Gestures and Widgets: Performance in Text Editing on Multi-Touch Capable Mobile Devices

Vittorio Fuccella
Dipartimento di Informatica
Università di Salerno
Fisciano (SA), Italy
vfuccella@unisa.it

Poika Isokoski
SIS / TAUCHI
University of Tampere,
Tampere, Finland
poika.isokoski@uta.fi

Benoît Martin
UFR MIM / LCOMS
Université de Lorraine
Metz, France
benoit.martin@univ-lorraine.fr

ABSTRACT
We describe the design and evaluation of a *gestural* text editing technique for touchscreen devices. The gestures are drawn on top of the soft keyboard and interpreted as commands for moving the caret, performing selections, and controlling the clipboard. Our implementation is an Android service that can be used in any text editing task on Android-based devices. We conducted an experiment to compare the gestural editing technique against the widget-based technique available on a smartphone (Samsung Galaxy II with Android 2.3.5). The results show a performance benefit of 13-24% for the gestural technique depending on the font size. Subjective feedback from the participants was also positive. Because the two editing techniques use different input areas, they can co-exist on a device. This means that the gestural editing can be added on any soft keyboard without interfering with user experience for those users that choose not to use it.

Author Keywords
Gestures; Text editing; Caret movement; Clipboard; Android

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: User Interfaces

INTRODUCTION
The recent move to touchscreen-based user interfaces in mobile devices has satisfied user needs in many ways. Most importantly, the whole visible surface of the device can now be used for display which makes web browsing and reading in general much easier. However, some tasks may have suffered. One such task is text editing which can be implemented quite nicely on a physical mini-qwerty keyboard, for example, but becomes much more challenging to design for a touchscreen.

For example, precise positioning of the caret is easy with physical arrow keys. On the touchscreen this task becomes more difficult when the size of the text decreases. This, and other issues, could perhaps be addressed with gestures that can be used to move the caret and give other editing commands. In this paper we report on a design and evaluation of a gesture set for caret movement, text selection, and clipboard control.

Much work has been published on text entry in recent years. Some of this work has mapped gestures drawn on the display to characters in different ways [8, 9, 13, 14, 16]. Surprisingly, the interaction techniques used for text editing have not been researched much lately. As it will become apparent in the following sections, the development in the field has not been led by academically oriented scientific work. Instead device and software manufacturers have taken the lead. Text entry method industry has entered an unforeseen time of opportunity with the emergence of the Application markets that have been set up for all major mobile operating system platforms. The Android platform offers the easiest way to implement text entry methods and consequently the number of available alternatives is large. Since editing methods can be conveniently implemented as a part of the text entry system, Android was also our choice for the experimentation platform.

Fusion of different techniques can be observed in many recent text entry applications. The basic text entry technique is often the soft keyboard. It is amended with additional functionalities such as word completion and disambiguation, techniques to speed up text entry with gestures [14], etc. In this paper we amend soft keyboards with gestures for text selection and clipboard control. In our implementation of gestural editing the gestures are drawn directly on the soft keyboard. This way, the technique can be used with any application without interfering with the gestures used in the application for scrolling and zooming, for example.

In the traditional desktop computing environment, the editing actions can be performed in different ways: the caret can be positioned with a pointing device or with arrow keys; text selection is performed by dragging the pointer upon the desired text chunk or by using a combination of one or more control keys with the arrow keys. Clipboard operations can be executed through a WIMP-based interaction involving a click and a menu selection or, more efficiently, through keyboard shortcuts. These techniques are not available on touchscreen devices. Furthermore the pointing device (i.e. the finger) is not as precise for fine pointing as the mouse [7]. In essence, finger pointing is difficult because of occlusion. Furthermore the contact area between the finger and the touchscreen can be asymmetric on finger landings and lift-offs leading to unintended moves of the caret. Finger dragging is difficult be-

CHI 2013, April 27–May 2, 2013, Paris, France.

cause it involves relatively high contact friction. In most mobile devices, keyboard shortcuts are not available because there is no room for control keys. One of the aims of our gestural editing technique is to bring the strengths of the old arrow key-based editing familiar from desktop computing on multi-touch displays. Arrow keys are substituted with very simple directional gestures that are easy to remember, draw, and recognize.

The state of the art in touchscreen text editing is what we call the widget-based technique. The essence of this technique is that graphical widgets are added on top of the text to make the caret and the selection borders draggable and for launching menus that contain editing commands. The same general approach is used on all major platforms.

In the remainder of this paper we will first take a closer look at the related work. We will then describe the details of our implementation and the experimental evaluation of the gestural editing technique.

RELATED WORK

The use of gestures as shortcuts for executing actions has been investigated extensively in HCI research. Most of this work took place before the emergence of finger pointing and multi-touch gestures (e.g. [21, 17, 25]). Finger pointing is known to be faster than the stylus/mouse but less accurate [7]. Bragdon *et al.* [5] found that in the presence of environmental distractions, gestures can offer significant performance gains and reduced attentional load, while performing as well as soft buttons when the user's attention is focused on the phone. With the same amount of practice, users can recall more shortcuts and they make fewer errors with stroke shortcuts than with keyboard shortcuts [2].

However, in past research, the use of gestures performed directly on the soft keyboard has been mostly limited to text entry. These text entry methods map gestures to sequences of characters: entire words, as in Cirrin [16] and Shapewriter [14], or part of words, as in menu-augmented keyboards [9, 13].

Research on text editing had its high point in the early days of graphical user interfaces. The cut/copy-paste technique was developed over many years at Xerox PARC [22]. Text editors became more and more complex, leading to studies concerned with the new features [4, 18, 19, 20]. They were studied from the global perspective as parts of the computing environment. Learning to use them was one of the critical factors as all users were unfamiliar with this new technology.

