ForceEdge: Controlling Autoscroll on Both Desktop and Mobile Computers Using the Force

Axel Antoine[1,2], **Sylvain Malacria**[1] **and Géry Casiez**[2]
[1]Inria, France [2]Université de Lille, France
axel.antoine@etudiant.univ-lille1.fr, sylvain.malacria@inria.fr, gery.casiez@univ-lille1.fr

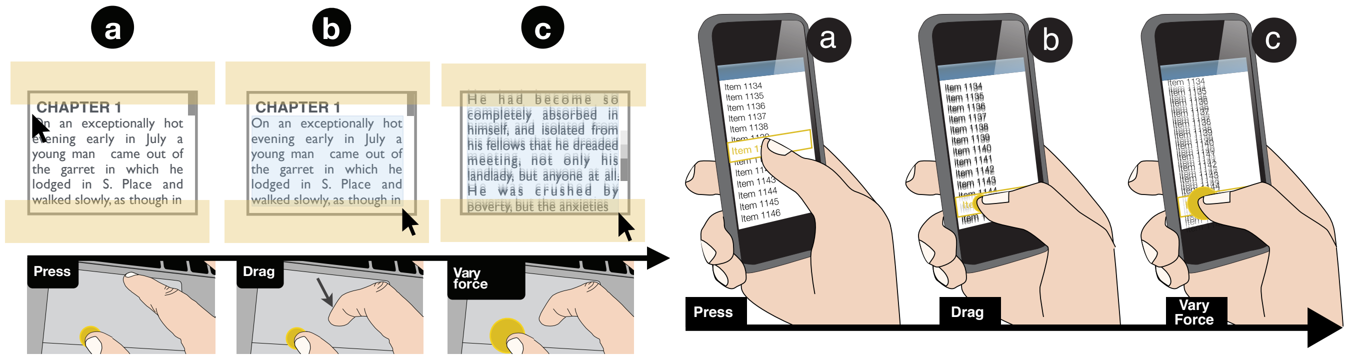


Figure 1: A user interacting with ForceEdge on a laptop computer (left) and on a smartphone (right). Left: She wants to select a large portion of text. In that purpose, (a) she presses the physical button of the trackpad and then (b) moves her finger on the trackpad to move the pointer in the control area. (c) She controls scrolling rate by varying the force applied to the trackpad. Right: she wants to move an object. (a) she starts moving the object, then (b) moves her finger in the control area and (c) controls scrolling rate by varying the force applied to the touchscreen.

ABSTRACT

Operating systems support autoscroll to allow users to scroll a view while in dragging mode: the user moves the pointer near the window's edge to trigger an "automatic" scrolling whose rate is typically proportional to the distance between the pointer and the window's edge. This approach suffers from several problems, especially when the window is maximized, resulting in a very limited space around it. Another problem is that for some operations, such as object drag-and-drop, the source and destination might be located in different windows, making it complicated for the computer to understand user's intention. In this paper, we present ForceEdge, a novel autoscroll technique relying on touch surfaces with force-sensing capabilities to alleviate the problems related to autoscroll. We report on the results of three controlled experiments showing that it improves over macOS and iOS systems baselines for top-to-bottom select and move tasks.

Author Keywords
Autoscroll; trackpad; touchscreen; scrolling; force control.

ACM Classification Keywords
H.5.2 Information interfaces (e.g. HCI): User interfaces

INTRODUCTION

Autoscroll, also known as edge-scrolling, is an interaction technique that allows users to scroll a viewport while in dragging mode: the user drags the pointer in a dedicated control area near the viewport's edge to trigger an "automatic" scrolling whose rate is usually proportional to the distance between the pointer and the viewport's edge. Autoscroll typically allows to scroll while *selecting* objects (e.g. a long portion of text) or while *moving* them (e.g. for drag-and-drop operations), both on desktop and mobile computers. And while alternative methods exist, previous work suggests that autoscroll remains used for selecting long portions of texts [1].

In spite of its wide availability, autoscroll still suffers from several limitations [1]. First, most autoscroll methods over-rely on the size of the control area, that is, the larger it is, the faster scrolling rate can be. Therefore, the level of control depends on the available distance between the viewport and the edge of the display, which can be limited. This is for example the case with small displays or when the view is maximized. This is also the case on touch-based devices such as smartphones that combine both limited screen real estate and maximized views. Second, depending on the task, the users' intention can be ambiguous. For *select* tasks (*e.g.* selecting a portion of text), there is no ambiguity about the viewport that contains the user's intended target: it has to reside within the viewport that contained the starting point (*e.g.* position where the user started selecting text). However, *move* tasks (*e.g.* dragging and dropping a file) are ambiguous as the user's target may be located within the initial viewport or in a different one on the same display (*e.g.* during drag-and-drop between windows). To reduce this ambiguity, the size of the control area is drastically smaller for move than for select tasks [1], which consequently 1) also affects scrolling rate control as the user has a limited input area to control the scrolling speed and 2) create inconsistency for similar operations operated in different contexts as the transfer functions to control the scrolling rate is different from one task to the other.

Several state-of-the-art input devices now offer *normal force-sensing* capabilities (typically, Force Touch trackpads on Apple Macbook [2], the Apple Magic Trackpad 2 [4], or the touchscreens of most recent iPhones [3]). In this paper, we explore how these force-sensing capabilities can be used to overcome the above-mentioned limitations of autoscroll, on both conventional desktop and mobile computers. Indeed, force-sensing is an interesting candidate for overcoming autoscroll limitations, especially when scrolling rate increases with the force applied to the input device, because: 1) users are usually already applying a (relatively soft) force on the input device when using autoscroll and 2) varying force on the input device does not require to move the pointer, thus making it possible to offer control to the user while using a small and consistent control area regardless of the task and the device.

In the following sections, after reviewing the use of force in Human-Computer Interaction, we present the design and evaluation of ForceEdge, an autoscroll technique that exploits force sensing capabilities for autoscrolling on both desktop and mobile computers. We first describe the theoretical foundations and the design of ForceEdge, as well as the transfer function used to convert the force applied to the input device in scrolling displacements. We then report on the results of two controlled experiments that compared ForceEdge to the macOS system baselines on a desktop computer, including for select tasks when the viewport edge is close from the border of the display. Finally, we report the results of an experiment that compared ForceEdge to the autoscroll techniques included on iOS for both *move* and *select* tasks.

THE DESIGN AND LIMITATIONS OF AUTOSCROLL

Various autoscroll techniques exist in commercial systems and applications. Aceituno et al. [1] recently reverse-engineered and compared 19 desktop autoscroll techniques from various operating systems and software applications in both select and move tasks. Their study confirmed that the control area usually expands to the edge of the display for select tasks (all reverse-engineered techniques), but is much more constrained for move tasks (all reverse-engineered techniques but one). It also confirmed that the vast majority of autoscroll techniques rely on rate-control based on the distance between the pointer and the viewport edge. This approach suffers from several problems. First, the available space around the viewport might be limited in some configurations, for instance with maximized windows, making it harder to control scrolling rate. Second, previous research suggest that rate-control is not adapted to isotonic devices such as trackpads and mouse controllers [36, 12]. Third, for *move* operations, the system cannot anticipate whether or not the target is located in the same viewport as the source. As a result, the system does not know which view should be scrolled, requiring alternative methods to raise this ambiguity, typically a smaller control area combined with a delay, which as a side-effect tends to limit user control. Finally, the system poorly suggests that autoscroll is available and what the size and location of the control areas are [1].

Novel autoscroll techniques have been proposed in the literature [25] or as inventions [7, 23]. Bardon et al. [7] proposed a relative position-based technique that scrolls the view of a given magnitude for every pointer movement outside and away from the viewport. This method remains restricted by the screen real estate as the pointer cannot endlessly move away from the viewport. Malacria et al. [25] proposed PushEdge and SlideEdge, two autoscroll techniques also using position-based control, but that *block* the pointer as it reaches the edge of the viewport and transform subsequent pointing device movements into scrolling displacements. These methods showed promising results for select tasks, but their application to move tasks remain unclear. Moreover, these methods could not be applied to touch-based computers as it is impossible to *block* the finger at the display edge. Finally, Kwatinetz et al. [23] proposed in a patent to map pointer acceleration to scrolling rate during move tasks, but no theoretical nor empirical evidence demonstrated the benefits of this approach.

