Side Pressure for Bidirectional Navigation on Small Devices

Daniel Spelmezan, Caroline Appert, Olivier Chapuis, Emmanuel Pietriga

Side Pressure for Bidirectional Navigation on Small Devices

Daniel Spelmezan[1,2] **Caroline Appert**[2,1] **Olivier Chapuis**[2,1] **Emmanuel Pietriga**[1,3]

[1] Inria
91405 Orsay, France

[2] Univ Paris-Sud & CNRS (LRI)
91405 Orsay, France

[3] Inria Chile (CIRIC)
7561211 Santiago, Chile

ABSTRACT

Virtual navigation on a mobile touchscreen is usually performed using finger gestures: drag and flick to scroll or pan, pinch to zoom. While easy to learn and perform, these gestures cause significant occlusion of the display. They also require users to explicitly switch between navigation mode and edit mode to either change the viewport's position in the document, or manipulate the actual content displayed in that viewport, respectively. SidePress augments mobile devices with two continuous pressure sensors co-located on one of their sides. It provides users with generic bidirectional navigation capabilities at different levels of granularity, all seamlessly integrated to act as an alternative to traditional navigation techniques, including scrollbars, drag-and-flick, or pinch-to-zoom. We describe the hardware prototype, detail the associated interaction vocabulary for different applications, and report on two laboratory studies. The first shows that users can precisely and efficiently control SidePress; the second, that SidePress can be more efficient than drag-and-flick touch gestures when scrolling large documents.

Author Keywords

Mobile device; Side pressure; Pressure input; Pressure-based interaction; Single-handed interaction; Scrolling task.

ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: User Interfaces - Graphical user interfaces

INTRODUCTION

Many handheld devices are now equipped with a touchscreen that covers most of their front side, enabling more information to be displayed simultaneously. Despite recent advances in display technology that enable screen resolutions nearing the limits of human visual acuity, handheld devices still cannot accommodate maps or long lists in their entirety, or book pages displayed at a scale where text is legible. Scrolling and other virtual navigation capabilities such as zooming remain essential features, typically invoked through tapping, dragging, flicking and pinching. These gestures suffer from two strong limitations: first, navigating in a document entails occluding a significant part of that document; second, most (if not all) events performed on a document are mapped to navigation actions, requiring users to perform mode switches via software buttons, multi-finger taps, or techniques like Bezel-Swipe [17], to manipulate content in the current viewport.

SidePress alleviates these problems by augmenting a mobile device with two continuous pressure sensors, co-located on one of its sides. Recent concept phones like Synaptics' Fuse and NTT DoCoMo's Grip UI use side pressure input only for invoking commands. In this paper, we show how side pressure input is particularly well-suited for precise and efficient bi-directional navigation in one dimension. We focus on two low-level generic tasks performed frequently on mobile devices: navigating a large document (scrolling, zooming), and selecting a value in a continuous range, as when navigating to a given time position in a video, or when adjusting audio volume and screen brightness, i.e., actions usually performed by direct manipulation of a slider.

Thanks to its two sensors, SidePress provides bi-directional navigation capabilities at different levels of granularity, all seamlessly integrated: continuous rate-based control (pressure intensity), jumps of different amplitudes (*light-click* or *strong-click*), jump to the minimum or maximum values in the value's range (*strong-press*). Altogether, these events provide users with the same level of expressiveness as a traditional scrollbar, but without cluttering the screen, and without causing visual occlusion or interfering with touch input, as control gets delegated to sensors outside the display area. Users can also press both sensors simultaneously, an additional event that can be useful for, e.g, holding a call, switching between navigation and editing modes, or displaying a control panel to invoke a command or to jump to a bookmark.

We first give an overview of related work. Next, we present our hardware prototype and the algorithm we implemented to recognize the vocabulary of pressure events used to control bidirectional navigation. We then report on two laboratory studies. The first study evaluates the accuracy of our recognizer and provides empirical evidence that users can control the set of pressure events we propose. The second study compares SidePress with classic drag-and-flick touch interaction in a target acquisition task using scrolling in a single-handed context. This experiment shows that SidePress can be faster than touch when navigating to distant targets. We conclude with directions for future work.

