A text selection technique using word snapping

Motoki Miura\textsuperscript{a,}\textsuperscript{*}, Kenji Saisho\textsuperscript{b}

\textsuperscript{a}Faculty of Basic Sciences, Kyushu Institute of Technology, Japan
\textsuperscript{b}Department of Integrated System and Engineering, School of Engineering, Kyushu Institute of Technology, Japan

Abstract

Conventional copy-and-paste technique for touch screen devices utilizes region handles to specify text snippet. The region handles appear so as to select the initially tapped word, and the user controls the region handles. Most of the text-selection task is performed at the boundary of words, however, the minimum movement unit of the region handle is still a character. We propose a context-sensitive text-selection method for the tablet OSs. For the initial consideration, we investigated a word-snapping method that meant a word as a minimum movement unit. From our experiment, we confirmed that the word-snapping method can significantly reduce the text-selection time if the target text consists of one or two words, and no line breaks exist.

© 2014 The Authors. Published by Elsevier B.V.

Keywords: Multi-touch screen, text-selection, copy-and-paste

1. Introduction

Since the diffusion of smart phones and tablets, we often read and edit text with these touch screen devices for email, blog, SNS and so on. Most of the touch screen device provides virtual keyboard for inputting text. However, due to the lack of the physical tactile feedback, the inputting text with the virtual keyboard afford insufficient usability rather than physical keyboard. In order to reduce the amount of typing and burdens of inputting, we usually make full use of copy-and-paste technique especially for making quotation and reusing text during the editing.

Conventional copy-and-paste technique for touch screen devices utilizes region handles to specify text snippet. For example, Apple iOS and Google Android OS provide region handles when the user taps and holds him/her finger on a word (see Fig. 1). In both tablet OSs, the initial selected region is decided where the place of the tap-and-hold. If the place is on a “word,” the initial selection becomes the word. After the initial selection, region handles appear on the screen. The user can move the region handles for further selection. After the moving, the user press a “copy” or a “cut” button to keep the region in the clipboard.

\textsuperscript{*} Corresponding author. Tel.: +81-93-884-3429 ; fax: +81-93-884-3429.
\textit{E-mail address:} miuramo@mns.kyutech.ac.jp
In iOS, the user can precisely select the region by seeing the magnifying glass. The magnifying glass is an effective solution of fat-finger problem\(^1\). However, in the both tablet OSs, the minimum unit of the region handle movement is still a “character” while the further selection task. The “character-based selection” may decrease the efficiency of the text selection because it requires precise and careful control of the handles by fingers. If the tablet OSs provide a different minimum unit of the text selection by considering the context of the text, the usability of the text selection task can be improved.

We propose a context-sensitive text-selection method for the tablet OSs. In this paper, we mainly focus on a “word” as a context of the text. The “word” is a fundamental unit of a sentence, and it can be acceptable in various cases and situations of the text-selection task.

![Fig. 1. Region handles for text selection (left: Google AndroidOS, right: Apple iOS).](image)

![Fig. 2. Proposed method.](image)

2. Context-sensitive text-selection and its application to Tablet OSs

In this section, we describe the concept of the context-sensitive text-selection, and word snapping method as an instance.
2.1. Context-sensitive text-selection

The concept of the context-sensitive cut-copy-paste technique was presented by Wallace et al. in terms of programming\textsuperscript{2}. In their research, the editor recognizes the context of the source code, and enables the programmers to select a possible source code block (region) by simple repetitive click operations. Their research aimed to reduce the burden as well as the errors caused by the cut-copy-paste operations on the source code. Kerr and Stuerzlinger enhanced the approach to automatically fix the error caused by the difference of context when pasted\textsuperscript{3}.

The direction of enhancing cut-copy-paste edition by considering context is similar to our approach. We will apply the direction to a multi-touch interface.