Other works investigated specifically (and exclusively) the disambiguation problem during move tasks [8, 9, 24]. Indeed, the most common approach in current systems is to constrain the size of the control area, and sometimes to activate autoscroll after a short delay [1]. Berry et al. proposed as an alternative to use a dedicated physical button on the pointing device to specify which view the user wants to autoscroll [9]. Li et al. proposed a similar approach relying on a specific key of the keyboard [24]. Finally, Belfiore et al. proposed to trigger autoscroll only if the velocity of the mouse pointer is above a certain threshold when it enters the control area [8].

NORMAL FORCE FOR AUTOSCROLL

The psychophysics of normal force through fingers

The psychophysics of applying force (and by extension, pressure) through fingers have been extensively studied [16, 18, 19, 21, 27, 31], with Wilson's PhD thesis [34] providing an extensive review.

Overall, applying force on a surface through fingers relies on muscles located predominantly outside the hand, the sensory receptors of these muscles providing an instant feedback to the user on how much force is applied [22]. The maximum force that one can perform with a finger is around 17 N for a woman, and 45 N for a man, with a resolution of 0.3 N, regardless of the applied force [33]. It is usually accepted that the precision with which one can apply force when only inherent haptic feedback is available is relative to the magnitude of the pressure compared to one Maximum Voluntary Contraction (MVC), which corresponds to the maximum force one can apply. The relationship is approximately U-shaped, as precision decreases when applying low and high levels of pressure (relative to MVC) [19]. Also, additional visual feedback can significantly influence the precision of applied force with the index finger [20].

Detecting variation in applied force in touch-based HCI

From a hardware perspective, different methods can be used to estimate the normal force applied to a touch-sensitive surface. The most direct method is to use *force-sensing resistors* (FSR) [35], sensors whose resistance change when force is applied and that can be used to directly estimate a normal force in Newtons (N). It is possible to position FSRs between a smartphone and its case to estimate the normal and tangential force applied on its touchscreen [17, 26]. Strain gauges are also commonly used to measure force, typically by the Apple Magic Trackpad 2 [2] that can estimate a force up to 10 N.

Another method is to combine capacitive sensing with accelerometers values to estimate force when it bends a touch-surface, which is how the most recent Apple iPhone smartphones proceed [5]. From an input perspective, increasing the force applied to the touchscreen indeed increases the force estimated by the system. However, the actual value measured is not a force in Newtons. As a result, the returned value for a given force (in N) can vary depending on the user, the impedance of the finger, etc.

Touch-sensitive surfaces also provide a *pseudo-pressure* value that depends on the size of the contact area. While the size of the contact area generally increases with force [30], pseudo-pressure cannot be used to accurately measure normal force since different contacts with same area size can be performed with different forces, especially depending on the finger.

Finally, Goel et al. [13] proposed to combine orientation, contact area size and vibrations of a smartphone to infer the amount of pressure applied on a touchscreen, but managed to discriminate between three discrete levels of force only.

Use of force for autoscrolling

To the best of our knowledge normal force sensing has never been used or studied as an input canal for improving autoscroll techniques. Commercial products currently use force sensing for a limited number of very specific interaction scenarios, few of them in relation with autoscroll. The closest usage to autoscroll in commercial products is probably the use of force in Apple Quicktime where varying the pressure on fast-forward and rewind buttons will accelerate the rate at which a video file is browsed.

However, several interaction techniques using the force for improving document navigation have been proposed in the literature. Ramos et al. proposed to control the scroller of a scrollbar using a force-sensitive stylus, by mapping scrolling accuracy on the applied force [28]. Miyaki et al. used force for designing GraspZoom, an interaction technique combining force and tiny circular gestures for scrolling and zooming on a smartphone [26]. Heo and Lee [17] used tangential forces applied on a touchscreen as a method for quickly scrolling to the top or bottom of a document. In this case, the tangential force is used as a shortcut mechanism. Spelmezan et al. positioned additional FSRs on the outer edge of a smartphone to support rate-based scrolling [32]. Finally, pseudo-pressure was also used, either to decrease inertial scrolling speed [6] or changing the zoom level of a document [10].

Conceptually speaking, the most related work is probably Push-Push [14, 15] that proposes to use a threshold on force in order to discriminate when a touchscreen is *touched* or *pressed* (strongly touched). The user can perform *press* actions, for instance to define the start- and end- points of a text selection, and *touch* actions (as well as hovering over the touchscreen) to navigate in the document. While Push-Push can be used to perform similar tasks as autoscroll, it is a different interaction mechanism (yet compatible) that can be seen as using shift+click actions on a desktop computer. Moreover, Aceituno et al. [1] showed that autoscroll remains used, even when alternative methods are available, which motivates to improve autoscroll methods rather than designing novel interaction techniques that might ultimately remain unused.

FORCEEDGE

Concept

ForceEdge (illustrated Figure 1) has been designed to overcome the limitations of current autoscroll methods, by not relying on the distance between the pointer and the window edge. Instead, autoscroll velocity increases with the force applied to the input device.

Three conditions are required to trigger ForceEdge. First, the user has to enter and maintain *dragging* mode (figure 1-a), typically for text selection or a drag-and-drop operation. Second, the user has to move the pointer in a dedicated control area located in the vicinity of the viewport's edge. Third, the force applied on the pointing device has to be above a certain threshold (figure 1-b). Once ForceEdge is activated, the user controls scrolling velocity by applying more or less force on the input device (figure 1-c). ForceEdge scrolls as long as the pointer remains in the control area and the force remains above another threshold.

ForceEdge has several theoretical advantages over existing autoscroll techniques.

First, because ForceEdge does not rely on the distance between the pointer and the window edge, the size of the control area does not have to be as large as with existing techniques for text selection (that is, expanding from the viewport to the edge of the display). As a result, the control area can be the exact same size for selection and drag-and-drop operations, making interacting with ForceEdge consistent regardless of the task.

Then, ForceEdge does not require significant hand movement compared to current text selection or drag-and-drop operations. Since these operations imply that the button of the input device (*e.g.* trackpad) is already pressed, users can control scrolling velocity by adjusting the force applied to the device.

Finally, ForceEdge can be easily adapted to both desktop and touch-based interfaces, as long as the input device is force sensitive, which is the case for some commercially available trackpads and touchscreens [2, 3, 4, 5].

ForceEdge transfer function

The transfer function defines the mapping between the force applied on the touch surface in Newtons and the scrolling speed in millimeters per second. Using physical units allows to design a transfer function that is replicable as it is independent of the device input and output resolutions and frequencies [11]. We first determined the relationship between the raw force input from an Apple Magic Trackpad 2 and the force applied in Newtons using brasses of known weights and we simulated a touch by wiring the hand to the weight. We obtained the following linear relationship between the raw input measures raw_{input} and the force F in Newtons: $F = 9.8\,10^{-3} \times raw_{input} + 1.3\,10^{-3}; (R^2 = 0.99)$.

We first experimented a simple linear function but it did not yield a good control of the scrolling rate. Inspired by the transfer function designed for the TrackPoint [29], we used a sigmoid parabolic shape function (Figure 2). First the function requires a dead band near zero force to avoid accidental activations. In addition, as the user is already applying some force on the surface during dragging, we defined an hysteresis where the user has to apply more than 3.4 N to activate ForceEdge (trigger curve on Figure 2) and then switch to the control curve above 3.8 N. The beginning of the control curve is designed to get an accurate control at low scrolling rate without introducing too much strain in fine motor control. For long distances reaching, high speed is required. We defined a plateau at 800 mm/s, reached at 8 N (which is less than half the the max force females can apply [33]) to get a speed high enough and keep the content on screen perceivable, if not readable. The transition between low and high speeds follows a parabolic shape to keep best control at moderate speeds.

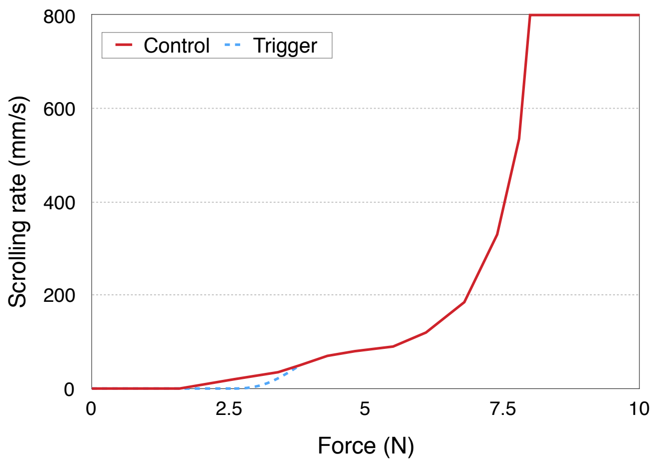


Figure 2: ForceEdge transfer function mapping the normal force applied on the surface in Newtons to the scrolling rate in millimetres per second. The function relies on an hysteresis to avoid accidental activations and keep a good control over the scrolling speed: the *trigger* curve is used upon invocation of autoscroll and switches to the *control* curve once the force applied reaches 3.8 N.