From these early studies, we searched for methods of evaluating editors to apply in our experiment. The earlier work on basic editing tasks [11, 12, 20] relied on a protocol where printed documents with revision markings were used. The task was to do the revisions with text editors on the electronic version of the original document. Roberts' thesis [18] is perhaps the most detailed work on these issues. Eight core tasks were identified: insert, delete, replace, move, copy, transpose, split, and merge. These tasks emerge in the context of a character, word, number, sentence, paragraph, line, or section. We based our protocol on a subset of these tasks. However printed documents with editor's markings are rarely used nowadays especially in the mobile context. This is why we chose to present the editing tasks in an electronic document.

Figure 1. The magnifying lens on iPhone 4.

Other previous work focused on sequences of editing actions. The goal was to study users in larger editing tasks to evaluate their behavior and detect bottlenecks. In these studies the evaluation protocol was less strict. It was up to the user to figure out how to complete the task. For example Embley and Nagy [10] showed important differences between users and the difficulty to build the optimal sequence. The more complicated tasks in our task set were aimed at giving the participants similar freedom.

The early work mentioned above was in the context of desktop computers and terminals. Similar body of work on editing techniques on mobile devices is not available. Because of the shortage of scientific publications on editing techniques for mobile touchscreen devices we have to turn to the devices themselves and to the applications available for them to understand the state of the art.

Positioning and Moving the Caret

Usually coarse positioning of the caret is done by tapping with the finger. For fine positioning different direct interaction techniques exist. In Apple iOS a magnifying lens is displayed above the point of touch if the finger is held down for a while (see Figure 1). Then the finger can be moved and the magnified view is updated in real time. This gets around the occlusion problem and makes it easier to see the position of the caret even if the text is very small. In the Samsung's implementation of Android 2.3 that was used in our experiment, a graphical widget is attached to the lower end of the caret (see Figure 2). This widget can be dragged. This technique avoids occlusion but does not offer the magnification.

Some soft keyboard products support caret movement by the means of arrow keys on the soft keyboard. The Hacker's Keyboard[1] and the Arrows Keyboard[2] have the arrow keys visible all the time. The Thumb Keyboard[3] lets the user choose whether they are visible or not. The Perfect Keyboard[4] employs a gesture on the space bar to show and hide the caret movement keys. In Swype[5], TouchPal[6], and the Custom Keyboard[7] the arrow keys are available in an alternate layout that the user can switch to when needed. The caret movement in

[1] http://code.google.com/p/hackerskeyboard/
[2] http://arrows-keyboard.android.informer.com/
[3] http://www.beansoftapps.com/?products=thumb-keyboard
[4] http://androidperfectkeyboard.blogspot.fr/
[5] http://www.swype.com
[6] http://www.touchpal.com
[7] http://www.android-spa.com/customkeyboard/

the Custom Keyboard is implemented with a virtual touchpad. An alternate layout adds an extra step in the interaction and tends to make editing cumbersome decreasing the potential advantage that users could gain from the arrow keys.

Using gestures on the soft keyboard for supporting caret movement has also been implemented. The Smart Keyboard[8] allows user-configured functions to be associated with up, down, left, and right gestures. The Gesture Keyboard[9] allows left and right gestures anywhere on the keyboard surface. Short gestures move the caret and long gestures generate backspace and space. In MessagEase[10] sliding left or right from the space-key moves the caret left or right by one letter. Slide-and-return moves the cursor by one word. In all cases, auto repeat is available by holding the finger down after sliding. Moving the caret is also possible by sliding left from the a-key, right from the i-key, up from the i-key and down from the s-key.

On the iOS the text entry systems are more difficult to modify than in Android. However, there are at least two tweaks SwipeShiftCaret[11] and SwipeToMoveCursor[12] that connect gestures drawn outside the keyboard to left and right cursor movements. Some applications also implement editing gestures. For example, Tyype HD[13] text editor implements multi-touch gestures for text manipulation. Two-finger double tap selects all the text. One-finger horizontal slide moves the caret. Two-finger horizontal slide is used to select text from the current position. Three-finger horizontal slide moves the caret at high speed. Apple has also patented finger gestures on the soft keyboard already in 2008 [23].

Clearly, many developers have seen problems in caret positioning. However, we have not found data on the performance of gestures in this context. Through our evaluation we contribute this missing information.

Selecting Text

There are multiple ways to implement text selection on touchscreens. One could press and slide the finger over text to select it. One could hold the finger down on a word to select it. A double or triple-tap could select a word or perhaps a whole line or sentence.

The Samsung Galaxy II phone with Android 2.3.5 that we used in our experiment used a widget based technique to select text as shown in Figure 2. When the user tapped on the widget under the caret a menu popped up. In the menu there was a command ”select word”. After selecting the word, both ends of the selection had a widget hanging from them as shown in Figure 3. The user could then drag these widgets to move the ends of the selection.

The only application independent example we know on gesture-based selection adjustment is in MessagEase. The gesture to initiate a selection is a clockwise circle on the ”123” button. It causes a ”select all” command. When a selection is active, fine-tuning it is done by moving independently the end points of the selection by sliding the finger on the space key. Movements initiated on the left part of the space key move the beginning of the selection and movements initiated on the right end of the space key move the end of the selection. As previously, slide-and-return moves by one word and a repeating effect is possible by holding the finger down after sliding.