2.2. Word snapping

As we described above, the usability of the text selection task can be improved if the tablet OSs consider context of the target text. We propose a word snapping method for text selection on the tablet OSs. The word snapping method changes the minimum unit of text selection as “a word” rather than “a character” while moving the region handles (see Fig. 2). Since most of the text selection is performed by the meaningful text like words or sentences, the proposed method reduces an irrelevant selection of text for cut-copy operations. The proposed method can also relief the burden of precise region handle controls.

In this paper, we investigate basic characteristics of the proposed word snapping method on several texts written in natural languages.

3. Related works

Baby-face problems\textsuperscript{4} and fat-finger problems\textsuperscript{1} had been recognized for designing handheld interfaces including PDA and cellular phones. For making pointing operations accurate, several approaches have been investigated\textsuperscript{5,6}. However, for text selections on tablet-OSs, not so many researches were performed yet.

Fuccella et al.\textsuperscript{7} proposed a gestural text editing technique for touch-screen devices. The (multi-touch) gestures drawn on the soft keyboard area are interpreted as commands for moving the caret and text selections. The gestures also control the clipboard. Since the input area was different, the proposed technique can coexist with conventional widget-based input methods. We consider that these gestures affect as similar to the alternative short-cut keys (ex. Ctrl+C, Ctrl+V) which are omitted on the smaller screen keyboards.

Scheibel et al.\textsuperscript{8} presented a virtual stick controller for precise caret positioning tasks. The virtual stick that emulates the function of joystick on a touch screen was implemented, and the performance was revealed. By comparing the finger tapping, the method had advantages when the movement distance was shorter, and the font size was smaller.
Cockburn et al.\textsuperscript{9} analyzed the fundamental performance of three input devices (the mouse, the stylus and the finger) across three different types of target acquisition activity (tapping, dragging, and radial dragging) on touch-based interactions. They revealed that the finger was fastest for tapping activities, but slowest for dragging. They also investigated that the errors in tapping activities was worst for the finger. We think that these results support that our proposed word-snapping method is appropriate for text-selection tasks because it will accept inaccurate and lower positioning resolutions by the fingers.

4. Experiment

We have conducted an experiment to verify the improvement of the usability of the text-selection interface according to the word-snapping technique. To evaluate the usability, we compared our method with a conventional text selection method. This chapter describes the procedure and results of the experiment and discussion.

4.1. Procedure

We asked 10 participants (all belonging to a student) to read documents on a 7-inch tablet (Nexus 7) in two different methods. Nine participants had been experienced with the (multi-)touch operations like tap, pinch, and swipe on a tablet. The participants utilize tablets or smartphones in their daily life. For text-selection and cut-and-paste functions on the tablet/smart-phon device, five participants utilized daily, and other four participants utilized once a week.

In order to conduct the experiment, we have developed a prototype application specialized for the experiment with Processing for AndroidOS environment. During the experiment, the participant were required to select a text snippet specified by blue-colored text with white-colored background (see Fig. 3). Hereafter, we call the text snippet as “target text.” When the participant tapped on the document, a character (or a word) at the point was selected, and two selection handles were appeared at the beginning and ending of the selected region. The range of the initial selected region was determined by the “snapping mode.” The selected region was represented as a red-colored text with yellow-colored background (see Fig. 4). After the initial tapping, the participant dragged the handles so as to select all target text specified. If the “word snapping mode” was selected, both handles were snapped at the word boundary. When the participant precisely select the target text, he/she taps on the “Answer” (or “Start”) button shown below the touch screen to proceed to the next task. If the selection was wrong, the overshooted region(s) were highlighted by pink-colored background (see Fig. 4). If the participant pressed the “Answer” button with overshooted, the system notices the error by playing sound.
We collected (1) time to complete a task, (2) touch count, and (3) overshoot time for each method. Since the purpose of this application is to estimate the fundamental properties of the proposed method, the all texts used in the experiment were pre-determined, and word separations of Japanese text were inserted by manual. In future, these separated text data can be generated by a morphological analyzer such as KAKASI\(^1\) and MeCab\(^2\).

We also conducted a questionnaire survey to the participants. The following items were asked in 5 Likert scale.

