HAL
open science

Virtual Stick in Caret Positioning on Touch Screens

Jean-Baptiste Scheibel, Cyril Pierson, Benoît Martin, Nathan Godard, Vittorio Fuccella, Poika Isokoski

To cite this version:

Jean-Baptiste Scheibel, Cyril Pierson, Benoît Martin, Nathan Godard, Vittorio Fuccella, et al.. Virtual Stick in Caret Positioning on Touch Screens. 25ème conférence francophone sur l'Interaction Homme-Machine, IHM'13, AFIHM, Nov 2013, Bordeaux, France. 10.1145/2534903.2534918 . hal-00877314v2

HAL Id: hal-00877314

https://inria.hal.science/hal-00877314v2

Submitted on 6 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Virtual Stick in Caret Positioning on Touch Screens

Jean-Baptiste Scheibel[1], Cyril Pierson[1], Benoît Martin[1], Nathan Godard[1], Vittorio Fuccella[2], Poika Isokoski[3]

[1]UFR MIM / LCOMS
Université de Lorraine
Metz, France
benoit.martin@univ-lorraine.fr

[2]Dipartimento di Informatica
Università di Salerno
Fisciano (SA), Italy
vfuccella@unisa.it

[3]SIS / TAUCHI
University of Tampere,
Tampere, Finland
poika.isokoski@uta.fi

ABSTRACT
We present our design exploration in the area of virtual stick controllers and a preliminary evaluation in an editing task. Virtual stick controllers are one solution to the problem of precise pointing on touch screens. They operate by using an area of the touch screen as a rate control device that moves a pointer. We implemented and evaluated this technique in a text editing context where unaided precise placement of the caret is difficult. The results showed that with large fonts and long pointing distance positioning the caret with the virtual stick is significantly slower than with conventional finger touching. On the contrary, with small fonts and short pointing distance, we noted no difference.

Keywords
Gestures; Text editing; Caret movement; Android.

RÉSUMÉ
Nous présentons un travail exploratoire sur la définition d'un stick virtuel et une première évaluation pour définir des tâches pouvant en bénéficier. Un stick virtuel est une solution au manque de précision du pointage sur les écrans tactiles. Il utilise une zone interactive pour contrôler un pointeur en vitesse et en direction. Nous avons implémenté et évalué cette technique dans le cadre de l'édition de texte où le placement précis du curseur texte est difficile. Nos résultats montrent que lorsque la taille de la police est grande et que la distance de pointage est grande, le stick virtuel est plus lent que le pointage traditionnel. Au contraire, avec une petite taille de police et une distance de pointage courte, nous n'avons pas noté de différence.

Mots Clés
Gestes; Edition de texte; Placement du curseur texte; Android.

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

© ACM, 2013. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Actes de la 25ième conférence francophone sur l'Interaction Homme-Machine, 2013.
http://dx.doi.org/10.1145/2534903.2534918

INTRODUCTION
Mobile phones today are, as a matter of fact, powerful computers that can run many kinds of software traditionally associated with larger computers. The dominant user interface technology on these high-end phones is the touch screen. Some tasks are not well suited for touch screen use. One such task is text editing. People used to editing on desktop computers quickly become frustrated when trying to edit text using a touch screen.

On desktop computers, where indirect pointing with the mouse is used, there are many types of cursors. Two of them, the mouse pointer and the caret that shows the current insertion point in text are often visible simultaneously. On touch screens the mouse pointer does not exist as direct finger pointing is used. However, precise manipulation of the caret is still needed.

Caret positioning is one of the more frustrating parts of text editing on touch screens. Precise caret placement is difficult especially with smaller font sizes and especially near the edges of the display. Fundamentally, the difficulty stems from using the finger as the pointing device. Finger pointing is not as precise as mouse pointing [5] and using the finger brings with it the problem of occlusion.

An obvious workaround in the editing context for caret placement is to use virtual arrow keys on a soft keyboard. Some publicly available solutions are shown in Figure 1.