[8]http://www.dexilog.com/smartkeyboard/wiki/Smart_Keyboard
[9]http://gesture-keyboard.android.informer.com/
[10]http://www.exideas.com
[11]http://www.iphone4.fr/swipeshiftcaret/
[12]http://www.iredsnow.net/download-swipetomovecursor-0-1-1/
[13]http://itunes.apple.com/us/app/hyype-hd/id527863008

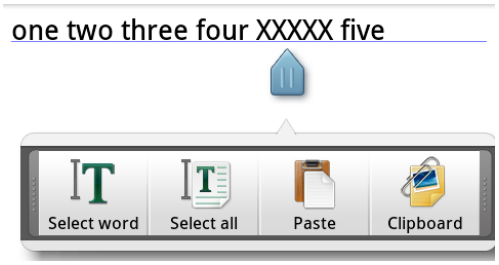


Figure 2. The editing widget and menu in Samsung Galaxy S II (Android 2.3).

Figure 3. Text selection, the adjustment handles, and the associated menu on Samsung Galaxy S II (Android 2.3).

Clipboard Control

The ability to re-structure text by copying and pasting parts of it is characteristic to writing with computers. In touchscreen editing these functions are typically available in a menu that pops up when text has been selected. In the Samsung’s implementation of the Android platform that we used in our experiment the menu popped up, stayed on the screen for a few seconds, and then disappeared taking the handles with it. Being too slow in choosing the command led to the need to redo the text selection. Text selection and the menu are shown in Figure 3.

At least one example of gestural methods of clipboard control exists. It is the Command Strokes by Kristensson and Zhai [15] where the Shapewriter system included the possibility to write the names of commands to launch them.

DESIGN AND IMPLEMENTATION OF GESTURAL EDITING

There are many ways to implement the idea of gestural commands for editing. Our implementation is described in this section.

Gesture-Operation Mapping

The gestures we used are shown in Table 1. The last column of the table shows the shape of the gesture. The gesture begins from the dot, continues through the line and ends at the arrowhead. One arrow means an one-finger gesture and two arrows mean a two-finger gesture. The central design

Name	Category	Actions	Shape
Left	CP/Sel	Moves the caret/selection endpoint one char left	
Right	CP/Sel	Moves the caret/selection endpoint one char right	
Up	CP/Sel	Moves the caret/selection endpoint one row up	
Down	CP/Sel	Moves the caret/selection endpoint one row down	
2Left	Sel	Moves selection endpoint one word left	
2Right	Sel	Moves selection endpoint one word right	
2Up	Sel	Moves selection endpoint to the beginning of the text	
2Down	Sel	Moves selection endpoint to the end of the text	
Copy	CC	Copies the selected text to the clipboard	
Cut	CC	Cuts the selected text and copies it to the clipboard	
Paste	CC	Pastes the text from the clipboard	

Table 1. The gesture-operation mapping used in our design. Each gesture was associated with action(s) from three categories: Caret Positioning (CP); Selection Manipulation (Sel); Clipboard Control (CC).

goals were to minimize the number of gestures and to keep them as simple as possible. For example, instead of having multiple caret movement commands for different distances to move, we opted for a simple gesture that can easily be repeated many times if necessary. The small number of gestures also helps in learning.

Keyboards often have the auto-repeat function which is an advantage in repetitive tasks such as moving a long distance using a character-by-character movement commands. As described above, similar function can be included in gestures through a hold at the end of the gesture. However, this leads to questions about the spatial and temporal thresholds for initiating the auto-repeat as well as questions about the appropriate rate of repeat. Both parameters are probably user-dependent. We chose not to implement auto-repeat. Instead we had the word-level and text-level selection/movement commands in our gesture set for rapid selection of large chunks of text.

Our set consists of three categories of gestures:

- *Caret Positioning*: The caret can be moved in the one of the four directions (one character left or right, one row up or down) by sliding the finger in the corresponding direction;

- *Selection Manipulation*: A selection is always started from the current position of the caret with a two-finger gesture. The whole word on the left (or right) of the cursor can be selected with a horizontal gesture. The whole text preceding (or following) the caret can be selected by sliding two fingers vertically. It is possible to adjust the selection by repeating the selection gestures or by using the one-finger caret movement gestures. Only the ”free” end of the selection, i.e. the end where the caret was not, can be moved. To reset the selection, the user can tap anywhere on the text.

- *Clipboard Control*: Three clipboard operations are supported: cut, copy, and paste. The gesture shapes resemble the characters on the keys used for the same purpose in Microsoft Windows (’X’, ’C’ and ’V’).

This gesture set was developed iteratively over a number of months of tests and discussions. Earlier designs involved features that were later abandoned. For example, single and double swipes were both used for cursor movements and for selecting text: two-finger gestures were also used to move the caret and a selection could also be started through a one-finger gesture. A selection mode was activated with the SHIFT key. These designs were revised for several reasons. The most important was that the users could not easily perceive the state of the SHIFT key and this led to confusion regarding the modes. Furthermore, we realized that the fastest way to place the caret is by tapping on the text and then adjusting its position through single character movements. This made many of the long distance movement commands unnecessary.

Gesture Recognition

Our gesture recognition worked in stages. First, gestures were distinguished from taps by the length of the stroke. The discrimination threshold was set according to be clearly smaller than the width of the smallest key. Next, in the recognition process the number of pointers discriminated between one-finger and two-finger gestures. Before performing the recognition, the two-finger gestures were reduced to one-finger gestures by removing the shortest stroke. This way, the recognition of our multi-touch gestures could be performed through a unistroke classifier. For classification we used a nearest neighbor classifier derived from the 1$ unistroke recognizer by Wobbrock *et al.* [24]. The system remembered the original two-finger status and adjusted the recognition result accordingly after the classifier was done.

There were two important differences between our implementation and the original 1$ recognizer. First, our gestures were not rotation invariant. Second, we had one-dimensional gestures that would lose their direction if a non-uniform scaling is applied. Therefore, the scaling in our classifier was uniform.