1. Do you think the character mode was easy to select texts? (1-5)
2. Do you think the word-snapping mode was easy to select texts? (1-5)
3. How much you want to use the word-snapping mode for future similar tasks? (1-5)

Fig. 5 shows the scene of the experiment. The participant sat on the chair, held the tablet by the non-dominant hand, and put the hand on the table. The participant used his/her dominant hand to tap-and-drag operations.

Fig. 6. Average task complete time. ** denotes 1% significance, and * denotes 5% significance.

4.2. Results and discussion

Fig. 6 shows average complete time of the tasks by 10 participants. The “task type” represents the number of lines and words of the target text. For example, “L2w7” means 7 words were included, and layed out in 2 lines. Appendix shows all screenshot of the tasks. The error bar denotes standard errors. Asterisk mark after the task number denotes significance level from conducting pairwise t-tests.

Regarding the average complete times, the word-snapping was significantly faster than the character-based in the three tasks (No. 2, No. 3, and No. 10). But in the two tasks (No. 5 and No. 13), the word-snapping was significantly slower than the character-based. We consider that the phenomena were caused by the following reasons. Firstly, the word-snapping method worked well if the target text is a word. Because the participants could select the region by one-tapping near the word. Also the word-snapping method was suitable for single line target. Because the participants could finish the task by moving handles horizontally. However, in the case of multiple lines target text, the participants should move the handle vertically. By our system design, the word snapping point was defined at the lower-middle of the word. When a word consist more characters, the gap between the word snapping point and the end point of the

\(^1\) http://kakasi.namazu.org/
\(^2\) http://mecab.sourceforge.net/
word was increased. Since we found that the gap distance influenced the performance of the word-snapping, we will fix the issue for future experiment. Secondly, the word-snapping method decreased the level of finger-tip feedback while dragging. This was caused by the reduction of the number of the possible snapping points for text-selection. In our implementation, the region handles were always stuck at one of the snapping points, and no extra carets/cursors were displayed. We can relief the issue by feedback of finger-tip position as well as the possible snapping points determined by the word separations.

Fig. 7 and Fig. 8 show average touch count and overshoot count during experiments, respectively. Regarding the touch count, no significant differences were observed except the task No. 13. The significance also caused by extra touches during moving the region end handle. Regarding the overshoot count, a few words tasks were advantageous for word-snapping method. The task No. 13 (2 lines, 3 words) was the most unsuitable task for word-snapping method.

Table 1 shows the result of the questionnaire survey. We performed a Mann-Whitney test for Q1 and Q2 answers under 5% significance levels. Consequently, were significantly differed (U = 8, p < .01). The result implies that the word-snapping method has a potential to relieve the burden of text-selection tasks if the current issues are solved.
5. Conclusion

In this paper, we introduced a word-snapping method, which considers text context for relieving burdens of text-selection on the touch-sensitive screens with tablet OSs. The word-snapping method provides the snapping of the region handles at the boundary of word. The method can relieve the precise control of the region handles.

We conducted an experiment to compare the word-snapping method with a conventional character-based method. The experiment revealed the word-snapping method can reduce the text-selection time if the target text consists of one or two words, and no line breaks exist. We could clarify the issues of the word-snapping method. If the issues are solved, the merit of the word-snapping method will be increased. In future work, we will also investigate the further extension by adding controlling handles by gestures.

References


Appendix A. Tasks

Figure A.9 – A.22 show the task screenshots of the experiment.
モトキ ミウラ と ケンジ サイトウ / プロデキシア コンピューターステーショング 35 (2014) 1644 – 1651

Fig. A.9. Task No.01

Fig. A.10. Task No.02

Fig. A.11. Task No.03

Fig. A.12. Task No.04

Fig. A.13. Task No.05

Fig. A.14. Task No.06

Fig. A.15. Task No.07

Fig. A.16. Task No.08

Fig. A.17. Task No.09

Fig. A.18. Task No.10

Fig. A.19. Task No.11

Fig. A.20. Task No.12

Fig. A.21. Task No.13

Fig. A.22. Task No.14