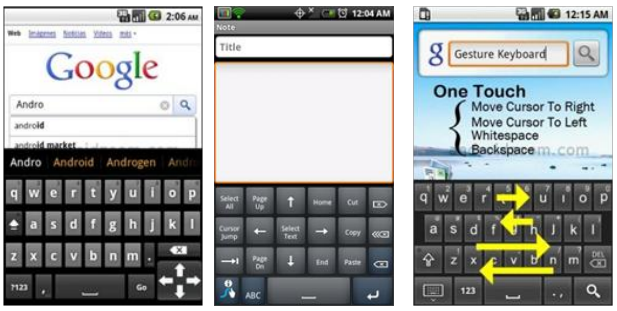

Figure 1. Solutions for caret movement: arrow keys (left - Arrows Keyboard [2], middle - Swype [17]) and gestures (right - Gesture Keyboard [9]).

The arrow keys can take the shape of a direction pad (D-PAD) or consist of separate keys for each direction. Further, they can be present all the time as in the Arrows

Keyboard [2] (Figure 1 – left), optional, but always present when turned on as in the Thumb Keyboard [18] or present when requested as with Perfect Keyboard [12]. In all cases, the arrow keys need a significant portion of the area of the soft keyboard. Often keyboard designers prefer not to show the arrow keys on the main layout. Instead they are available on an alternative layout as in Swype [17] (Figure 1 – middle). The Custom Keyboard [6] gives access to an onscreen touchpad instead of the arrows keys.

To avoid switching layouts, some keyboards use gestures done on top of the keyboard. On the Gesture Keyboard [9] swipes to different directions are mapped to the caret movements (Figure 1 – right). Gestures offer great possibilities that can be enhanced to text selection, clipboard access, etc. [8]. Unfortunately, there is competition for the gesturing space over soft keyboards. For example word gesture keyboards [19] can use the gesturing space for text entry. This makes it necessary to use a special starting point for command gestures to differentiate them from text entry gestures [10].

Bezel Swipe [13] helps in accomplishing the mode-switch from text entry to text selection, but does not improve the pointing accuracy for character level caret placement. An overview of the touch screen target acquisition techniques by Roudaut *et al*. [14] list a number of studies that have been successful in avoiding occlusion, improving precision, or both. The main approaches were zooming, various forms of object pointing or snapping, and the offset cursor. The most relevant prior work involves using the touch screen as a rate control device. We will discuss this work in the context of our work in the next section.

THE VIRTUAL STICK

In this section we describe our virtual stick implementation that we call the VS for brevity.

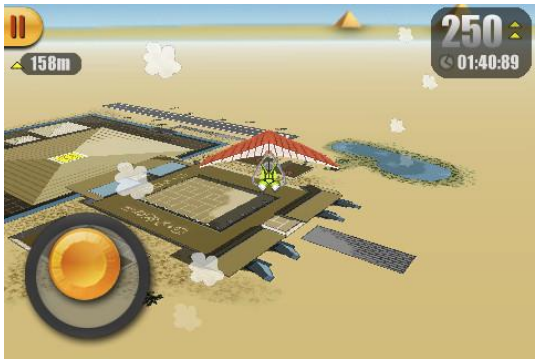

Figure 2. The virtual stick in "Flight School" an iOS game by Dough Scandrett [7].

Virtual sticks that emulate the function of joystick on a touch screen are used in the gaming context. Figure 2 shows an example at the bottom left corner of the display. In this paper we propose to use such a virtual stick to manipulate the caret in text editing and report on user preference and task efficiency. The virtual sticks we have seen in games inspired our work, but exact technical details on their implementation are usually not available. Thus our design was mostly guided by intuition and trial and error over many iterations.

In the context of a soft keyboard the virtual stick should consume as little space as possible and be usable without leaving the soft keyboard. We will first describe the general operation of the VS and then discuss the details of the design and implementation.

The zone of interaction

The VS uses a circular zone of interaction. It can be used with or without visual feedback. In our experiments we used it without visual feedback. The following figures are illustrations drawn for this paper they were not part of the user interface we constructed.

When the user touches the surface of the touch screen the VS is centered to the touch position: the finger is in the neutral zone (Figure 3 - left). The radius of the neutral zone is one of the parameters that must be decided in implementations. While the finger stays in the neutral zone, the caret does not move. Also, if the finger is lifted at this point, the key below the finger can be activated as usual. Thus, VS and the normal text entry function by tapping on the keys can coexist without a mode change.

When the user moves the finger out of the neutral zone onto the active zone (Figure 3 - left) the device vibrates to give tactile feedback. Moving the finger can be done by sliding it on the surface or by doing micro-rolls [15]. The relative efficiency and user comfort of micro-rolls and sliding is an interesting issue that for now remains unsolved. The size of the active zone is another parameter to be decided in implementations. Our decisions were based on trial and error. Determining the optimal size with a larger user population is another task for further work.