BACKGROUND AND MOTIVATION

Several recent research projects have investigated the coupling of touch location and pressure input on handheld devices. For instance, buttons can be turned into multi-level

D. Spelmezan, C. Appert, O. Chapuis, and E. Pietriga. Side Pressure for Bidirectional Navigation on Small Devices. In MobileCHI '12: Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services, 10 pages, to appear, ACM, 2013.

buttons by measuring the pressure applied to them to, e.g., facilitate text entry [3, 6, 12], or control zooming and 2D scrolling using a phone's keypad [6]. Pressure can also be used to give different semantics to taps and drags to support advanced navigation: Force Gestures [9] rely on a prototype device whose screen is covered with a film plate linked to several pressure sensors that capture normal and tangential forces applied on the touch location. This turns taps and drags into richer navigation actions like go to previous/next page or scroll to top/bottom. Another example is GraspZoom [13], a technique that relies on a pressure sensor located on the back of the device. By combining touch on the front and pressure applied on the back, users are able to scroll and zoom with only one hand. Changes in direction (up/down) and switches between zooming and scrolling modes require users to perform touch gestures before applying pressure and/or to grasp the device in another way so as to apply pressure at a different location. One limitation of these two projects [9, 13] is that the simultaneous use of touch and pressure input causes visual occlusion, which may hinder usability and performance for virtual navigation tasks, as those rely on a tight perception-control loop [7].

SidePress is inspired by device prototypes that enrich the vocabulary of events by delegating control to other sensors than the tactile screen. Researchers have previously considered: using tilt orientation, captured via built-in accelerometers and gyroscopes (e.g. [8, 10, 14]); turning the device's back side into a tactile input surface [2, 26]; and adding proximity sensors to the sides of the device [4]. However, none of these approaches provide a fully satisfying solution for navigation in arbitrary contexts. Using proximity sensors, or the rear of the device, is too demanding to effectively support the frequent single-handed use situations [11]. Tilting to control scrolling addresses finger occlusion, but introduces other issues: the technique is prone to overshooting, as many first-order controls, and the device's orientation relative to the user's line of sight can make the display hard to read when tilted too much.

Peripheral pressure sensors have also been used to augment other devices, such as PressureMove and PressureFish [18, 19], that attach a pressure sensor to the side of a mouse. Another example is the tablet prototype presented in [22]: a pressure sensor is attached to the front bezel held with the dominant hand. As most previous work that investigated pressure as an input channel [20, 24], they considered a single pressure sensor, which limits the range of tasks users can achieve with it. Indeed, a pressure sensor is typically unidirectional and has a rest position. This makes it fit for command selection on a menu of moderate size, but not for navigation in a large range of values (e.g., sliders or scrollbars).

The need for bidirectional control motivates the use of two pressure sensors, an approach that Cechanowicz *et al.* investigated when designing their pressure mouse [5]. They built a mouse prototype augmented with one sensor dedicated to the thumb and another one dedicated to the index finger. Other projects have considered using even more pressure sensors. Harrison *et al.* [8] were probably the first to consider such hardware configurations with their handheld device that combines left, right and back 2-level pressure sensors to detect users' handedness. Very recently, Wilson *et al.* investigated the physiological limits of such an approach by designing a mobile device augmented with seven continuous pressure sensors distributed on the sides and back of the device [23]. The placement of the different sensors is driven by the typical hand posture right-handed users adopt when holding the device with a single hand in portrait mode. In a study, participants had to select one pressure level among six, using either a single sensor or a group of sensors.

Previous work mentioned in this section considered pressure applied via a pen, on a mouse, directly on the tactile surface or on the device's periphery for mostly unidirectional control. Those do not allow us to predict users' performance for bidirectional input with two pressure sensors located on one side of a smartphone, as controlling pressure input strongly depends on the user's hand posture and on the muscles involved [23]. However, these projects provide useful insights into the use of pressure as an input modality. First, there is evidence that when using pressure input, rate-based control outperforms positional control [22]. Second, pressure interaction remains robust and usable while walking [3, 22]. Third, these studies are in accordance with earlier ones about pen pressure [15, 16], that highlighted the importance of feedback mechanisms for efficient pressure control [24]. Finally, two-sided (grasping type) pressure input is preferable to single-sided (pointing type) in a mobile context [20]. The design of the SidePress hardware prototype and the associated software was informed by these experimental findings.

HARDWARE DESIGN AND REALIZATION

The SidePress prototype (Figure 1) lets users apply pressure onto the top sensor (A), the bottom sensor (B), or both simultaneously. Like GraspZoom [13] and [23], SidePress uses two-sided (grasping type) pressure input, which is preferable to single-sided (pointing type) pressure input in a mobile context [20]. However, unlike GraspZoom, we overcome occlusion problems by squeezing the device along its y-axis rather than pinching it along its z-axis.