For the majority of the gestures, a single template per class was used. Only for the most elaborate gestures in the *Clipboard* category, two templates were used. Furthermore, since we noticed a bias of the recognizer in favor of the straight gestures, due to an easier correct execution, the (*Best Angle*) step in the recognition process was only performed with the gestures in the *Clipboard* category and a factor less than one was used to reduce their distance.

To verify that the gesture recognizer did its job as expected we tested it against a publicly available gesture set. The SIGN database [1] contains 34,000 handwritten unistroke gestures belonging to 17 different classes, drawn by 20 users on a Tablet PCs. Since our two-finger gestures were reduced to unistrokes before recognition and since the recognition of the two-finger gestures in the four directions was trivial, we found this database useful for a preliminary sanity check of the recognizer. The database was adapted for our purposes: only the samples belonging to the classes used in our gestural

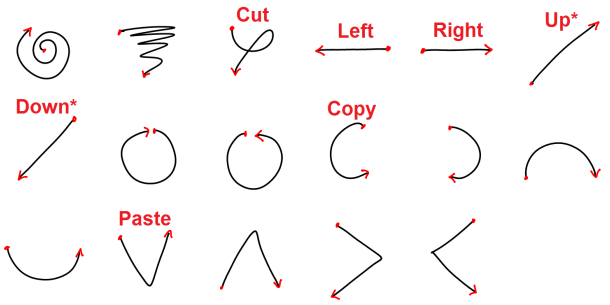


Figure 4. Sample gestures from the SIGN database. The named gestures were used in the evaluation of our recognizer. Diagonal gestures (*) were rotated by 45 degrees counterclockwise to obtain vertical gestures.

editing implementation were selected for the test. Furthermore, since the database did not contain vertical gestures, the samples for our *Up* and *Down* classes were obtained by artificially rotating the diagonal gestures contained in the database. Figure 4 shows a sample gesture for each class contained in the database and the class of our editing gestures where it was assigned.

A total of 13,653 samples were used in the recognition test. The results of the classification showed an accuracy of 99.76%. While this test was performed with stylus-drawn gestures and no real two-finger gestures were present, it helped us to ascertain that the recognizer was probably good enough to serve in our experiment. We would have been satisfied with anything above 95% accuracy.

EXPERIMENT

As it is apparent in the related work section above, we were not alone in believing that gestures may make editing more pleasant and perhaps more efficient on touchscreen devices. However, we have not seen empirical evidence of this. This is why we ran an experiment to record users' subjective impressions and performance in editing tasks.

Design

The experiment was a two-factor within-subjects design. The factors were the font size (1.75, 3.25, and 4.75 mm) and the editing technique (gestural and the widget based technique that comes with Android 2.3 on Samsung Galaxy II phones). The font sizes were selected by picking a comfortable font size for the phone model in question and then a smaller and bigger at 1.5 mm intervals.

The primary dependent variable was the task completion time. In addition it was possible to compute a variety of other measures that may correlate with perceived ease of use such as the number of gestures, text selection events, key presses, etc. Some of these are reported in the results section.

To record subjective impressions we used the System Usability Scale (SUS) [6] that was filled in for both editing techniques. The questionnaire is composed of 10 statements to which participants assign a score indicating their strength of agreement in a 5-point scale. The final SUS score ranges from 0 to 100. Higher scores indicate better perceived usability. In addition we made notes of participants' comments during and after the experiment. We also collected background information including whether the participants owned and operated touchscreen devices and whether they edited text on their devices.

Participants

We recruited a total of 12 participants among the staff and students of our universities in Finland and France. Five of the participants were female and seven were male. The ages ranged from 25 to 52 (M=34.7, SD=7.8). All participants had some experience with touchscreen devices. Eight participants had mobile phones with touchscreens (four iPhones and four other devices). In addition one participant owned an iPad, but reported that the only editing task she used was entering URLs to a web browser. Interestingly, three participants did not yet own or operate a touchscreen device although all were mobile phone owners.

When asked about text editing habits with the touchscreen devices, the participants who used touchscreen devices typically mentioned using backspace to erase text and tapping on the text to move the caret if the error was far away from the current position. Use of copy, cut, and paste commands was rare and limited to copying URLs to the web browser and sometimes copying the whole message in order to construct the reply around the quoted material.

All participants reported their skill in the English language to be at least good (on the scale of poor, good, near native). Note that while the tasks were presented in the English language, only the last task required language skills beyond knowing the order of small numerals.

Apparatus

The experimental software had two parts. The first part was an Android text entry service that implemented the QWERTY soft keyboard with the ability to interpret the gestural commands listed in Table 1. This soft keyboard was constructed based on the text entry method example delivered with the Android Software Development Kit. An Android *toast* (a message shown for a short time in a small window) appeared superimposed on the keyboard after each clipboard operation. It was used to give a positive or negative feedback on the success of the operation.

The second part of the experimental software was a text editor whose initial screen presented a task list. The participants started a task by selecting it from the list. The text editor underlined the erroneous passages that needed editing and displayed a message when the editing task had been completed. Besides logging input events, the editor also automatically measured the duration of a task from the time when the task was shown to the time when it was completed. The software determined the completion status by comparing the edited text to the correct solution.

The phone model was chosen because of the Android operating system, sufficient processing and storage capabilities and the widespread availability[14]. The display had a 480x800 pixel resolution and measured approximately 110 mm in diagonal. The multi-touch sensing was capacitive.

[14]Internet sources seem to agree that about 20 million phones were sold in ten first ten months of availability

Figure 5. A screen shot of the Samsung Galaxy II display during an editing task.

The editing tasks used in our experiment are listed in Table 3. Earlier work did not always report the tasks used in the experiments. We would prefer to have a standard task set to use in editing experiments. Table 3 is included in this paper in full to support the development of such a set.