As long as the touch point P remains in the active zone, the caret moves with a speed that depends on the distance d to a direction that depends of the angle α shown in Figure 3 (right).

In text editing, horizontal and vertical caret movements are frequent. Therefore, a technique to facilitate them was found to be helpful in the informal evaluation phases of our iterative development of VS. Our facilitating technique uses "snapping". It can be related to prior work on pseudo-haptic that was first proposed by Lecuyer et al. [11] : pseudo-haptic systems are defined as "systems providing haptic information generated, augmented or modified, by the influence of another sensory modality".

In VS the user will have the feeling of a sticky axis. Non-orthogonal movements seem to require more "virtual effort" than they would without the snapping setup. The VS contains four snapping areas (Figure 4 - left). A touch point in such an area is projected circularly to the closest axis. The distance d remains the same

while the angle α is modified according to the axis. The sizes of the snapping areas are a third parameter to be decided for each implementation. It is noted as δ being the chosen half-angle of the snapping areas (Figure 4 - left).

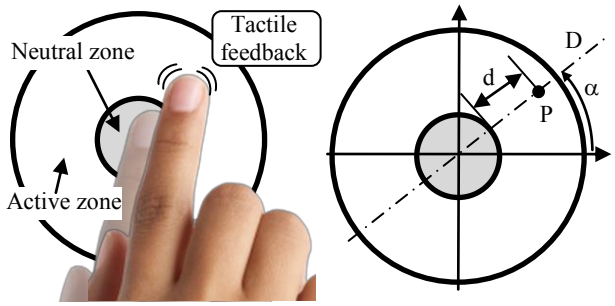

Figure 3. The VS input area (left) and its parameters (right).

To do non-orthogonal moves, the user has to move the touch position in the active zone outside the snapping areas. In this case, the interval of possible values is reduced. For example, in the north-east quarter (Figure 4 - right), the angle α only varies in the interval [δ;π/2-δ] with δ being the chosen angle of the snapping areas. To still make possible to move at any angle, we stretched the interval [δ;π/2-δ] to [0;π/2]. The distance d remains the same while the angle α is modified to α'. i.e. angular precision of non-orthogonal movements was traded for less precision required in orthogonal movements.

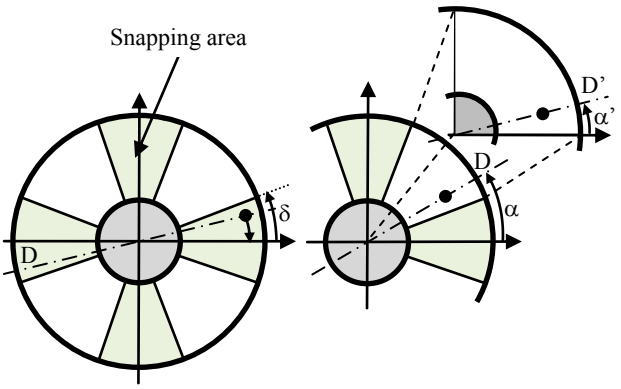

Figure 4. The snapping areas and stretching of the remaining arc.

Two behaviors were implemented to ease the use of the VS. The first one was intended to ease spatial positioning of the finger in the VS. When the finger is in the active zone, the user has a visual feedback with the movement of the caret. According to this movement, the user can partially evaluate where the finger is depending of the speed and the direction of the move. But when the finger is in the neutral zone, the pointer doesn't move. The user can be anywhere in the neutral zone. At the beginning of the interaction, when the user touches the surface, the VS is centered on the finger. We applied the same behavior when the finger enters in the neutral zone from the active zone (Figure 5 - left). The VS is centered on the first touch position in the neutral zone and the user feels a vibration as tactile feedback. He can continue his interaction as he was starting it.