The SidePress prototype shown in Figure 1 consists of three components: an iPod touch 4G (iOS 5.0), a custom-designed plastic casing that hosts two force-sensitive resistors (Interlink Electronics FSR 400), and a custom-built circuit board.

The casing is made of polyamide, produced by a 3D printer. One side of the casing has two edges that are separated by a gap of 1.5mm. The active area of the FSR is located inside this gap. Each sensor's tail lies horizontally on the rear face of the casing. The head is bent upwards such that the active area remains between the two edges. To enable users to adopt comfortable grip postures when operating the device, we use a thin piece of rubber between the sensor and the inner edge. This piece extends upward and downward beyond the surface of the sensor. With this layout, the forces that occur in the sensor's proximity get translated to the sensor's active area. The latter thus remains responsive even if the main pressure point exerted onto the casing does not coincide with the sensor's active area, which is only 5mm in diameter.

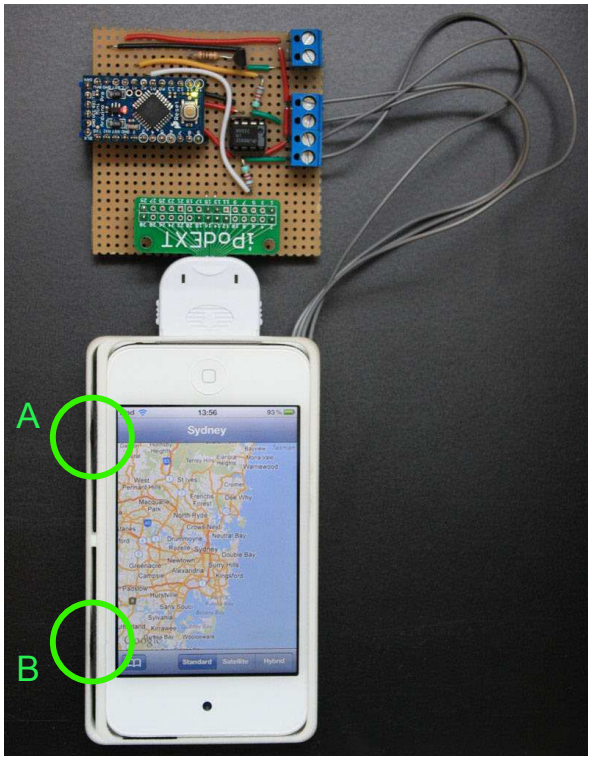


Figure 1. The SidePress prototype has two pressure sensors on one of its sides, providing a rich input vocabulary for interaction.

Force-sensitive resistors do not have a linear resistance vs. force characteristic. For linear increases in force, the output voltage is not linear. This issue was discussed in [20], that recommends a linear mapping function for pressure input. In order to linearize pressure input, we used an op-amp based current-to-voltage converter circuit [20]. With this circuit, sensor output increases linearly with applied pressure.

The circuit board is powered by the iPod touch and hosts an Arduino Pro Mini (`http://arduino.cc`). The Arduino samples sensor data at 60 Hz and 10 bit resolution, low-pass filters the data (moving average of the last five samples), and sends the preprocessed data to the iPod over its serial port. The iPod further processes and normalizes the sensor data, and runs an algorithm for recognizing pressure input.

Figure 2 shows the two symmetric casings we have built to support both handedness and two different hand postures. Figure 2-(a) illustrates a right-handed user applying pressure with the two groups of digits <index, middle> and <ring, little>, that have been identified as particularly efficient for pressure control [23]. Figure 2-(b) illustrates the same user applying pressure with her thumb and palm. We could imagine having a casing with four sensors, two on each side, and activating the appropriate sensors and corresponding actions based on handedness and hand posture [25].