The activation threshold of 3.4 N was defined after running a pilot experiment to determine how much force users apply during dragging operations. We asked 12 participants to perform sequences of 45 *select* and *move* tasks using an Apple Magic Trackpad 2 connected to a 15" Macbook pro laptop. We implemented in Objective-C/Cocoa a dedicated software that displayed in blue various lines of text (10, 70, 210 lines) that the participants had to select or move using the default macOS autoscroll method, giving a total of 1080 trials. The average force applied to the trackpad was 2.8 N (σ =0.2 N). The average force applied was relatively similar for *select* (2.7 N) and *move* (3.0 N) operations. The average maximum force applied was 4 N (σ =0.2 N). For operations that did not require autoscroll, the average (2.6 N) and maximum (3.5 N) forces applied were slightly less than for the others (3.0 N and 4.2 N respectively). The activation threshold of 3.4 N was chosen as a value in-between the average force and maximum force, as a good trade-off to minimize accidental activations of ForceEdge while keeping a good range of force to control the scrolling speed.

To fine tune the transfer function, we designed an application to interactively manipulate control points and perform autoscroll tasks. The application also highlighted the current force being applied on the transfer function curve: the curve from zero force to current force was thicker. This information was then used to adjust the scrolling speed for a given force if necessary. Based on trial and error, the transfer function designers (the co-authors of this work) tuned the different parts of the curve until getting what was considered as a correct control of the scrolling rate for each part of the curve.

When several fingers are in contact with the trackpad, the total force is used to control ForceEdge. We also considered using only the force from the first or last finger in contact but informal tests revealed this mapping was more difficult to discover and did not appear to offer more control. For mobile devices, we decided to use the same transfer function by simply replacing the force applied in Newtons (value between 0 and 10 N for the trackpad) by the force estimated by the touchscreen (value between 0 and 6.67 for an iPhone6S, not in Newtons). As a result, the activation threshold on the smartphone is 2.3 units and the control curve is reached for 2.5 units.

STUDY 1: FORCEEDGE ON DESKTOP

We conducted a first experiment to compare the performance and perceived workload between *ForceEdge* and the macOS *System-Based* autoscroll methods in top-to-bottom *select* and *move* tasks on a desktop computer.

Method and apparatus

Our experimental method was strongly inspired by Malacria et al. [25] and Aceituno et al. [1] experimental procedures. The experiment was conducted on a 15" Apple Macbook Pro running macOS 10.11.5, with display resolution set to 1440×900 pixels, 110 ppi. Input was provided through an Apple Magic Trackpad 2 with force-sensing capabilities. The overall force value applied on the trackpad was used for ForceEdge. Experimental software was written in Objective-C with the Cocoa API. Contact point locations and force values were monitored using Apple's private multitouch API. The software implemented ForceEdge, and it used Apple NSTextViews' default rate-based autoscroll methods for the baseline conditions, whose behaviours are described in [1].

Procedure, task and design

Participants were instructed to perform a sequence of top-to-bottom autoscroll operations (*select* and *move*) as quickly and accurately as possible. The window was displayed at the centre of the screen, was 104 mm tall and contained 5496 lines of text typeset with *Helvetica Neue* 13px (figure 3). For each trial of the *select* condition, participants had to select a section of text framed and coloured blue – starting ten lines from the bottom of the viewport, and varying in size (in the following, we will refer to size as *distance*, since it is the distance the user had to scroll). For each trial of the *move* condition, participants had to drag a line highlighted in blue, located ten lines from the bottom of the viewport, and drop it on a target located at varying distance. Supposing that autoscroll is mainly target-directed, we overlaid a gradient on the scrollbar as a hint of the number of lines participants had to autoscroll. As the task was one-dimensional, selecting anywhere on a line selected the entire line. Each selection required concurrently dragging and scrolling using the requested autoscroll technique. Other scrolling methods were disabled. Once in dragging mode, the control area was emphasized as a semi-opaque blue area (see figure 3) regardless of the task or technique used. We decided to emphasize the control areas because the lack of visibility of the control areas has been highlighted as a potential issue for autoscroll techniques [1], especially for move tasks where it can result in accidental start and stop of the autoscroll, for both the ForceEdge and Baseline conditions.

To complete a trial, participants had to move the pointer to the top of the starting target (blue text), press the pointing device to start the selection (and enter in dragging mode), and autoscroll downwards until they reached the destination. Once the pointer was positioned over the target line, they could leave dragging mode to complete the operation by releasing the trackpad button. If the entire required text was not correctly selected or the line was not dropped at the right position, an error was recorded, and the trial was repeated until successfully completed. Participants were free to rest anytime between trials, as long as they were not operating the pointing device.

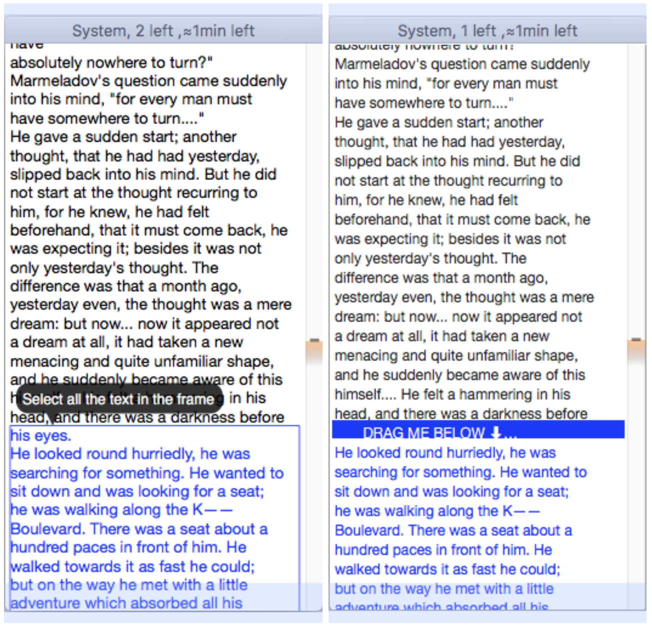
The experiment used a $2 \times 2 \times 3 \times 3$ within-subjects design for the factors: *task* (Select or Move), *technique* (ForceEdge or Baseline), *block* (1-3, with the first block serving as opportunity for learning the new method), and *distance* (short: 15; long: 135; longest: 250 lines). The order of *task* and *technique* was counterbalanced across participants. Distances were presented in ascending order, one for each *block*, with 5 consecutive repetitions for a given distance within a block – for a total of $2 \times 2 \times 3 \times 3 \times 5 = 180$ correct trials per participant. Participants completed NASA-TLX worksheets after each technique. The experiment lasted ~30 minutes.

Participants

Sixteen university staff and students (one female) participated in the experiment (mean age 25.7, SD=4.4). Ten used a mouse as main pointing device, but all were familiar with trackpads.

Hypothesis and dependant measures

The primary hypothesis were **H1:** selection time would be lower for ForceEdge than for system-based functions; and **H2:** ForceEdge would offer better control and result in less overshoots in long-selections. Therefore, the primary dependant measures were the total time to complete the trial and the overshoot distance.