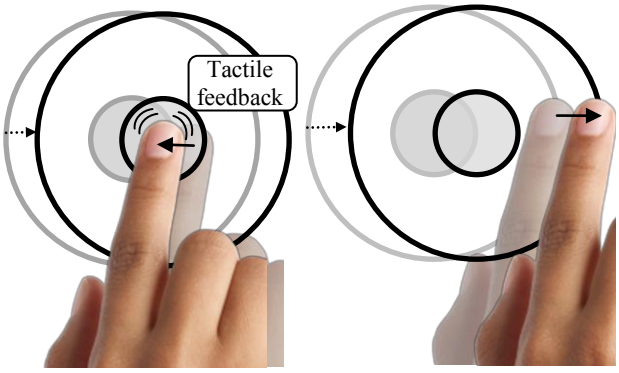

Figure 5. Behaviors of the Virtual Stick. Re-centering upon re-entry of the neutral zone (left) and VS follows the finger (right)

The second behavior guaranteed that the touch point always remained in the VS. When the touch point goes outside, the VS moves to follow it (Figure 5 - right).

The pointer

The VS is intended to move a pointer on the screen instead of moving directly the caret. This is done by interpreting the parameters of the VS, the distance d and the angle α. The angle indicates the direction of the move, no specific treatment is needed. The distance indicates the amplitude of the movement in the interval [0;1]. We can map this distance to speed in different ways. We decided to use mapping functions f:

$$s = \mathrm{f}\,(d)$$

A mapping function takes one parameter, the distance value in the interval [0;1]. It returns a normalized speed in the interval [0;1]. The value 0 corresponds to the neutral zone and the value 1 corresponds to the maximum speed the user can achieve. This result is multiplied by a maximum speed value to obtain the speed at which the pointer can move. This maximum can be adjusted by the user.

Choosing a mapping function is far from trivial. For example there has been a lot of work in investigating mapping functions for mice. For a recent example see the work by Casiez and Roussel [4]. Similarly work on mapping functions for isometric joysticks has been fruitful [16, 3]. However, we are unaware of similar work on virtual joysticks on touch screens. It is also not clear that the work on isometric joysticks is directly applicable when the device is not self-centering and does not give similar haptic feedback. Thus, we turned to the iterative trial-and-error process just like Dutledge and Selker in their work on isometric joysticks [16].

We tried several different original mapping functions. Figure 6 shows four of them. All the functions give the same maximum speed but the curve to obtain it is

different. The first one is a linear function that we chose for its simplicity:

$$f(d) = d \qquad (1)$$

The other three come from exponential function g:

$$g(d) = e^{((c*d-3)/1.4)} - e^{(-3/1.4)}$$

From this function g, we built the following mapping function to obtain a normalized speed in [0;1]:

$$f(d) = g(d) / g(1) \qquad (2)$$

The "constant" c of the function g can be adjusted to produce functions with different "sensitivity". The sensitivity is intended to be a user configured parameter. Figure 6 shows three examples of this function (2) with different sensitivity. The functions have been labeled "Exponential (c)" where c is this sensitivity. As shown in Figure 6, when the sensitivity is higher the control at low speed is better but at the end the speed will increase faster. With smaller values of c the function approaches the linear one (1).

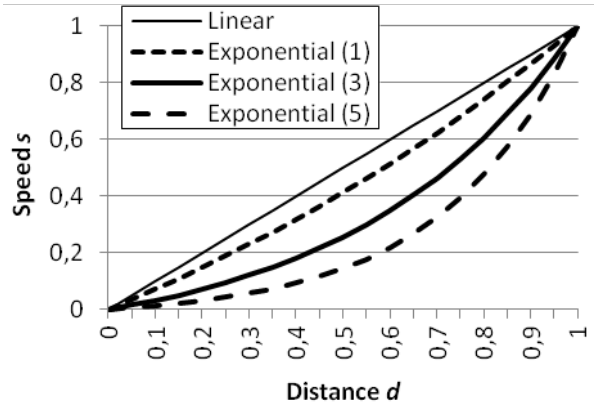

Figure 6. Four examples of mapping functions: one linear and three exponentials.

The appearance of the pointer is also important. It must be visible but it must not hide the caret or the underlying text. We decided to use a round green pointer that is displayed in the background of the text. The pointer also reflected the state of VS. When the finger was is in the neutral zone it was displayed as a circle (○) but when the user was in the active zone it was displayed as a disc (●).

The caret

When the user touched the surface, the pointer was centered on the caret. During the interaction with the VS, the caret moved by following the pointer. However, since the pointer moved continuously but the caret could only reside between characters, they do not always move in perfect synchrony. Instead, the caret is the projection of the pointer in the text. Also, the ends of the line of text are special. The pointer can proceed past the end, but the caret has no useful function there. In our implementation the caret remained at the end of the line when the pointer was beyond it.