SOFTWARE EVENTS AND INTERACTION TECHNIQUES

The hardware prototype is driven by software that translates variations of both pressure sensor values into interaction events. Figure 3 illustrates the state machine corresponding to

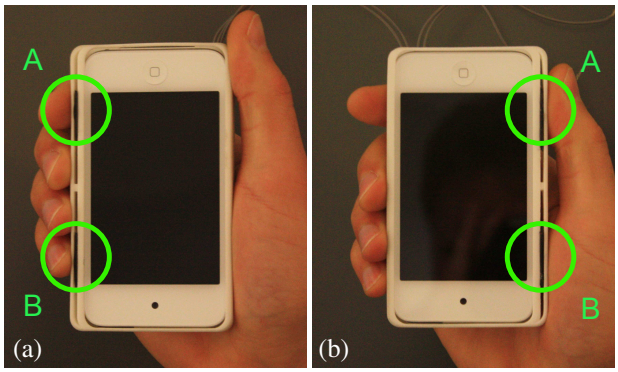


Figure 2. Pressure Sensors can be positioned on either side of the device: sensors actuated by the fingers (a), or by the thumb and palm (b).

one sensor. Two such machines run in parallel, one for each of the two sensors A and B. Whenever the user actuates one of the sensors, i.e., as soon as one of the machines leaves the `Idle` state, the other machine gets disabled.

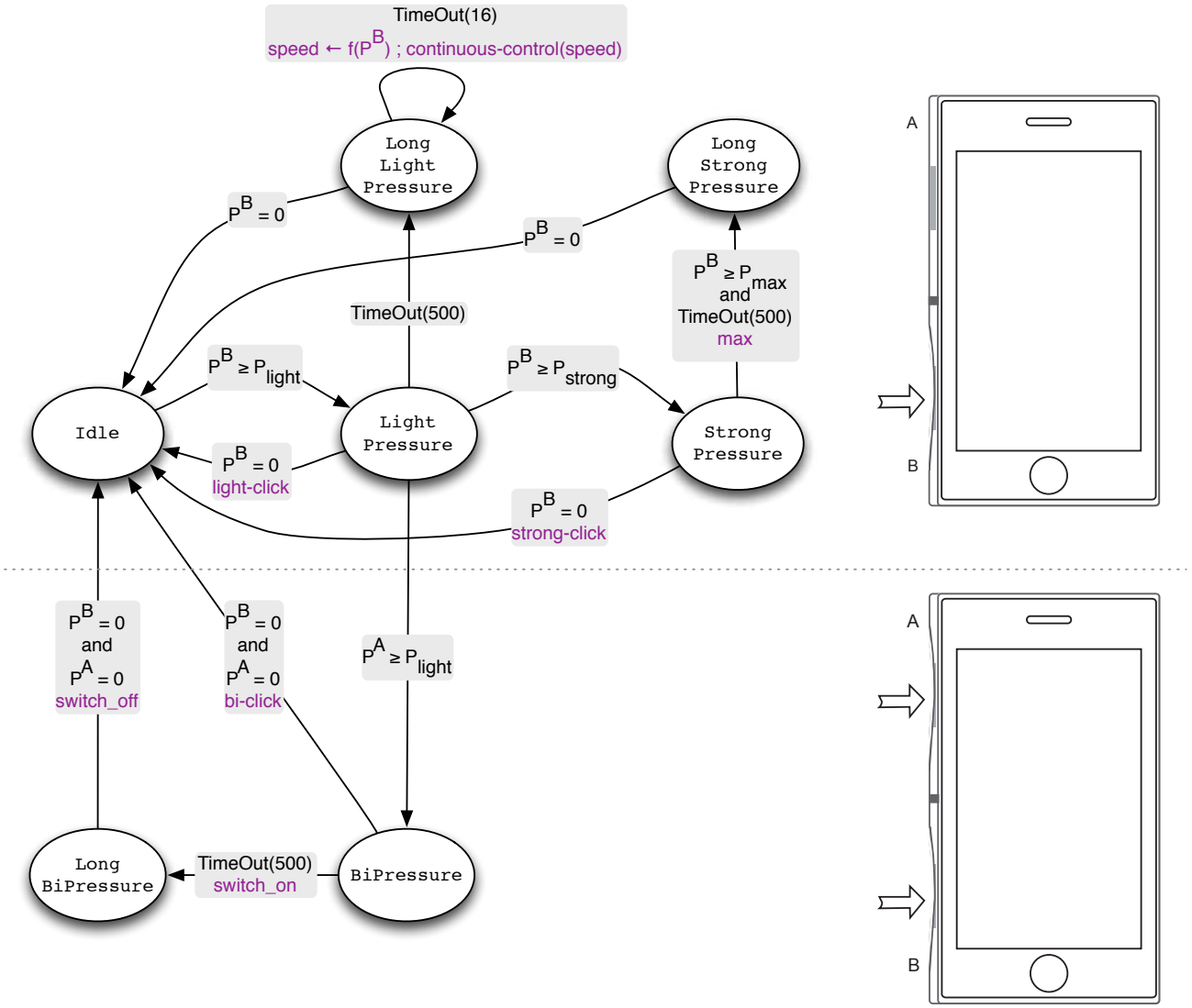
In the following, we describe the interaction techniques enabled by these state machines using a simple example: navigating a document. Other examples follow, based on the exact same generic state machines, whose events can be mapped to different actions (see Table 1 and Figure 4).