System, 2 left ,≈1min left

absolutely nowhere to turn?"
Marmeladov's question came suddenly
into his mind, "for every man must
have somewhere to turn...."
He gave a sudden start; another
thought, that he had had yesterday,
slipped back into his mind. But he did
not start at the thought recurring to
him, for he knew, he had felt
beforehand, that it must come back, he
was expecting it; besides it was not
only yesterday's thought. The
difference was that a month ago,
yesterday even, the thought was a mere
dream: but now... now it appeared not
a dream at all, it had taken a new
menacing and quite unfamiliar shape,
and he suddenly became aware of this
h Select all the text in the frame in his
head, and there was a darkness before
his eyes.
He looked round hurriedly, he was
searching for something. He wanted to
sit down and was looking for a seat;
he was walking along the K——
Boulevard. There was a seat about a
hundred paces in front of him. He
walked towards it as fast he could;
but on the way he met with a little
adventure which absorbed all his

System, 1 left ,≈1min left

Marmeladov's question came suddenly
into his mind, "for every man must
have somewhere to turn...."
He gave a sudden start; another
thought, that he had had yesterday,
slipped back into his mind. But he did
not start at the thought recurring to
him, for he knew, he had felt
beforehand, that it must come back, he
was expecting it; besides it was not
only yesterday's thought. The
difference was that a month ago,
yesterday even, the thought was a mere
dream: but now... now it appeared not
a dream at all, it had taken a new
menacing and quite unfamiliar shape,
and he suddenly became aware of this
himself.... He felt a hammering in his
head, and there was a darkness before
DRAG ME BELOW ⬇...
He looked round hurriedly, he was
searching for something. He wanted to
sit down and was looking for a seat;
he was walking along the K——
Boulevard. There was a seat about a
hundred paces in front of him. He
walked towards it as fast he could;
but on the way he met with a little

Figure 3: The interface used for our experiments. For select tasks (left), participants had to select the entire text framed in blue. For move tasks (right), participants had to drag the top line and drop it on the last blue line. For both tasks, a gradient was overlaid on the scrollbar as a hint of the number of lines participants had to autoscroll. For each technique, the control area was highlighted in blue (here for ForceEdge)

Results

Trial time

Trial time is the main dependent measure and is defined as the total selection/move time, from the first pointer movement after the trackpad button was pressed, to the button release. Trials marked as errors were removed from the timing analysis.

Repeated-measures ANOVA[1] revealed significant effect of *block* on trial time ($F_{2,30} = 62.1$, $p < 0.0001$, $\eta_p^2 = 0.80$; block 1: 6.6s, 2: 5.5s, 3:5.2s). Pairwise comparisons showed a significant decrease ($p < 0.0001$) in the trial time between the first block and the two remaining, due to a familiarization with the experimental procedure. As we are concerned with user performance after familiarization, the first block was removed from subsequent analysis.

The overall trial time was 6.4s for *select* tasks, and 4.4s for *move* tasks. It was 4.3s with *ForceEdge* and 6.4s with *Baseline*. Repeated measures ANOVA showed a significant main effect of *task* ($F_{1,15} = 61.4$, $p < 0.0001$, $\eta_p^2 = 0.80$), *technique* ($F_{1,15} = 121.6$, $p < 0.0001$, $\eta_p^2 = 0.89$), *distance* ($F_{1.1,17.3} = 108.6$, $p < 0.0001$, $\eta_p^2 = 0.88$) and significant *task* × *technique* ($F_{1,15} = 42.7$, $p < 0.0001$, $\eta_p^2 = 0.74$), *task* × *distance* ($F_{2,30} = 34.2$, $p < 0.0001$, $\eta_p^2 = 0.69$), *technique* × *distance* ($F_{2,30} = 78.2$, $p < 0.0001$, $\eta_p^2 = 0.84$) and *task* × *technique* × *distance* ($F_{2,30} = 19.8$, $p < 0.0001$, $\eta_p^2 = 0.57$) interactions on trial time. For each *task*, post-hoc analysis showed significant differences ($p < 0.003$) between ForceEdge and Baseline. For *select* tasks, the average trial time was 4.1s with

[1]Greenhouse-Geisser corrections to the degrees of freedom were applied when sphericity was violated. Pairwise comparisons used Bonferroni correction.

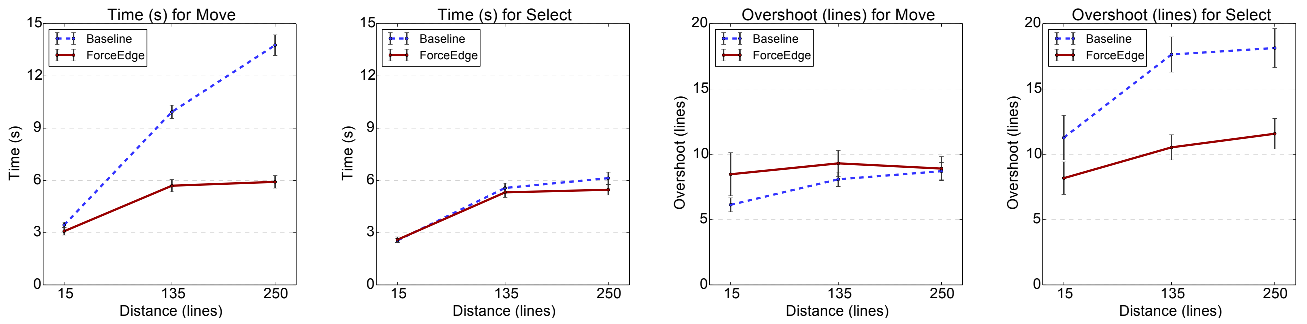


Figure 4: Results of Study 1, from left to right: trial time (s) for move, time (s) for select, overshoot (lines) for move, overshoot (lines) for select.

ForceEdge and 4.6s with Baseline. For *move* tasks, average trial time was 4.5s with ForceEdge and 8.2s with Baseline. As a result **H1** is confirmed. Pairwise comparisons showed significant differences ($p < 0.01$) between the two techniques for each distance of the two tasks, except for the shortest distance and the Select task where ForceEdge and Baseline showed similar performance (Figure 4).

Overshoot distance
Overshoot distance was measured as the maximum scroll distance beyond the target bounds that was reached during the trial. Trials marked as error were removed from the analysis.

Repeated-measures ANOVA revealed a significant main effect of *block* ($F_{2,30} = 4.2$, $p < 0.03$, $\eta_p^2 = 0.22$) and significant *block* × *distance* interaction ($F_{4,60} = 3.5$, $p < 0.02$, $\eta_p^2 = 0.19$) on overshoot distance. Pairwise comparisons showed a significant increase ($p < 0.03$) in the overshoot distance between the first block and the two other ones, only for the shortest distance (Block 1: 7.0 lines, 2: 9.5 lines, 3: 9.1 lines). There was also a significant ($p < 0.03$) increase in the overshooting distance between blocks 1 and 3 (Block 1: 9.8 lines, 2:11.0 lines, 3:10.8 lines). As these results do not clearly suggest a learning or fatigue effect, we kept all blocks for subsequent analysis.

Repeated-measures ANOVA revealed a significant main effect of *task* ($F_{1,15} = 44.7$, $p < 0.0001$, $\eta_p^2 = 0.75$), *technique* ($F_{1,15} = 15.3$, $p < 0.001$, $\eta_p^2 = 0.50$), *distance* ($F_{2,30} = 17.3$, $p < 0.0001$, $\eta_p^2 = 0.54$) and significant *task* × *technique* ($F_{1,15} = 54.0$, $p < 0.0001$, $\eta_p^2 = 0.78$) and *technique* × *distance* ($F_{2,30} = 5.9$, $p < 0.007$, $\eta_p^2 = 0.28$) interactions on overshoot distance. Post-hoc analysis reveal that for the Move task the overshooting distance is significantly ($p < 0.0001$) lower for ForceEdge (10.1 lines) compared to Baseline (15.7 lines) (Figure 4). As a result **H2** is partially validated: the better results obtained with ForceEdge for the Move task cannot be simply explained by lower overshooting distances, but because of a limited control of scrolling velocity. There is also a significant ($p < 0.002$) difference between the shortest distance (8.5 lines) and the two other ones (medium: 11.4 lines, long: 11.8 lines).

Error rate
Error rate is measured as the percentage of trials not successfully completed. Repeated-measures ANOVA revealed significant effect of *block* on error rate ($F_{2,30} = 11.8$, $p < 0.0001$, $\eta_p^2 = 0.44$; Block 1: 8.4%, 2: 4.6%, 3: 5.1%). Pairwise comparisons showed a significant decrease ($p < 0.037$) in the error rate between the first block and the two remaining. No other effect was found.

NASA-TLX Subjective preferences
A Friedman analysis on the NASA-TLX responses found significant effects for *performance* ($\chi^2(3) = 14.1, p < 0.005$) and *frustration* ($\chi^2(3) = 21.9, p < 0.0001$). Wilcoxon post-hoc analysis revealed participants found themselves significantly ($p < 0.002$) more successful using ForceEdge compared to Baseline, in both tasks. They also had higher frustration ($p < 0.007$) using Baseline compared to ForceEdge in both tasks. Finally, out of the 16 participants, 14 participants preferred ForceEdge over the baseline for move tasks, and 12 preferred it for select tasks.