EXPERIMENT

We know that the virtual sticks are used in games but data on their performance was not available. In our experiment we compared caret positioning with ordinary finger pointing to the VS.

Design

We had three interaction techniques for the same caret placement task. In the **Finger condition** the caret was placed by tapping with the finger, The **Virtual Linear (VL) technique** was VS with linear mapping (1) (function Linear in Figure 6). The **Virtual Exponential (VE) technique** was VS with the exponential mapping (2) (function Exponential(3) in Figure 6).

We used a between groups design to test the two mapping functions (VL and VE). The participants would probably have been unable to learn to use both at the same time. Both groups also performed the tasks in the Finger condition.

Both groups did the tasks twice, one time with a font size of 14sp and one time with a font size of 18sp. The unit sp means scale-independent pixels: one sp is one pixel on a 160 dpi screen if the user's global text scale is set to 100%. Both groups performed two sessions separated by at least 24 hours and less than 48 hours. Conditions were balanced among participants and sessions.

To record subjective impressions we used a questionnaire. The questionnaire was composed of 8 statements to which participants assigned a score indicating their strength of agreement in a 5-point scale. 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 touch screen.

Participants

We recruited a total of 16 participants (8 per group) from the university community in Metz, France. The mean age of the participants was 23.5 years (SD=1.4). One participant was left-handed. They were all users of smartphones with touch screen interfaces.

Apparatus

The software was run in the ACER Liquid Metal smartphone with Android (Gingerbread) 2.3.6. The device had a capacitive touch screen. The width of the display was 47 mm and the height was 77 mm. Its resolution is 480 x 800 pixels (259 dpi). With the user's text scale set to 100% the font size of 18sp was 2.8 mm and the font size of 14sp was 2.2 mm.

The setup for the VS was:

- radius of the neutral zone: 2mm;
- radius of the acive zone: 8mm;
- angle δ of the snapping areas : 20°;
- maximal speed of the pointer: 35mm/s;
- sensitivity for the exponential mapping: 3 (Exponential (3) in Figure 6).

When developing the software, we discovered differences between touch screens. For example some touch screens provided input data at one pixel intervals while some others provided input at lower resolution. Depending on the hardware, the feeling of the mapping in VS can be very different.

Figure 7 shows the exponential mapping (sensitivity of 3) both with a high accuracy and a low accuracy. With a high accuracy, the user can feel the variation of the speed as continuous. But with a lower accuracy, the variation is discrete and different according to the slope of the function. The low accuracy curve shows the actual curve on the device that was used in the experiment. It provided input resolution equal to 5 pixels.

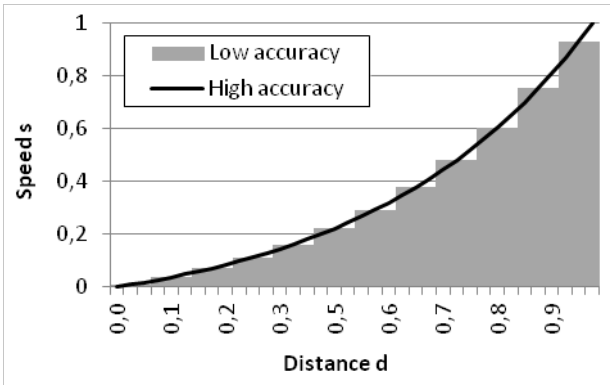

Figure 7. The exponential (3) mapping with different accuracies.

Procedure

Each session consisted of four blocks. Two blocks were performed with the finger technique and two blocks with the VS. For each technique one block was performed with the smaller font size (14sp) and the other with the larger font size (18sp). Each block contained 20 tasks of caret positioning. The next block started when all the tasks of the previous block had been finished.

The task was to move the caret from a start position to a target position. The start position of a task was the target position of the previous task. In a block, half of the tasks were pooled to the "near" category while the other half was pooled to the "far" category. The distance to move in a near task was less or equal to 4 characters to left or right and no more than 1 row up or down. In the far tasks the distances were longer (up to 25 characters horizontally and 10 rows vertically). The 20 tasks were identical in all blocks.