When navigating a document, the two sensors provide a set of actions seamlessly integrated together to move upward (sensor A) and downward (sensor B). When the user starts pressing one of the sensors, e.g., sensor B, the corresponding machine goes into the `LightPressure` state and waits a short amount of time (500ms) for one of the following events to occur depending on what the user inputs.

- She can release sensor B. This takes the machine back in the `Idle` state, and triggers a *light-click* event that translates the document downward by one line.
- She can apply a stronger level of pressure on sensor B. This takes the machine in the `StrongPressure` state. From there, either she releases the sensor within 500ms, which triggers a *strong-click* event that takes her to the next page; or she keeps the sensor pressed for more than 500ms, which triggers a *max* event that takes her to the last page of the document.
- She can press sensor A (in addition to B). This takes the machine in the `BiPressure` state. From there, either she stops applying pressure within 500ms, which triggers a *bi-click* event that bookmarks the current position in the document; or she keeps them pressed for more than 500ms, which triggers a *switch-on* event that toggles another mode and takes the machine in the `LongBiPressure` state until the sensors get released (*switch-off*). In our example, toggling to the other mode pops-up a list of previously-bookmarked locations in the document, that can be selected, e.g., using direct touch or by tilting the device.

If none of the above three events happen within the first 500ms after the user started pressing sensor B, the machine enters state `LongLightPressure`, which allows for pressure-dependent *continuous* control. In our example, this translates to rate-based continuous downward scrolling.

Two state machines run in parallel, one for the upper sensor (P^A) and one for the lower sensor (P^B). Whenever a machine leaves the Idle state, it disables the other machine, which gets re-enabled when the former gets back to the Idle state.

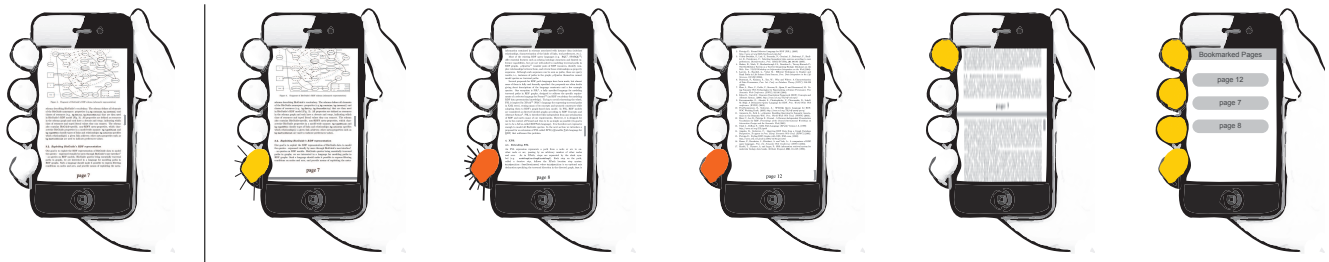


$P_{light} = 1$, $P_{strong} = 5$ and $P_{max} = 6$ (approximate force in Newton)

Figure 3. State machine for sensor B.

EVENTS	ABSTRACT ACTIONS	SCROLLING A DOCUMENT	NAVIGATION IN A VIDEO	TEXT SELECTION
light-click A/B	increase/decrease by one unit	go to the previous/next line	go to the previous/next minute	go to previous/next character
strong-click A/B	increase/decrease by one step/gradation	go to the previous/next page	go to the previous/next chapter	go to previous/next word
max A/B	jump to the minimum/maximum value	go to the first/last page	go to the start/end of the video	go to start/end of sentence
continuous A/B	increase/decrease continuously	continuous up/down scrolling*	fast-forward/backward*	move caret forward/backward*
		*(speed depends on pressure level)		
bi-click	invoke a command	bookmark current page [1]	bookmark current time index	caret / selection handles switch
switch-on/off	enter/leave a mode	display list of bookmarked pages†	display list of bookmarked time indexes†	display text editing menu
		†(selection of an item using, e.g., direct touch or tilt, takes to the corresponding position)		

Table 1. Events, corresponding abstract actions, and three example mappings to navigation actions



idle
from page 7…

soft-click (B)
moving to
next line

strong-click (B)
moving to
next page

max (B)
moving to
last page

continuous (A)
rate-based
scrolling (upward)

switch-on
display
bookmarks

Figure 4. Application to navigation in a document

The same events performed on sensor A translate to upward navigation actions in the document: up by one line, previous page, first page, continuous rate-based upward scrolling.