The choice of tasks should ideally be based on statistics on the frequency of editing actions that occur in real-life situations. Lacking such statistics, we based our design on the inclusion of the widest possible range of editing situations without making the experimental sessions too time consuming. We chose the editing operations on the basis of a previous research on the use of gestures in text editing [25]. There was also a significant overlap with the basic tasks by Roberts[20] as almost all operations were included in our tasks. Furthermore, character insertion and deletion operations were considered both individually and when included in a series. We also designed the tasks to let the user exploit all of the gestures provided by the gestural editing technique.

Although all tasks contained more than one type of interaction, it is useful to divide the task set into subsets according to the dominant type of interaction to aid in the analysis of the performance results. This made it possible to relate possible performance differences to the types of interaction. In our analysis of the tasks we ended up with three sets. The first set was dominated by keyboard use (tasks 2, 3, 7, and 8), the second set was dominated by caret movement (tasks 1, 4, 5, 6, 15), and the third set is dominated by text selection and clipboard use (tasks 9, 10, 11, 12, 13, and 14).

Procedure

Each participant participated in one session lasting about 1.5 hours. In the beginning of the session each participant was given a sheet of paper. One side of the paper contained instructions in the use of the editing techniques including a table of the available gestures. The instruction sheet remained visible during the experiment. The other side of the sheet contained an informed consent form that the participants signed before the experimental tasks began.

The experiment consisted of eight blocks. In each block, the participants had to complete all of the the previously described fifteen tasks. The participants were allowed to rest as long as they wanted between tasks. Longer breaks were taken between blocks while the experimenter was setting up the next block. Four of the blocks were completed using the widget-based technique and the remaining four blocks were completed using the gesture-based technique. Half of the participants did the four gesture blocks first and then the four blocks with the widgets, while the rest reversed the order of the techniques. The first block of each editing technique was completed using the medium (3.25mm) font size. It was aimed at training the use of the editing technique and at familiarizing the participant with the tasks. The three blocks after the initial training block varied in the font size. This way, we completely counterbalanced the order for the two factors, technique (widget, gesture) and font size (small, medium, large), obtaining 12 different permutations for 12 users, as summarized in Table 2.

Participant	Order of Font Sizes		Order of Techniques
	Training	Blocks	
1	m	s-m-l	gesture-widget
2	m	s-l-m	gesture-widget
3	m	l-s-m	gesture-widget
4	m	l-m-s	gesture-widget
5	m	m-s-l	gesture-widget
6	m	m-l-s	gesture-widget
7	m	s-m-l	widget-gesture
8	m	s-l-m	widget-gesture
9	m	l-s-m	widget-gesture
10	m	l-m-s	widget-gesture
11	m	m-s-l	widget-gesture
12	m	m-l-s	widget-gesture

Table 2. The counterbalancing scheme used in the experiment. The font size is reported abbreviated (s=small, m=medium, l=large).

After completing all blocks with one editing technique, the participants responded to the System Usability Scale concerning that technique. At the end of the experiment we asked which technique they would prefer if they had to make the choice. There was room for free-form feedback at the end of the SUS form and we kept notes on the comments the participants made during the experiment and during the final debriefing.

RESULTS

Aggregate Results

Task completion times are shown in Figure 6. The gestural technique exhibited lower means with all font sizes. A repeated measures ANOVA showed that the difference between the editing techniques was statistically significant ($F_{1,11} = 24.1, p = 0.0004$). Figure 1 also suggests an overall effect of font size. This too was statistically significant ($F_{2,22} = 26.9, p < 0.0001$). The interaction of these factors was also statistically significant ($F_{2,22} = 7.08, p = 0.004$). The interpretation of the interaction based on Figure 6 is that the widget-based editing technique suffered more of the small font size.

Task	Title	Instruction	Presented form	Correct form
1	Delete Character	Delete the X character in the sentence	one two thrXee four five	one two three four five
2	Delete Word	Delete the X characters in the sentence	one two three four XXXXX five	one two three four five
3	Delete Phrase	Delete the incorrect phrase in the sentence	one XXXXX XXX two three four five	one two three four five
4	Delete Characters	Delete the X characters in the sentence	oneX XXXXX two thrXee fouXXXr Xfive	one two three four five
5	Insert Character	Insert a space in the sentence	one two threefour five	one two three four five
6	Insert Characters	Insert spaces in the sentence	onetwothreefourfive	one two three four five
7	Insert Word	Insert the correct word in the sentence	one three four five	one two three four five
8	Insert Phrase	Insert the correct words in the sentence	one four five	one two three four five
9	Move Word	Move a word to restore the correct order	one three two four five	one two three four five
10	Move Word 2	Move a word to restore the correct order	one three two four five	one two three four five
11	Move Phrase	Move the words to restore the correct order	one four five two three	one two three four five
12	Move Line	Move a line to restore the correct order	one one one one three three three three two two two two four four four four five five five five	one one one one two two two two three three three three four four four four five five five five
13	Move Lines	Move the lines to restore the correct order	one one one one four four four four five five five five two two two two three three three three	one one one one two two two two three three three three four four four four five five five five
14	Complete Text	Fill in the missing text	one one one one two three three three three four five five five five	one one one one two two two two three three three three four four four four five five five five
15	Correct Errors	Correct the misspelled words	Twenty years form now you will be more disappointed by the thXings you didn't do than by the ones you dd. So throw off the bowlines, Sail away from the safe harborX. Catch the trade winds in oyur sails. Explore. Drem.	Twenty years form now you will be more disappointed by the things you didn't do than by the ones you did. So throw off the bowlines, Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream.