STUDY 2: FORCEEDGE WHEN CLOSE-TO-EDGE

The results of the previous experiment highlighted the benefits of ForceEdge compared to the system baseline for both tasks, with a smaller effect size for select tasks. It is likely that the actual performance of the system baseline for this task is actually overrated as the experimental procedure used a relatively small window centred on screen, which is a 'best case' scenario for the baseline because of the large control area outside the window.

In this follow-up experiment, we compared the performance, amount of control and perceived workload between ForceEdge and the System-Based autoscroll for select tasks, with different amount of space between the window and the display edge. The main hypothesis was that selection time would be lower for ForceEdge than for the system baseline when the window is close from the display edge.

Method

Our experimental method was based on Study 1, with modifications described below.

Procedure, task and design

Participants were instructed to perform a sequence of top-to-bottom text selections using the given autoscroll as quickly and accurately as possible. We decided to investigate select tasks only as the system-based function for move tasks is theoretically not influenced by the distance between the window and display edge because the size of the control area is consistent regardless of the window configuration. The window was displayed with its bottom edge either 5.2 (*center*), 2.6

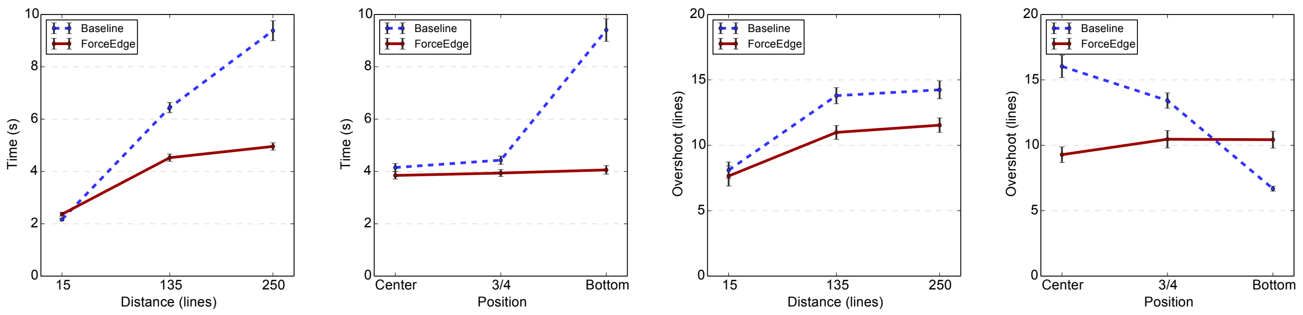


Figure 5: Results of study 2, from left to right: trial time (s) per distance, per position; overshoot (lines) per distance, per position

(3/4) or 0.4 (*bottom*) cm away from the bottom edge of the display. Center is the same condition as in study 1. Bottom corresponds approximately to the height of a status bar, which is usually the only space available for performing autoscroll when a window is maximized. All other characteristics of the experiment were the same as in Study 1.

The experiment used a $2 \times 3 \times 3 \times 3$ within-subjects design for the factors: *technique* (ForceEdge or Baseline), *position* (Center, 3/4 or Bottom), *block* (1-3), and *distance* (short: 15; long: 135; longest: 250 lines). The order of *technique* and *position* was counterbalanced across participants. *Distance* were still presented in ascending order, one for each *block*, with 5 consecutive repetitions for a given distance within a block – for a total of $2 \times 3 \times 3 \times 3 \times 5 = 270$ correct trials per participant. Primary dependent measures were the same as in Study 1. Participants completed NASA-TLX worksheets after each technique. The experiment lasted ~40 minutes.

Participants

The same sixteen university staff and students participated in the experiment right after Study 1.

Hypothesis and dependant measures

Primary hypothesis were **H1:** selection time with baseline would increase when the window is close from the display edge; and **H2:** Baseline would result in less overshoots when the window is close from the display edge because of a capped scrolling rate. Therefore, primary dependant measures were the total time to complete the trial and the overshoot distance.

Results

Trial time

Trials marked as errors were removed from the timing analysis. Repeated-measures ANOVA revealed significant effect of *block* on trial time ($F_{2,30} = 5.4$, $p < 0.01$, $\eta_p^2 = 0.27$; block 1: 5.0s, 2: 4.9s, 3: 4.9s). Pairwise comparisons showed a significant decrease ($p < 0.006$) in the trial time between the first and last blocks. Considering the mean absolute difference between the first and last block is very small, we kept all blocks for subsequent analysis. Repeated-measures ANOVA revealed a significant main effect of *technique* ($F_{1,15} = 356.0$, $p < 0.0001$, $\eta_p^2 = 0.96$), *position* ($F_{2,30} = 623.9$, $p < 0.0001$, $\eta_p^2 = 0.98$), *distance* ($F_{1.1,17.4} = 418.3$, $p < 0.0001$, $\eta_p^2 = 0.96$) and significant *technique* $\times$ *position* ($F_{1.3,19.1} = 384.7$, $p < 0.0001$, $\eta_p^2 = 0.97$), *technique* $\times$ *distance* ($F_{2,30} = 376.2$, $p < 0.0001$, $\eta_p^2 = 0.96$), *position* $\times$ *distance* ($F_{4,60} = 565.8$, $p < 0.0001$, $\eta_p^2 = 0.97$) and *technique* $\times$ *position* $\times$ *distance* ($F_{4,60} = 293.9$, $p < 0.0001$, $\eta_p^2 = 0.95$) interactions on trial time. Post-hoc analysis reveal that for Baseline the bottom position is significantly ($p < 0.0001$) slower compared to the two other positions (bottom: 9.4s, 3/4: 4.4s, center: 4.1s), when ForceEdge remains on average at 3.9s. No significant differences between *position* was found for ForceEdge. As a result **H1** is confirmed. Significant differences ($p < 0.03$) were found between ForceEdge and Baseline for each *position* $\times$ *distance*, except for the center position and short distance and also for the 3/4 position and the medium distance (Figure 5). ForceEdge is significantly faster than Baseline for all *position* $\times$ *distance*, except for the shortest distance ($p < 0.03$) where it is significantly slower (ForceEdge: 2.4s, Baseline: 2.2s).

Overshoot distance

Trials marked as errors were removed from the overshoot distance analysis. Repeated-measures ANOVA revealed no significant main effect of *block* thus all blocks were kept for the analysis. Significant main effects were found for *technique* ($F_{1,15} = 13.6$, $p < 0.002$, $\eta_p^2 = 0.50$), *position* ($F_{2,30} = 57.3$, $p < 0.0001$, $\eta_p^2 = 0.80$), *distance* ($F_{1.1,17.1} = 21.2$, $p < 0.0001$, $\eta_p^2 = 0.64$) and significant *technique* $\times$ *position* ($F_{2,30} = 86.7$, $p < 0.0001$, $\eta_p^2 = 0.84$), *technique* $\times$ *distance* ($F_{2,30} = 11.9$, $p < 0.0001$, $\eta_p^2 = 0.30$) and *technique* $\times$ *position* $\times$ *distance* ($F_{2.5,37.9} = 4.4$, $p < 0.02$, $\eta_p^2 = 0.26$) interactions on overshooting distance. Post-hoc analysis revealed significant ($p < 0.001$) differences between the two techniques for each *position*. ForceEdge shows significantly less overshooting than Baseline for the center (ForceEdge: 9.3 lines, Baseline: 16.0 lines) and 3/4 (ForceEdge: 10.4 lines, Baseline: 13.4 lines) positions while it is the opposite (ForceEdge: 10.4 lines, Baseline: 6.6 lines) for the bottom position (Figure 5). Thus **H2** is confirmed. For the bottom position, ForceEdge has significantly ($p < 0.0001$) higher overshooting distance compared to Baseline for all distances. For the center and 3/4 positions, it is the opposite ($p < 0.01$) for all distances, except for the short distance and the 3/4 positions where there is no significant difference.

Error rate

Repeated-measures ANOVA revealed a significant main effect of *technique* ($F_{1,15} = 7.9$, $p < 0.02$, $\eta_p^2 = 0.35$) on error rate. Pairwise comparison reveal that ForceEdge has a significantly higher ($p < 0.02$) error rate compared to Baseline (ForceEdge: 6.6%, Baseline 4.3%). No other effect or interaction was found.