We designed the above state machines for general unidimensional bidirectional navigation. Table 1 gives examples of mappings between events and the corresponding actions in different contexts. The mappings are designed so that all events form a coherent set of actions that are tightly integrated, easy to relate to one another, and learn. To achieve this, we assign actions according to the amount of pressure applied to the sensors, which naturally maps to navigation speed. This is obvious in the case of *continuous* rate-based control (the stronger users press, the faster they scroll), but also applies to discrete navigation actions: a *light-click* moves by one unit, a *strong-click* by one gradation, a long strong press (*max*) to the range's start/end.

SidePress is particularly well-suited for navigating large collections of items, such as the pages of a long text document, collections of pictures, a feature film or any other long video. It can also be useful when precisely adjusting the value controlled by any kind of slider. Sliders are usually operated by direct manipulation of the knob with the finger on the touch screen. This creates visual occlusion and makes it challenging to precisely set a value. With SidePress, a *light-click* moves the knob of the selected slider by one unit in the direction corresponding to the actuated sensor; a *strong-click* takes the slider to the previous/next gradation (tick mark); a *max* event sets it to its minimum/maximum; a long (initially light) pressure applied to the sensor enables *continuous* rate-based control of the knob.

SidePress can also be used to select and edit text. Doing this usually requires dragging a caret with a finger to position it in-between characters. The finger occludes a significant part of the text, and despite the availability of helpers such as the iPhone's small lens that displays the content hidden by the finger in a callout above the fingertip [21], selecting text and precisely positioning the caret for insertion and editing remains a tedious task. With SidePress, users can quickly and precisely move the caret without occluding the corresponding text. See Table 1 for more detail.

Commands and Mode Switch

As mentioned earlier, users can perform additional actions by actuating both sensors simultaneously. Pressing both sensors at the same time consists in squeezing the device, and is a relatively natural gesture. Though it has no obvious mapping to actual navigation actions, this gesture can be associated with actions related to navigation.

The gesture is interpreted either as a *bi-click* if users release the sensors less than 500ms after press, or as a mode switch if they keep them pressed for a longer period, triggering a *switch-on/off* (lower part of Figure 3). These two different events can be associated with the invocation of a command, like bookmarking or undoing an action, or entering/leaving another mode, respectively.

Mappings to concrete actions are highly dependent on the context and current application. For instance, when navigating a document, bookmarking the current position can be a useful feature [1]. A *bi-click* can be used for that purpose. The list of recent bookmarks can then be displayed by a *switch-on*. When adjusting the value of a slider, a *bi-click* can set the current value as the default one. When navigating in a map, a *bi-click* can drop a pin at the current location. Mode *switch-on/off* events can toggle between navigation and edit mode, letting users manipulate content, rather than the viewport, with direct touch. Various applications feature different modes (selection, navigation, editing, …); this event can have broad applicability beyond maps: mode *switch-on/off* events can be used to temporarily enter a second mode (while the sensors are pressed), which would give a different meaning to the direct touch interactions performed on the display.

EVALUATING SIDEPRESS VOCABULARY OF EVENTS

We designed an experiment to evaluate users' ability to trigger, on demand, each of the ten events defined in the previous section. The experiment followed a 2×10 within-participant design with the following factors:

- HANDCTRL: the part of the hand the pressure sensors are in contact with, *Fingers* and *Palm* (Figure 2);
- EVENT: the ten events, i.e., four events {*light-click*, *strong-click*, *max*, *continuous*} for each of the two sensors (4×2) plus two events {*bi-click*, *switch-on/off*} that involve the simultaneous use of both sensors.