Table 3. The editing tasks. Spotting the errors was easier in the experiment than it seems here because the spell checker underlined words that did not match the corrected form.

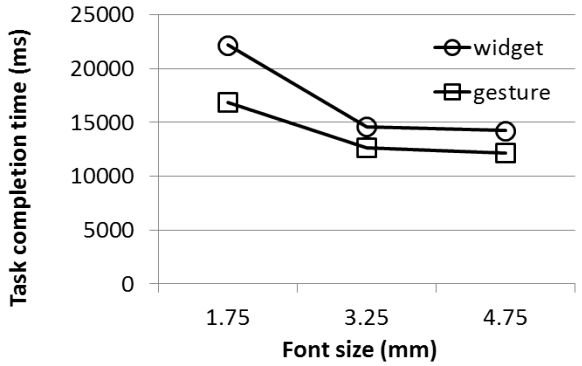


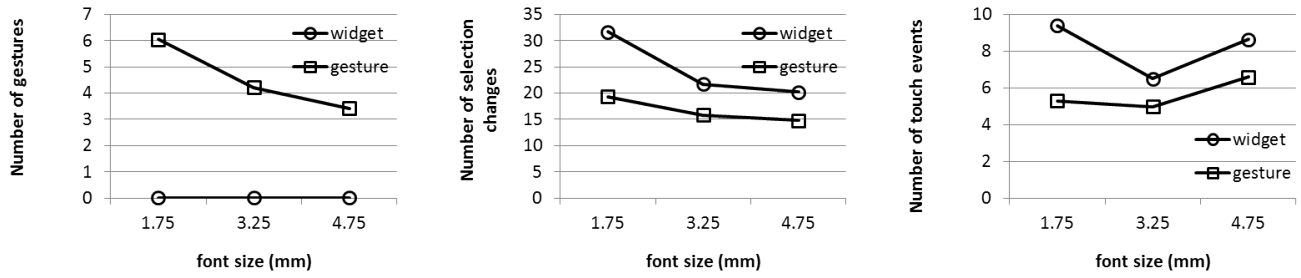
Figure 6. Comparison of the task completion times obtained in the two different designs under three different font sizes.

Because both editing techniques were available throughout the experiment, we needed to verify that the participants used them only when instructed. The number of gestures entered can be seen in sub figure *a* in Figure 7. Only a negligible number of gestures was recorded in the widget condition. Still the possiblity of confounding transfer effect in the transition between the techniques remains. We re-ran our ANOVA with the order of the techniques as an added between subjects factor. The order had no statistically significant effects or interactions indicating that transfer effect, if present, was not statistically significant.

Sub figures *b* and *c* in Figure 7 give us an indication of why the widget interface was slower. The reason appears to be that the participants had to do more work with the widget interface. The larger number of user-initiated events is most likely due to problems in utilizing the widgets. If the caret was incorrectly placed, the placement action (tap of a finger or drag of the widget) needed to be repeated until the caret landed in the desired location. The user's comments supported this interpretation. Especially with the small font size the caret tended to move when they lifted their finger from the widget. Similarly it tended to move when they landed their finger on the widget to launch the menu. They found this quite frustrating and felt liberated when the gestural interface did not exhibit the same problem.

User Satisfaction and Free-form Feedback

The mean SUS score for the widget interface was 57.5 ($SD = 20$). The mean SUS score for the gestural interface was 82.3 ($SD = 8.6$). This value can be taken as a very good level of satisfaction, especially considering that cellular phone interfaces are generally rated low, compared to other interface types [3]. A Wilcoxon matched-pairs signed-ranks



(a) Gestures were used only in the gestural mode as instructed.

(b) Number of selection changes was higher for the widget technique.

(c) Number of touch events (re-positioning of the caret) was higher for the widget interface.

Figure 7. Comparison of the average number of events logged per participant per editing task in the two editing techniques. Note that the numbers do not match exactly to user actions because some user actions generate more than one event in the system. However, the difference between editing techniques is meaningful.

test showed that the difference was statistically significant ($Z = 2.81, p = 0.005$). The standard deviations show that there was more disagreement on the widget interface. The highest SUS score for it was 85 and the lowest was 25.

When asked to choose their preferred editing technique, one participant was undecided. All others preferred the gestural technique. Several participants suggested that the optimal editing technique would perhaps contain features from both techniques. Typical comments from participants were complaints about the widget technique including the difficulty of selecting from the beginning of the line and the difficulty of placing the caret accurately with the small font size. The need for the undo function was often mentioned regarding both techniques. Comments on the gestural technique included suggestions for changing the semantics of the gestures such as to add a gesture for selecting all text till the end of the line.

Task-Specific Results

As mentioned earlier, the tasks can be divided into three sets according to the dominant form of interaction. In this section we present the performance results according to this division.

Keyboard Dominated Tasks

The mean task completion times with the medium font size are shown in Figure 8. The grouping of the data in the figure corresponds to the order in which the results are discussed here. Some of the differences in advantage of the gestural technique were greater with the small font size, but we display the results for the medium font because they are more relevant in practice.

Tasks 2, 3, 7, and 8 were classified as keyboard dominated. The soft keyboard used for text entry and erasing of text using the backspace key was the same with both editing techniques. Therefore we did not expect large differences in keyboard dominant tasks. Each task required at least one caret placement to start the deletion or insertion of text. Differences in the ease of caret placement could, therefore, cause minor differences in efficiency.