Figure 8 shows a screenshot of the display during a task. The text was always the same and displayed at the top of the display. The start position was indicated by the caret and the target was indicated by a vertical green line. To indicate the completion of the task, the participant pressed the button at the bottom of the display. If the caret was not at the target position the task remained active and the participant had to adjust the caret position until it was correctly placed.

With the two versions of VS, the participant interacted in the "VS Interaction Zone". In real world implementations this area would be the soft keyboard. To simplify the experiment we chose to display empty space instead of a soft keyboard. All other interactions in the "Text Zone" were disabled.

With the finger tapping technique, the participants interacted directly in the "Text zone". The caret was moved to the touched position. A draggable widget appeared below the caret and the participant could also drag the caret to another position as usual in touch-screen editing.

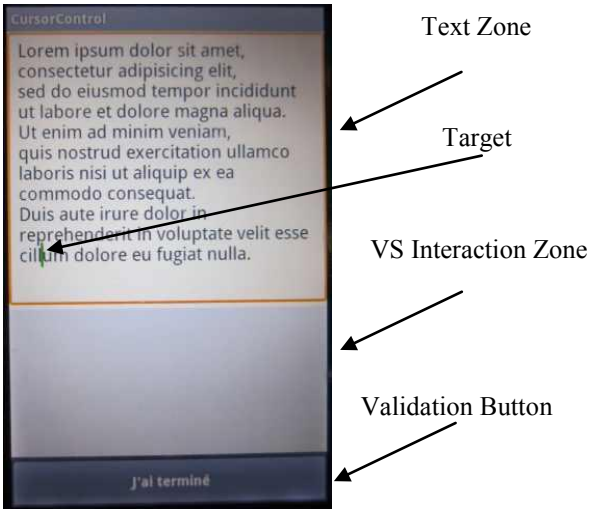

Figure 8. A screen shot of the display during a task.

No training was offered to the participants before the first session.

RESULTS

Our design was aimed at testing the linear and exponential mapping functions against each other and against the finger tapping. However, there were no statistically significant differences between the groups. Thus, we will report only the results comparing finger tapping and VS.

Performance results were compared using a repeated measures ANOVA separately for the two groups. The independent variables were moved distance (near, far), font size (14, 18) and the interaction technique (VS, Finger). The dependent variable was the task completion time. We report in detail only the data for the second session because in the first session the performance varied a lot as the task was new to the participants.

Performance of the VS

Figure 9 shows the mean task completion times for near and far targets with both font sizes for the VL group. The Finger technique was faster ($F_{1,7}$=7.6, p=0.028). The font size also had a main effect with larger font leading to

faster performance ($F_{1,7}$=9.1, p=0.02). Also, the target distance had a main effect ($F_{1,7}$=34.4, p=0.001) with longer distance leading to slower performance. Of the three possible interaction effects only the interaction of technique and distance was statistically significant $F_{1,7}$=32.3, p=0.001.

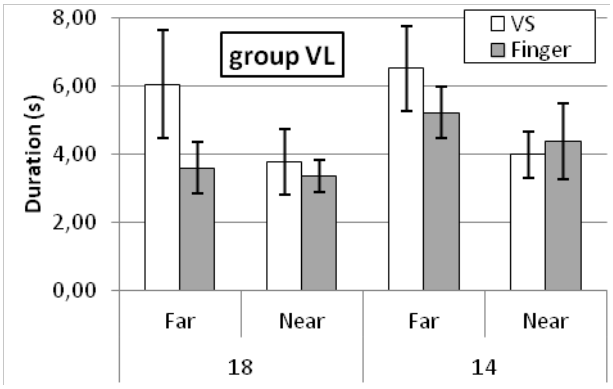

Figure 9. Group VL: performance in the second session.

Figure 9 helps in interpreting this interaction as larger effect of distance to task completion time with the VS. Bonferroni corrected pairwise t-tests showed that the difference between VS and Finger was statistically significant only in the far 18sp font condition. No other statistically significant effects were found.

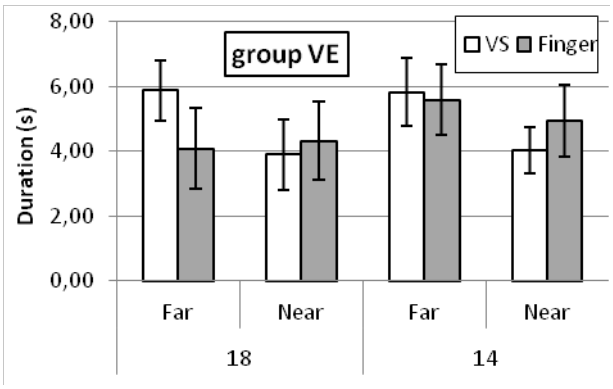

Figure 10. Group VE: performance in the second session.