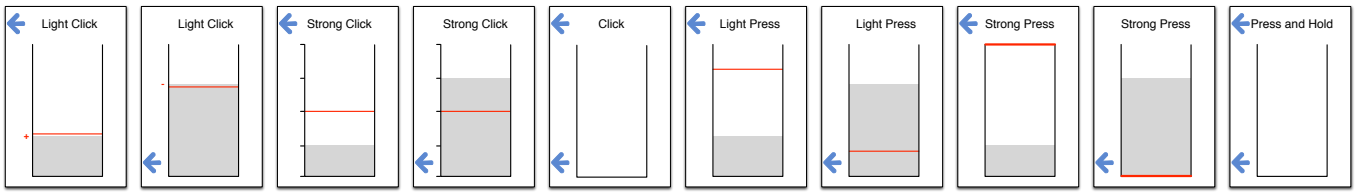


Figure 5. The ten experimental conditions for HANDCTRL = *Fingers*

Experimental task

A trial consists in triggering one of the ten events. Because simply displaying the name of the event as a stimulus would have been too artificial and cognitively demanding, we conveyed the stimulus information as follows (Figure 5):

- textual instructions indicate the type of action participants have to perform (Light Click, Strong Click, Light Press, Strong Press, Click, Press and Hold);
- one (or two) arrow(s) indicate which sensor(s) should be actuated;
- a tank containing some liquid, whose level varies depending on the actions performed.

The *tank filling* metaphor is intended to reinforce the stimulus and make it more explicit. A *light-click* adds (A) or removes (B) one volume unit. A *strong-click* sets the volume to the closest upper tick (A) or to the closest lower tick (B). A *max* event fills (A) or empties (B) the tank. A *switch-on/off* makes the tank disappear; the participant then has to wait for a Release instruction that pops up 2 seconds later.

Simply displaying the current level and the target level to reach would have been too ambiguous. Indeed participants could have chosen non optimal strategies to reach the intended goal, triggering multiple events to eventually reach the same result (liquid level). For example, instead of continuously filling the tank in response to the *continuous*(A) stimulus, participants could have fully filled the tank (*max*(A)) and then gradually removed liquid until it had reached the target level (*continuous*(B)). To reduce the number of misinterpretations of the given instuctions, we reinforced the differences between event conditions by adding specific visual elements. The *light-click* conditions display a small plus (+) or minus (−) sign close to the current level. The *strong-click* conditions display tick marks, and the *max* conditions use a thicker line for showing the target level to reach.

A trial ends as soon as participants input the event that matches the stimulus. For *continuous* events, the trial stops when they release the pressure sensor after *continuous* control. *Continuous* at full pressure fills the tank in 1.67s. The distance between the current level and the target level is then recorded (the initial distance is set to 50 units in all *continuous* conditions). All input events are recorded so as to analyze the number and type of errors made. Some incorrect sequences of events can lead to a state where participants can no longer enter the right event without making more errors. In such cases[1], the screen flashes, an audio beep is played, and the tank is set back to its initial level. Audio feedback is enabled during the whole experiment. Each type of event triggers a different built-in sound like a *click* sound for a *light-click* or a *camera shutter* sound for a *bi-click*.

[1]This happens for A when the target level is below the current level, and for B when the target level is above the current level.

Procedure

At the very beginning of the experiment, the operator describes all possible events. Participants freely practice filling the tank for a maximum of ten minutes, getting familiar with the prototype. Trials are blocked by HANDCTRL, half of the participants starting with the *Fingers* condition, the other half with the *Palm* condition. A HANDCTRL block is split into four sub-blocks. Each sub-block contains three repetitions of each of the ten events (30 trials) presented in a random order. The first sub-block is for training purposes in actual experimental conditions. Participants perform the tasks, the operator providing additional instructions. At the end of the experiment, participants have to fill a questionnaire in which they indicate their preferences, rate task difficulty, and report an estimation of how many errors they think were due to a misinterpretation of the instruction, as opposed to performing the wrong action.

Participants & Apparatus

Twelve unpaid volunteers (3 females), age 22 to 39 year old (average 29.8, median 28), all right-handed, participated in the experiment using the prototype described earlier. Eight participants were daily users of smartphones, the remaining four only used one from time to time. The experiment lasted approximately 40 minutes.

Results

The main measure of interest is the rate of success, i.e., the percentage of trials where participants successfully input the stimulus event at first try. The data collected also contains a count of each EVENT type that occurred in each trial. This enables us to study the kind of errors participants made and assess the associated cost. Although measuring users' accuracy with *continuous* control was not the focus of this study, we were also interested in getting a rough estimation of it. We thus logged the distance between the level reached and the target level at the end of *continuous* trials.