The results matched our expectations. The differences between the techniques were small. The gestural interface was faster except for tasks 3 and 8 with the *large* font size. These tasks involved the deletion (task 3) and insertion (task 8) of

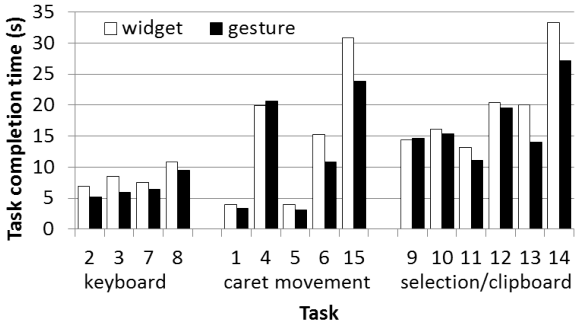


Figure 8. The mean task completion time for the medium font size grouped according to dominant interaction.

two words making the keyboard use highly dominant. These and task 4 with the medium font size were the only instances of widget superiority in the whole experiment.

Caret Movement Dominated Tasks

We classified tasks 1, 4, 5, 6, and 15 as caret movement dominated. Caret placement should be affected by the font size. With smaller font it becomes more difficult to hit the correct position. The gestural technique did result in shorter mean task completion times especially with the small font size.

Selection and Clipboard Use Dominated Tasks

Tasks 9-14 involved copy-paste actions which required a segment of the text to be selected, copied, and pasted (possibly multiple times). The tasks were fairly complicated with an average duration from 13 to 45 seconds. It was not uncommon for the participants to make a mistake that required them to redo actions. A particular difficulty involving tasks 10, 12, 13, and 14 was making selections at the beginning of the row. The beginning of the row coincided with the edge of the display making it difficult to place the caret especially with the widget-based technique. These tasks resulted in some of the largest differences to the advantage of the gestural technique.

Duration of Operations

Above we have given results on event counts and overall task duration. In this section we report the duration of the

operations within the tasks. We consider four basic categories: caret positioning, selection, clipboard use, and keypress. Note that some of these results correspond to sequences of user actions (e.g. retrying caret positioning until it succeeds) whereas others are single atomic actions (clipboard command, single keypress). This classification is partly due to the structure of our log files and partly because we think it reveals useful information about the interaction techniques.

Operations were timed starting from the end of the system response to the previous event and ending at the last input event included in the timed operation. The duration of our operations includes both the time needed to mentally prepare the operation and the time needed for the execution of the interactions. The durations were firstly averaged per participant and then over all participants.

The results are shown in Figure 9. Figure 9(a) shows the average time needed to perform a caret positioning sequence. The font size seems to have an effect on the duration of this operation. It can be performed faster in the gesture condition than in the widget condition. However, the difference is remarkable only with the small font.

Figure 9(b) shows the average time needed to perform a text selection sequence. It appears that operation can be performed faster with the gestural technique. The time is reduced by about 50%. In this case, the font size does not seem to have a dramatic effect on the duration of the operation.

Figure 9(c) and Figure 9(d) show the average time to perform a single clipboard operation (copy, cut or paste) and a single keypress. The clipboard operation is faster with the widget-based technique. As we noticed during the experiment, the users spent some time to figure out which gesture to perform, because they had some difficulty in memorizing the clipboard gestures (especially the Cut). A keypress seems to be slightly faster in the gestural technique. This shows that the use of the keyboard to perform the gestures does not hinder the normal use of the keyboard to enter text. The slight advantage in keypress performance achieved with the gestural technique can be because the caret position after the caret adjustment preceding the keypress was more predictable. Another partial explanation may be that the finger travel to the key is shorter when the keypress is preceded by a gesture on the keyboard rather than a widget operation on the text box farther away.

DISCUSSION

In our experiment the gesture set was seen by the participants all the time. In real editing situations such help would not be visible. A help function could be included in the user interface, but consulting it would be slow. A separate experiment on the learnability and discoverability of the gestures must be conducted before we know how easy they are to learn. We would expect the one-finger caret movement commands to be the easiest to memorize followed by the two-finger text selection commands. In the experiment we observed some initial difficulty in remembering the gestures for the clipboard commands.

Based on our results it seems that in the majority of editing tasks the gestural technique outperformed the widget-based technique. However, before concluding on the issue we must take into consideration the frequency of different types of editing tasks. The frequency of tasks in our experiment was probably not completely realistic. Clipboard control was probably overrepresented. We do not have data on the frequency of editing commands used in real-world editing tasks. However, we would expect one-character and one word corrections to dominate. While clipboard may be used, for example to copy an email address or an URL from the web, tasks such as task 14 that require repeated use of clipboard are probably rare. In real-world editing tasks the main advantage of the gestural technique would most likely be the easier caret placement. As mentioned earlier, some text entry systems already implement this.

The fact that the caret movement commands are probably the most useful ones is a good result concerning the learnability of the gestures. These gestures are probably the easiest to memorize in our set as they have an obvious spatial mapping between the gesture and the resulting movement. These gestures are also familiar to many users from cursor control using a touchpad.

Even if the performance advantage that can be gained with gestural editing is limited, it may be worth implementing because there is no downside. It is compatible with widget-based editing. Those users that do not wish to use one of the techniques can just ignore it. Also partial use, such as only doing caret movement with gestures is possible and potentially beneficial.

CONCLUSIONS AND FURTHER WORK

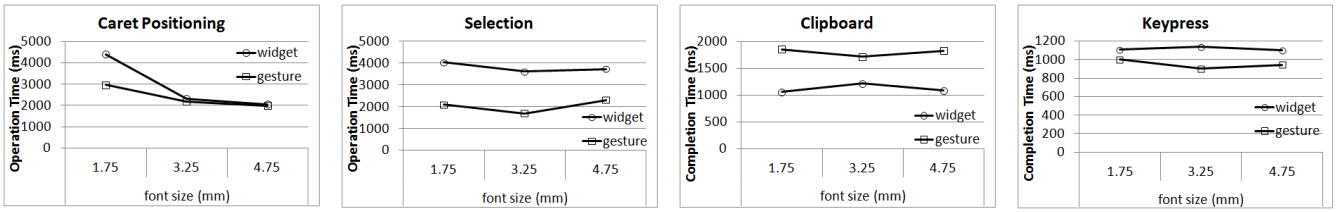
We have presented an implementation of a gestural editing technique that works on top of a soft keyboard on touchscreen devices. The technique can co-exist with the widget-based technique that is currently dominant in such devices. It cannot co-exist with other gestural techniques such as Shapewriter or Swype that already occupy the gesturing space on the soft keyboard.