For the group with exponential displacement-speed mapping (VE), figure 10 shows the result. The results were similar. Font size ($F_{1,7}$=6.6, p=0.037) and distance ($F_{1,7}$=59.2, p<0.001) had statistically significant main effects. The interaction of technique and distance was also statistically significant ($F_{1,7}$=25.8, p=0.001). The Bonferroni corrected pairwise t-tests also led to the same conclusion – only large font and long distance yielded a statistically significant difference.

The overall pattern of the results is that at long distances and large font sizes the finger technique was more efficient. With smaller font sizes and shorter distances the difference vanished.

Learning

Figure 11 shows the evolution of the performance between session 1 and session 2. As usual in experiments were participants are subjected to a new task they learn to perform the task more efficiently.

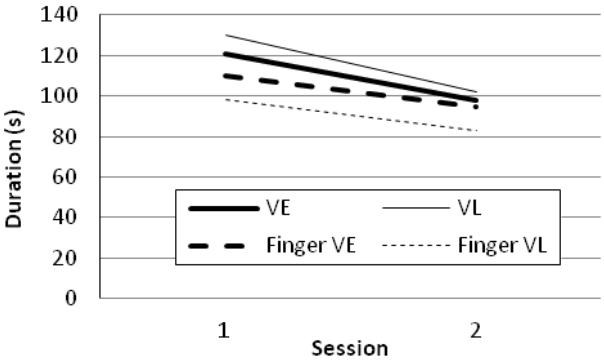

Figure 11. Performance improved from session 1 to session 2.

User Satisfaction and Free-form Feedback

At the end of experiment, the participants answered to the following statements presented to them in French:

1. I found easy to move the caret with the Virtual Stick
2. I have difficulties to stop the caret with the Virtual Stick
3. With the Virtual Stick, I know easily when the caret starts to move
4. The movements done on the Virtual Stick are not suitable to control the caret
5. Controlling the caret with the Virtual Stick is ergonomic
6. Controlling the caret with the Virtual Stick gave me fatigue and effort
7. I control easier the caret with the Virtual Stick than with my finger
8. I have had difficulties to move the caret with my finger.

Figure 12 shows the results (larger number indicates higher agreement). Only question 8 indicates a potentially meaningful difference.

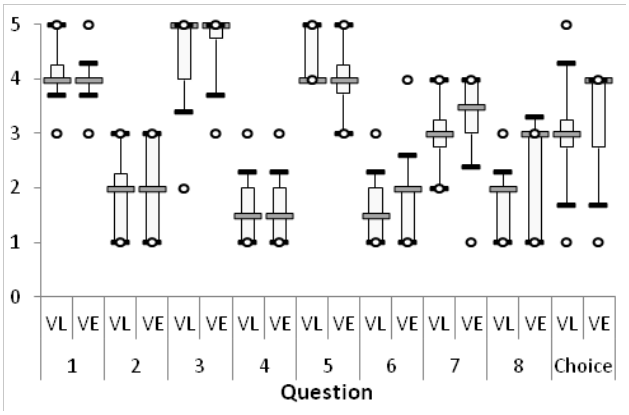

Figure 12. Subjective impressions.

We also asked the participants to choose between finger condition and VS condition (0 was the finger condition and 5 the VS condition). On Figure 12, the "Choice" bars

indicate the result. Most users did not have a clear preference. Instead they preferred to have the both techniques available and to be able to choose which to use according to the situation. From the participants' comments, we can note that the group VE found that the pointer was too fast and sometimes difficult to control. That can be due to the resolution issue mentioned in Figure 7. This effect appears more smoothly with the linear mapping. With a more accurate touch screen this problem may be smaller. In both groups, some participants mentioned a wish for tactile feedback for each movement of the caret or at the end points of a word. One participant mentioned that VS was preferable when the target was in between to 'l' letters.