NASA-TLX and subjective preferences
A Friedman analysis on the NASA-TLX responses found significant effects for *performance* ($\chi^2(1) = 6.2, p < 0.01$) and *frustration* ($\chi^2(1) = 5.4, p < 0.02$). Wilcoxon post-hoc analysis revealed participants found themselves significantly ($p < 0.003$) more successful using ForceEdge compared to Baseline. They also had higher frustration ($p < 0.001$) using Baseline compared to ForceEdge. In this condition, 14 participants out of 16 preferred using ForceEdge than the system baseline.

STUDY 3: FORCEEDGE ON SMARTPHONES

Autoscroll is an interaction technique available on various platforms, including touch-based computers such as tablets or smartphones. In the context of autoscroll, touch-based interaction has several significant differences with conventional desktop interactions, mostly because it does not follow the WIMP (Window Icon Menus Pointer) paradigm. First, a touch-contact is usually less accurate than a system pointer. Second, mobile operating systems tend to rely on fullscreen views rather than windows, thus removing available space between the window and display edge. Third, dragging mode is activated via various methods, for instance a time delay or by moving the finger from a specific location on screen. Finally, the finger is not constrained within the bounds of the display, which can result in leaving the dragging mode by moving the finger out of the tracking area of the display, even though it is still in contact with the surface. For these reasons, we decided to conduct a third experiment to compare the performance and perceived workload between *ForceEdge* and the *System-Based* autoscroll methods in top-to-bottom *select* and *move* tasks on a smartphone.

Method and apparatus

Our experimental method was based on Study 1 and conducted on an iPhone 6S running iOS 10 beta. Input was provided through the touch-screen with force-estimating capabilities of the smartphone. Experimental software was written in Objective-C using the CocoaTouch API.

Techniques
We decided to compare ForceEdge to the autoscroll methods used in Apple iOS for moving objects and selecting text.

ForceEdge. ForceEdge adopted the same behaviour as in the desktop experiment, therefore the same interaction technique was used regardless of the task (move or select). The only difference with the desktop was in the mapping between the force applied to the touchscreen and the scrolling rate, as detailed at the end of the *ForceEdge transfer function* section.

Baseline for move. The size of the control area for autoscrolling while moving an object on iOS starts from the edge of the display and expands off-screen. Conceptually, the control of the autoscrolling-rate remains distance-based, except that it depends on the proportion of the moved object that is intersecting the control area. Autoscrolling rate starts from 15 mm/s as soon as the moved object intersects the control area, and linearly increase up to 70 mm/s.

Baseline for select. The view autoscrolls when the finger is positioned in a 3 mm high control area, located near the edges of the display. Unlike conventional autoscrolls, the transfer is time-based rather than distance-based, that is scrolling rate starts at 4.6 mm/s and increases over time for eventually reaching 1560 mm/s after 3.5 seconds. As a result, autoscrolling rate can only increase and users must move their finger out of the control area (or lift it off the screen) in order to stop autoscrolling and reset scrolling rate.

Procedure, task and design

Participants were instructed to perform a sequence of top-to-bottom autoscroll operations (*select* and *move*) as quickly and accurately as possible. The view was displayed as a full screen view with the exact same physical properties as the view of study 1 and 2, that is 104mm tall, containing 5496 lines of text typeset with *Helvetica Neue*. To complete each trial, participants had to select a section of text or drag a line to a specific target, as in previous studies. A gradient was still overlaid on the right side of the view as a hint of the number of lines participants had to autoscroll. Every selection required to use the requested autoscroll technique. Other scrolling methods were disabled. Participants were free to hold and operate the smartphone with the finger/hand(s) they wanted.

To complete a trial, participants had to position their finger on the starting target, move their finger in the control area and autoscroll downwards until they reached the destination. Once the finger was positioned over the target line, they could lift their finger off to complete the operation. If the trial was not correct, an error was recorded, and the trial was repeated. Since the core of the experiment was to investigate performance with autoscroll techniques, and not mode switchers to perform select and move operations on touch-based devices the conventional touch scrolling techniques were disabled. As a result, any touch event selected/moved the object located underneath it. Moreover, in order to minimize errors because of the low accuracy of the finger, the software corrected touch events that were exactly one line above or below the starting target so the selection actually started from the target line. Participants were not informed of this correction mechanism.

The experiment used the exact same $2 \times 2 \times 3 \times 3$ within-subjects design than study 1 for the factors: *task* (Select or Move), *technique* (ForceEdge or Baseline), *block* (1-3, with the first block serving as opportunity for learning the new method), and *distance* (short: 15; long: 135; longest: 250 lines). Primary dependant measures were the same as in study 1 and 2. Participants completed NASA-TLX worksheets after each technique. The experiment lasted ~30 minutes.

Participants

Sixteen university staff and students (eight of which took part in the previous experiments) participated in the experiment (two female, mean age 27.5, SD=5.6). All owned a touch-based device such as a tablet or a smartphone.

Hypothesis and dependant measures

The primary hypothesis were **H1:** that selection time would be lower for ForceEdge than for system-based functions, and **H2:** that ForceEdge would result in less error; Therefore, the primary dependant measures were the total time to complete the trial and the overshoot distance and the error rate.

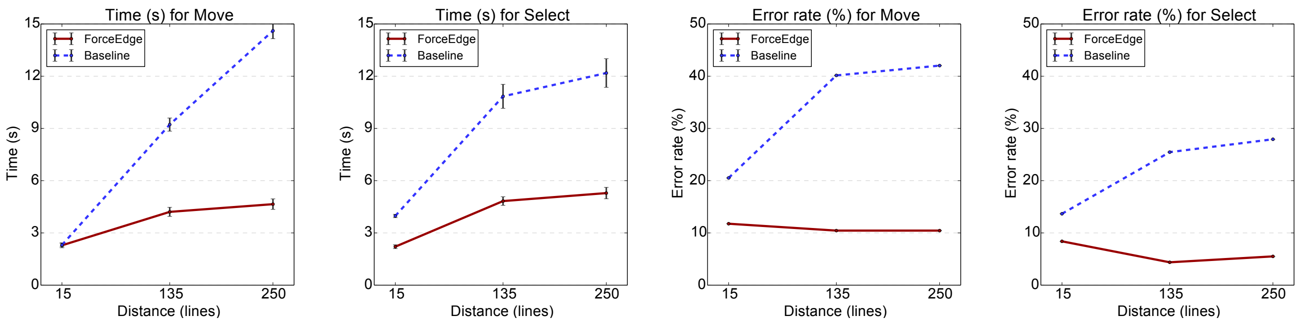


Figure 6: Results of Study 3, from left to right: trial time (s) for move, time (s) for select, error rate (%) for move, error rate (%) for select.

Results

Trial time

Trial time is the main dependent measure and is defined as the total selection/move time, from the first finger movement after the touchscreen was touched, to the finger release. Trials marked as errors were removed from the timing analysis. Repeated-measures ANOVA revealed significant effect of *block* on trial time ($F_{2,30} = 29.9$, $p < 0.0001$, $\eta_p^2 = 0.66$; block 1: 6.9s, 2: 6.2s, 3: 6.0s). Pairwise comparisons showed a significant decrease ($p < 0.0001$) between the first and the two remaining. The first block was thus removed from subsequent analysis. Repeated measures ANOVA showed a significant main effect of *technique* ($F_{1,15} = 453.9$, $p < 0.0001$, $\eta_p^2 = 0.97$), *distance* ($F_{1.2,18.3} = 175.3$, $p < 0.0001$, $\eta_p^2 = 0.92$) and significant *task* × *distance* ($F_{2,30} = 8.3$, $p < 0.001$, $\eta_p^2 = 0.36$), *technique* × *distance* ($F_{2,30} = 203.8$, $p < 0.0001$, $\eta_p^2 = 0.93$) and *task* × *technique* × *distance* ($F_{2,30} = 19.1$, $p < 0.0001$, $\eta_p^2 = 0.56$) interactions on trial time. For each *task* and *distance* post-hoc analysis showed significant lower trial times ($p < 0.0001$) for ForceEdge compared to Baseline, except for the Move task and the shortest distance where there was no significant difference (Figure 6, left). Overall the trial time was 3.6s using ForceEdge and 8.6s for baselines (Select: 3.9s ForceEdge, 8.7s baseline; Move: 3.4s ForceEdge, 8.5s baseline). As a result **H1** is confirmed.

