on devices with very small screens and limited computing power, and 2) is intuitive and usable by novice users with minimal training. This research presents some first steps in achieving that ultimate goal. So while the method may not be a global solution, we hypothesize that for certain tasks, certain users, and for small screens, the Touch-Point method will provide advantages over existing methods. It can also provide an alternative interaction method to users, something that is exceedingly important on mobile devices where context and user needs can vary from moment to moment. Touch-Point most certainly needs to be tested on smaller devices (e.g., watches) to obtain a better understanding of its usefulness.

Since the Touch-Point method requires only axial motions (up/down, left/right, lift), the motions required for its use are much less complex than the other stylus-based options available. This makes it potentially useful as a text entry method for people with limited motor capabilities (e.g., those who cannot use a keyboard, but can hold a stylus-like device or use a joystick), as it can be translated to a larger scale (i.e., larger screen environment) easily. This aspect of creating an assistive technology that supports universal usability of information systems is being investigated.

References

- Dunlop, M. (2004). Watch-top text entry: Can phone-style predictive text-entry work with only 5 buttons? In *Proceedings of MobileHCI 2004*, (pp. 342-346).
- MacKenzie, I.S. (2002). Mobile text entry using three keys. In *Proceedings of the Second Nordic Conference on Human-Computer Interaction*, (pp. 27-34).
- Mankoff, J. & Abowd, G.D. (1998). Cirrin: A word-level unistroke keyboard for pen input. In *Proceedings of UIST 1998*, (pp. 213-214).
- Perlin, K. (1998). Quikwriting: Continuous stylus-based text entry. In *Proceedings of UIST 1998*, (pp. 215-216).
- Sears, A. & Arora, R. (2001). An evaluation of gesture recognition for PDAs. In *Proceedings of HCI International 2001*, (pp. 1-5).
- Tarasewich, P. (2003). Evaluation of thumbwheel text entry methods. In *Extended Abstracts of CHI 2003* (pp. 756-757).
- Wobbrock, J.O., Myers, B.A., & Kembel, J.A. (2003). EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of UIST 2003*, (pp. 61-70).