DISCUSSION

Our results showed that tapping with the finger was generally the faster way to place the caret under the conditions that we tested. However, we also found that the relative efficiency of finger tapping and VS depended on the font size and distance to the target. VS was stronger with shorter distances and smaller fonts. We only tested two levels of distance and font size. Very short distances would probably show stronger benefit for VS against finger tapping. Font sizes below 14 would perhaps also make VS preferable.

To assess the value of VS in realistic editing tasks one needs to consider the frequency of long and short caret movement distances. Long distance movements do occur. For example, when locating the starting point of a text passage to copy the caret placement task corresponds to a long movement that is best accomplished with a direct tap or bezel swipe [13] or other such technique especially since whole words rather than individual characters are usually selected. However, short movements are frequent too. For example, when entering text into the computing device, one often makes typos that are noticed soon after they happen. In such cases only a short distance needs to be moved to correct the error. Also correcting some typos, such as transposition ("teh" instead of "the") requires one-character movement after the deletion and before the insertion.

We do not have statistics on the frequency of different caret movement distances in real world editing. However, in general VS can be recommended only if short distance caret movements are clearly dominant.

Further work is needed in selecting the best parameters for VS. We experimented with linear and exponential displacement-velocity mappings, but the results were inconclusive. It appears that the shape of the curve did not have a major effect under the conditions tested. Further work is also needed in measuring the beneficial area of VS use. The range of our "near" and "far" tasks was rather wide. Finer granularity especially in the "near" category is recommended in follow-up studies.

CONCLUSIONS

Based on our results it seems that the VS or another implementation of a Virtual Stick can compete with direct pointing in text editing tasks where short caret movements are often needed. While longer movements are usually faster completed by tapping with the finger, the final placement of the caret can be difficult without the VS or some other precise caret movement technique.

REFERENCES

1. Apple patent on http://www.touchuserinterface.com/2008/12/apple-patent-application-suggest-new.html
2. Arrows Keyboard on http://play.google.com/
3. R. C. Barrett, E. J. Selker, J. D. Rutledge, and R. S. Olyha. 1995. Negative inertia: a dynamic pointing function. *Conference Companion on Human Factors in Computing Systems CHI 1995*. ACM, 316-317.
4. Géry Casiez and Nicolas Roussel. 2011. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. *In Proceedings of UIST 2011*. ACM, 603-614.
5. Andy Cockburn, David Ahlström, and Carl Gutwin. 2012. Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse. International Journal of Human-Computer Studies. 70, 3, 218–233.
6. Custom Keyboard on http://play.google.com/
7. Flight School on https://itunes.apple.com/us/app/flightschool-lite/id351057268
8. Vittorio Fuccella, Poika Isokoski, and Benoît Martin. 2013. Gestures and widgets: performance in text editing on multi-touch capable mobile devices. *In Proceedings of CHI 2013*. ACM, 2785-2794.
9. Gesture Keyboard on http://play.google.com/
10. Per Ola Kristensson and Shumin Zhai. 2007. Command strokes with and without preview: using pen gestures on keyboard for command selection. *In Proc.of CHI 2007*. ACM, 1137-1146.
11. A. Lecuyer, S. Coquillart, A. Kheddar, P. Richard, and P. Coiffet. 2000. Pseudo-haptic feedback: Can isometric input devices simulate force feedback? *In Proceedings of IEEE VR'2000*, 83-90.
12. Perfect Keyboard on http://play.google.com/
13. Volker Roth and Thea Turner. 2009. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. *In Proceedings of CHI 2009. ACM*,1523-1526
14. Anne Roudaut, Stéphane Huot, and Eric Lecolinet. 2008. TapTap and MagStick: improving one-handed target acquisition on small touch-screens. In *Proceedings of AVI 2008*. ACM, 146-153.
15. Anne. Roudaut, Eric Lecolinet, and Yves Guiard. 2009. MicroRolls: expanding touch-screen input

vocabulary by distinguishing rolls vs. slides of the thumb. *In Proceedings of CHI 2009*. ACM, 927-936.

16. Joseph D. Rutledge and Ted Selker, 1990, Force-to-Motion Functions for Pointing, Proceedings of Interact 1990, North Holland, 701-706.
17. Swype on http://www.swype.com/
18. Thumb Keyboard on http://play.google.com/
19. Shumin Zhai and Per Ola Kristensson. 2012. The word-gesture keyboard: reimagining keyboard interaction. *Communications of the ACM*, 55, 9, 91-101.