Overshoot distance

No significant effect of *block* was found on overshooting distance. Repeated-measures ANOVA revealed a significant main effect of *task* ($F_{1,15} = 15.2$, $p < 0.001$, $\eta_p^2 = 0.50$), *distance* ($F_{2,30} = 7.4$, $p < 0.002$, $\eta_p^2 = 0.33$) and significant *task* × *technique* ($F_{1,15} = 11.0$, $p < 0.005$, $\eta_p^2 = 0.42$), *technique* × *distance* ($F_{2,30} = 7.9$, $p < 0.002$, $\eta_p^2 = 0.34$) and *task* × *technique* × *distance* ($F_{2,30} = 7.5$, $p < 0.002$, $\eta_p^2 = 0.33$) interactions on overshoot distance. Post-hoc analysis reveal that for the Move task and for each *distance* the overshooting distance is significantly ($p < 0.006$) larger for ForceEdge compared to Baseline. However for the Select task and the medium and long distances, it is significantly ($p < 0.04$) lower for ForceEdge compared to Baseline, except for the shortest distance where no significant difference was found. Overall overshooting distance was 16.9 lines for ForceEdge and 15.7 lines for Baselines.

Error rate

No significant effect of *block* was found on error rate. Repeated measures ANOVA showed a significant main effect of *task* ($F_{1,15} = 42.6$, $p < 0.0001$, $\eta_p^2 = 0.74$), *technique* ($F_{1,15} = 116.4$, $p < 0.0001$, $\eta_p^2 = 0.89$), *distance* ($F_{2,30} = 10.2$, $p < 0.0001$, $\eta_p^2 = 0.40$) and a significant *task* × *technique* ($F_{1,15} = 5.3$, $p < 0.035$, $\eta_p^2 = 0.26$) interaction on error rate. For both tasks pairwise comparisons showed significant ($p < 0.0001$) lower error rates for ForceEdge compared to Baseline (Select, ForceEdge: 4.4%, Baseline: 16.1%; Move, ForceEdge: 8.3%, Baseline: 28.5%) as illustrated figure 6 (right). In addition there is a significant ($p < 0.004$) difference between the error rate for the short distance (10.5%) and the two other ones (16.2%). As a result **H2** is validated.

NASA-TLX and subjective preferences

A Friedman analysis on the NASA-TLX responses found significant effects for *mental* ($\chi^2(3) = 24.3, p < 0.0001$), *physical* ($\chi^2(3) = 13.2, p < 0.004$), *temporal* ($\chi^2(3) = 32.0, p < 0.0001$), *performance* ($\chi^2(3) = 34.2, p < 0.0001$), *effort* ($\chi^2(3) = 31.3, p < 0.0001$) and *frustration* ($\chi^2(3) = 43.9, p < 0.0001$). Post-hoc analysis reveal significant negative answers for Baseline compared to ForceEdge in both tasks for all criteria, except for mental demand where there is no significant differences for the Select task. Finally, all participants preferred ForceEdge over the system baselines for both select and move tasks.

Performance compared to the desktop

To know if participants were more efficient using ForceEdge on desktop or mobile we kept ForceEdge data from experiments 1 and 3. Considering all conditions except the device were the same in the two experiments, the *device* was a between-subject factor while *task*, *block* and *distance* remained within-subject factors. The ANOVA showed a significant effect for the *device* × *task* interaction ($F_{1,30} = 8.1$, $p < 0.008$, $\eta_p^2 = 0.21$). Post-hoc comparisons showed a significant ($p < 0.02$) difference between mobile and desktop, only for the Move task (desktop: 4.5s, mobile: 3.4s).

DISCUSSION

Studies 1 and 2 compared ForceEdge to the the current macOS autoscroll baselines. Study 1 showed that ForceEdge improves over these baselines for top-to-bottom select and move tasks. While the benefits are significant for Move tasks, resulting in a trial time 45% shorter with ForceEdge, the benefits remain limited for *select* tasks (11% faster). One possible explanation is because the experiment was conducted with a window centred on screen, which corresponds to the best case scenario for the pointer-to-edge distance based baselines. Therefore, study 2 compared ForceEdge to the macOS baseline for select tasks with various configurations of windows. The results confirmed that unlike ForceEdge, the baseline is strongly influenced by the space available around the window, with trial time being as much as 141% longer with the baseline than with ForceEdge when the space around the window is limited. Finally, study 3 compared ForceEdge to the current iOS autoscroll baselines. The results showed that it improves on performance (58% faster) and resulted in less error (16% absolute difference), among others because users tend to move their finger out of the tracking zone of the touchscreen with the pointer-to-viewport distance based method. These studies confirmed the potential of ForceEdge when operated on a trackpad (desktop) or touchscreen (mobile), and that it is neither affected by the task or the location of the window.

Why did ForceEdge work for both tasks/platforms?

Our experiments showed that ForceEdge improved on macOS and iOS baselines for top-to-bottom autoscrolling operations.

This performance advantage can be explained through the fact that ForceEdge relies on the specific input canal of force-sensing, which is unlikely to be affected by the specific issues of autoscroll identified by Aceituno et al. [1]. Indeed, unlike conventional methods, varying force does not require pointer operations. Moreover, the user is most of the time already applying force to the input device when in need for autoscroll (because in dragging mode). In the end, interacting with ForceEdge required to 1) enter dragging mode, 2) position the pointer (or finger) in the control area and 3) control autoscrolling rate by varying force. These 3 steps are performed the exact same way for both tasks. This explains why the performance was similar with ForceEdge for both tasks, unlike with the system baselines.

ForceEdge outperformed the system baselines on the smartphone for similar reasons. The results are emphasized by the fact that the available space for controlling autoscroll on mobile devices is very limited because the views are almost systematically in full screen mode, which is probably the reason why iOS designers decided to implement a time-based method rather than a distance-based one. That being said, our experiment highlights the limitations of that approach as ForceEdge outperformed this time-based baseline by 58%. In addition, the comparison of the results of Study 1 and Study 3 suggests that ForceEdge is as efficient on desktop computers than on smartphones. The small difference in term of performance is likely to be the result of various factors (different participants, different force sensing technology) but the overall performance is very satisfying. This result suggests that ForceEdge could possibly be as efficient with other input devices, for instance a force-sensing mouse controller.

Limitations of our studies

Our experimental design had a limited number of baseline, distance and direction conditions in order to ensure that our experiments would be of bearable durations.

On the mobile platform, we used the default iOS behaviors as we are not aware of any better alternative that we could have used. On the desktop platform, we compared ForceEdge to the macOS system baselines in a concern of replicating the experiment from [25] and because for the distances we tested, it did not significantly differ from other rate-based methods in previous studies [1]. And while methods from the literature such as Slide-Edge [25] were shown promising for select operations on desktop, it is unclear how to apply them to move operations. That being said, comparing ForceEdge to other system-baselines remains interesting but is left as future work.

Finally, our experimental procedure focused on top-to-bottom autoscrolling operations because current GUIs still heavily rely on 1D vertical layouts (e.g, webpages, word processing, file browsers). However, motor performance can differ according to the direction and testing ForceEdge under these conditions is another interesting perspective for future work.

Compatibility with existing systems and techniques

ForceEdge relies on force-sensing input devices, that become more and more available on the market [4, 3]. This input canal however remains barely used in the current operating systems. Typically, the force sensing capabilities of the state-of-the-art input devices remain unexploited during the autoscroll operations, on both desktop and mobile computers, leaving room for including ForceEdge in current operating systems without interfering with the other interaction techniques.

Moreover, ForceEdge remains compatible with the alternatives that can be used for performing two-points operations, such as using Shift+click on desktop computers, or Push-Push [14, 15] on smartphones, so that the user can freely choose the most appropriate technique for the task at hand.

Accidental activation

One theoretical benefit of ForceEdge that we did not evaluate is its robustness toward accidental activations of autoscroll. A proper evaluation of this would have required a dedicated experiment, possibly asking participants to move an object between overlapping windows. We decided to focus our evaluations on the overall performance of ForceEdge and leave the evaluation to this specific aspect to future work. Moreover, when scrolling rate increases with the force applied on the device, an accidental activation of autoscroll is likely to be at one of the lowest possible rate, resulting in a limited accidental scrolling which is of relatively low cost to the user. Finally, participants in the experiments did not mention any specific problem regarding the accidental activation.