In performance comparison to the default editing technique on Samsung's implementation of Android 2.3 platform the gestural technique was better. User feedback was also very positive. We can recommend implementing such techniques on all touchscreen devices where the gesturing space on top of the keyboard is free. If the gesturing space is not free, further study is required to find out which is the wisest way to use it.

Further work is also needed to optimize the gesture set for learnability and efficiency. Our gesture set was generated iteratively using trial and error. This process may have ignored design alternatives that could produce the same or better results. This design work is worth doing well because it would be desirable to have a standard gesture set for editing so that the user does not need to learn a different gesture set for each device.

ACKNOWLEDGMENTS

We thank numerous colleagues for useful discussions on mobile editing. Special thanks go to Gianluigi Avella for help in software development and Nathan Godard for help, among other things, in video editing.



(a) (b) (c) (d)

Figure 9. Average times needed to perform an operation with the two techniques.

REFERENCES

1. Almaksour, A., Anquetil, E., Quiniou, S., and Cheriet, M. Personalizable pen-based interface using lifelong learning. In *Proc. of ICFHR'10* (2010), 188 –193.
2. Appert, C., and Zhai, S. Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proc. of CHI'09*, ACM (2009), 2289–2298.
3. Bangor, A., Kortum, P. T., and Miller, J. T. An Empirical Evaluation of the System Usability Scale. *IJHCI 24*, 6 (2008), 574–594.
4. Borenstein, N. S. The evaluation of text editors: a critical review of the roberts and morgan methodology based on new experiments. In *Proc. of CHI'85*, ACM (1985), 99–105.
5. Bragdon, A., Nelson, E., Li, Y., and Hinckley, K. Experimental analysis of touch-screen gesture designs in mobile environments. In *Proc. of CHI '11*, ACM (2011), 403–412.
6. Brooke, J. Sus: A quick and dirty usability scale. In *Usability evaluation in industry*, P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, Eds. Taylor and Francis, London, 1996.
7. Cockburn, A., Ahlström, D., and Gutwin, C. Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse. *Int. J. Hum.-Comput. Stud. 70*, 3 (Mar. 2012), 218–233.
8. Costagliola, G., Fuccella, V., and Capua, M. D. Interpreting gestures for text entry on touch screen devices. In *Proceedings of The 16th International Conference on Distributed Multimedia Systems* (2010), 315–320.
9. Costagliola, G., Fuccella, V., and Di Capua, M. Text entry with keyscretch. In *Proceedings of the 16th international conference on Intelligent User Interfaces*, ACM (2011), 277–286.
10. Embley, D. W., and Nagy, G. Can we expect to improve text editing performance? In *Proc. of CHI '82*, ACM (1982), 152–156.
11. Gould, J. D., and Alfaro, L. Revising documents with text editors, handwriting recognition, and speech recognition systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society 26*, 4 (Aug. 1984), 431–441.
12. Gould, J. D., Lewis, C., and Barnes, V. Cursor movement during text editing. *ACM Trans. Inf. Syst. 3*, 1 (Jan. 1985), 22–34.
13. Isokoski, P. Performance of menu-augmented soft keyboards. In *Proc. of CHI '04*, ACM (2004), 423–430.
14. Kristensson, P.-O., and Zhai, S. Shark2: a large vocabulary shorthand writing system for pen-based computers. In *Proc. of UIST '04*, ACM (2004), 43–52.
15. Kristensson, P. O., and Zhai, S. Command strokes with and without preview: using pen gestures on keyboard for command selection. In *Proc. of CHI '07*, ACM (2007), 1137–1146.
16. Mankoff, J., and Abowd, G. D. Cirrin: a word-level unistroke keyboard for pen input. In *Proc. of the ACM UIST 98*, ACM (1998), 213 – 214.
17. Pedersen, E. R., McCall, K., Moran, T. P., and Halasz, F. G. Tivoli: an electronic whiteboard for informal workgroup meetings. In *Proc. of CHI '93*, ACM (1993), 391–398.
18. Roberts, T. L. *Evaluation of computer text editors*. PhD thesis, Stanford, CA, USA, 1980. AAI8011699.
19. Roberts, T. L., and Moran, T. P. Evaluation of text editors. In *Proc. of CHI '82*, ACM (1982), 136–141.
20. Roberts, T. L., and Moran, T. P. The evaluation of text editors: methodology and empirical results. *Commun. ACM 26*, 4 (Apr. 1983), 265–283.
21. Rubine, D. Specifying gestures by example. In *Proc. of SIGGRAPH '91*, ACM (1991), 329–337.
22. Tesler, L. A personal history of modeless text editing and cut/copy-paste. *Interactions 19*, 4 (Aug. 2012), 70–75.
23. Westerman, W., Lamiraux, H., and Dreisbach, M. Swipe gestures for touch screen keyboards. US Patent 20080316183, 2008.
24. Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. of UIST '07*, ACM (2007), 159–168.
25. Wolf, C. G., and Morrel-Samuels, P. The use of hand-drawn gestures for text editing. *International Journal of Man-Machine Studies 27*, 1 (1987), 91 – 102.