CONCLUSION

We have presented ForceEdge, a novel autoscroll technique which exploits the force-sensing capabilities of state-of-the-art input devices. ForceEdge does not suffer from the limitations of conventional autoscroll techniques and can be applied to both desktop and mobile computers. We compared ForceEdge to the current system baselines on the macOS and iOS platforms, and our results show that ForceEdge improves over these baselines for top-to-bottom select and move tasks.

ACKNOWLEDGMENTS

This work was partially supported by ANR (TurboTouch, ANR-14- CE24-0009).

REFERENCES

1. J. Aceituno, S. Malacria, P. Quinn, N. Roussel, A. Cockburn, and G. Casiez. 2017. The design, use, and performance of edge-scrolling techniques. *International Journal of Human-Computer Studies* 97 (2017), 58 – 76. DOI:http://dx.doi.org/10.1016/j.ijhcs.2016.08.001

2. Apple. 2014. ForceTouch Trackpad. (9 September 2014). http://ns.inria.fr/mjolnir/forceedge/forcetouch.pdf, retrieved December 21st, 2016 from https://support.apple.com/en-us/HT204352.

3. Apple. 2015a. iPhone6S. (25 September 2015). http://ns.inria.fr/mjolnir/forceedge/iphones.pdf, retrieved December 21st, 2016 from http://www.apple.com/iphone/compare/.

4. Apple. 2015b. Magic Trackpad 2. (13 October 2015). http://ns.inria.fr/mjolnir/forceedge/magictrackpad.pdf, retrieved December 21st, 2016 from http://www.apple.com/magic-accessories/.

5. Apple. 2015c. Video explaining 3D Touch on iPhones. (25 September 2015). http://ns.inria.fr/mjolnir/forceedge/3dtouch.mp4, Retrieved December 21st, 2016 from http://www.apple.com/.

6. Mathias Baglioni, Sylvain Malacria, Eric Lecolinet, and Yves Guiard. 2011. Flick-and-brake: Finger Control over Inertial/Sustained Scroll Motion. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. ACM, New York, NY, USA, 2281–2286. DOI: http://dx.doi.org/10.1145/1979742.1979853

7. Didier D. Bardon, Richard E. Berry, Shirley L. Martin, S. A. Morgan, John M. Mullay, and Craig A. Swearingen. 1997. Method for Auto-Scroll with Greater User Control. *IBM Technical Disclosure Bulletin* 40, 6 (June 1997), 181–182. http://ip.com/IPCOM/000118763

8. Joseph D. Belfiore, Christopher J. Guzak, Christopher E. Graham, Stepehn M. Madigan, Tandy W. Trower, II, Randall L. Kerr, and Adrian M. Wyard. 1998. Auto-scrolling during a drag and drop operation. (1998). http://www.google.com/patents/US5726687

9. Richard E. Berry, Stephen S. Fleming, and A. C. Temple. 1991. Auto-Scroll During Direct Manipulation. *IBM Technical Disclosure Bulletin* 33, 11 (April 1991), 312.

10. Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012. The Fat Thumb: Using the Thumb's Contact Size for Single-handed Mobile Interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 39–48. DOI:http://dx.doi.org/10.1145/2371574.2371582

11. Géry Casiez and Daniel Vogel. 2008. The Effect of Spring Stiffness and Control Gain with an Elastic Rate Control Pointing Device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1709–1718. DOI: http://dx.doi.org/10.1145/1357054.1357321

12. Géry Casiez, Daniel Vogel, Qing Pan, and Christophe Chaillou. 2007. RubberEdge: Reducing Clutching by Combining Position and Rate Control with Elastic Feedback. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 129–138. DOI: http://dx.doi.org/10.1145/1294211.1294234

13. Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. GripSense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 545–554.

14. Jaehyun Han, Sunggeun Ahn, and Geehyuk Lee. 2014. Push-push: A Two-point Touchscreen Operation Utilizing the Pressed State and the Hover State. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST'14 Adjunct)*. ACM, New York, NY, USA, 103–104. DOI:http://dx.doi.org/10.1145/2658779.2658797

15. Jaehyun Han and Geehyuk Lee. 2015. Push-Push: A Drag-like Operation Overlapped with a Page Transition Operation on Touch Interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 313–322. DOI: http://dx.doi.org/10.1145/2807442.2807457

16. Henning Henningsen, Bettina Ende-Henningsen, and Andrew M Gordon. 1995. Contribution of tactile afferent information to the control of isometric finger forces. *Experimental brain research* 105, 2 (1995), 312–317.

17. Seongkook Heo and Geehyuk Lee. 2011. Force gestures: augmenting touch screen gestures with normal and tangential forces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 621–626.

18. Elizabeth Househam, John McAuley, Thompson Charles, Timothy Lightfoot, and Michael Swash. 2004. Analysis of force profile during a maximum voluntary isometric contraction task. *Muscle & nerve* 29, 3 (2004), 401–408.

19. Lynette A Jones. 1989. Matching forces: constant errors and differential thresholds. *Perception* 18, 5 (1989), 681–687.

20. Lynette A Jones. 2000. Visual and haptic feedback in the control of force. *Experimental brain research* 130, 2 (2000), 269–272.

21. Lynette A Jones and Ian W Hunter. 1983. Perceived force in fatiguing isometric contractions. *Perception & Psychophysics* 33, 4 (1983), 369–374.

22. Lynette A Jones and Susan J Lederman. 2006. *Human hand function*. Oxford University Press.

23. Andrew Kwatinetz. 1996. Scrolling contents of a window. (1996). http://www.google.com/patents/US5495566

24. Shih-Gong Li and Theodore Jack London Shrader. 1998. Scrolling a target window during a drag and drop operation. (1998). http://www.google.com/patents/US5740389

25. Sylvain Malacria, Jonathan Aceituno, Philip Quinn, Géry Casiez, Andy Cockburn, and Nicolas Roussel. 2015. Push-Edge and Slide-Edge: Scrolling by Pushing Against the Viewport Edge. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2773–2776. DOI: http://dx.doi.org/10.1145/2702123.2702132

26. Takashi Miyaki and Jun Rekimoto. 2009. GraspZoom: Zooming and Scrolling Control Model for Single-handed Mobile Interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09)*. ACM, New York, NY, USA, Article 11, 4 pages. DOI: http://dx.doi.org/10.1145/1613858.1613872

27. Xiao-Dong Pang, Hong Z Tan, and Nathaniel I Durlach. 1991. Manual discrimination of force using active finger motion. *Perception & psychophysics* 49, 6 (1991), 531–540.

28. Gonzalo Ramos and Ravin Balakrishnan. 2005. Zliding: Fluid Zooming and Sliding for High Precision Parameter Manipulation. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05)*. ACM, New York, NY, USA, 143–152. DOI: http://dx.doi.org/10.1145/1095034.1095059

29. Joseph D. Rutledge and Ted Selker. 1990. Force-to-motion Functions for Pointing. In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction (INTERACT '90)*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 701–706. http://dl.acm.org/citation.cfm?id=647402.725310

30. Elaine R Serina, Eric Mockensturm, CD Mote, and David Rempel. 1998. A structural model of the forced compression of the fingertip pulp. *Journal of biomechanics* 31, 7 (1998), 639–646.

31. Jacob J Sosnoff and Karl M Newell. 2005. Intermittent visual information and the multiple time scales of visual motor control of continuous isometric force production. *Perception & psychophysics* 67, 2 (2005), 335–344.

32. Daniel Spelmezan, Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. 2013. Side Pressure for Bidirectional Navigation on Small Devices. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 11–20. DOI:http://dx.doi.org/10.1145/2493190.2493199

33. Hong Z Tan, Mandayam A Srinivasan, Brian Eberman, and Belinda Cheng. 1994. Human factors for the design of force-reflecting haptic interfaces. *Dynamic Systems and Control* 55, 1 (1994), 353–359.

34. Graham Alasdair Wilson. 2013. *Using pressure input and thermal feedback to broaden haptic interaction with mobile devices*. Ph.D. Dissertation. University of Glasgow.

35. S. I. Yaniger. 1991. Force Sensing Resistors: A Review Of The Technology. In *Electro International, 1991*. 666–668. DOI: http://dx.doi.org/10.1109/ELECTR.1991.718294

36. Shumin Zhai. 1995. *Human performance in six degree of freedom input control*. Ph.D. Dissertation. University of Toronto.