First of all, we checked that there was neither a significant effect of the presentation order of HANDCTRL conditions on success rate (odd participants started with *Palm*), nor a learning effect for each HANDCTRL condition (by comparing the three measured sub-blocks). This confirms that participants got enough training in the context of this experiment and had reached a stable performance level before being recorded.

	light-click A	*light-click* B	*strong-click* A	*strong-click* B	*max* A	*max* B	*continuous* A	*continuous* B	*bi-click*	*switch*
light-click A	95.6 (95.4)	0	1.77	0	0	0	2.65	0	0	0
light-click B	0	76.6 (78.7)	0	5.67	0	0.71	0	17.02	0	0
strong-click A	3.91	0	84.4 (81.5)	0	1.56	0	6.25	0	3.91	0
strong-click B	0	3.39	0	91.5 (90.7)	0	0	0	1.69	3.39	0
max A	0	0	2.48	0	89.3 (87.9)	0	8.26	0	0	0
max B	0	0.81	0	3.23	0	87.1 (88.9)	0	7.26	0	1.61
continuous A	1.72	0.86	0	0	4.31	0	93.1 (92.6)	0	0	0
continuous B	0	2.65	0	0	0	0.88	0.88	95.6 (95.4)	0	0
bi-click	1.64	0	0.82	0.82	0	0	2.46	1.64	88.5 (90.7)	4.1
switch	0.88	0	0	0	0	0	1.75	0	2.63	94.7 (94.4)

Table 2. Experimental results, considering the best HANDCTRL condition only. The left column is the stimulus event. A line represents the distribution (percentage) of all events participants performed as a response to this stimulus, by type of event. Because participants did have to input the right event to end a trial, the reported percentages are slightly different from the rate of success (reported in brackets).

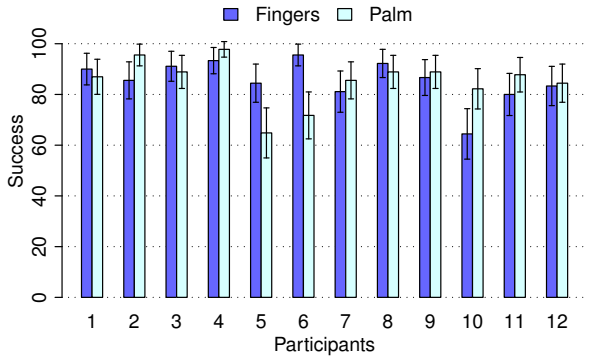


Figure 6. Success rate for both HANDCTRL conditions, by participant.

Figure 6 shows the success rate for each participant for both *Fingers* and *Palm* conditions. We can observe that for most participants performance is similar between the two conditions. Overall success rates are 85.6% for *Fingers* and 85.3% for *Palm*, respectively. An ANOVA reveals that the effects of HANDCTRL and HANDCTRL $\times$ EVENT on rate of success are not significant.

Three participants have more contrasted results between the two HANDCTRL conditions: participants 5 & 6 have far better results with *Fingers* while participant 10 follows the inverse tendency. Comments collected at the end of the experiment are in accordance with these quantitative results; participants were able to identify which condition they performed best in. For example, participant 5 reported having great difficulty to control sensor B with the lower part of his palm. Participant 10, who has a small hand, said that applying strong pressure with the fingers was conflicting with the device grip. The only surprising result is for participant 6, who reported preferring the *Palm* condition. Otherwise, when performances are comparable, participants tend to prefer the *Palm* condition. Ten participants out of twelve ranked the latter first in terms of preference.

In the following, for the sake of clarity, we only consider the best HANDCTRL condition for each participant. This is possible because the success rates for *Fingers* and *Palm* are very similar (the effect of HANDCTRL $\times$ EVENT on success rate is not significant), and because considering the best HANDCTRL condition makes sense in the context of the more elaborate prototype envisioned earlier, that is equipped with two sensors on each side, one side only being activated at any given time, based on handedness and preferred hand posture.

We obtain a success rate of 90%. This is a very encouraging result given that our recognizer is a basic one, that does not use any user-specific calibration. Moreover, despite the care taken to show the different stimuli, a substantial number of errors were apparently due to a misinterpretation of the instruction. For example, participants tended to confound the words press and click in the textual instructions. Using the participants' answers to the final questionnaire, we roughly estimate that about half of the errors are such cognitive errors rather than actual motor errors. For instance, 3 participants (7, 9 & 12) reported that, in more than 75% of error situations, they actually misinterpreted the instruction.

Table 2 helps understand the types of errors participants made. It reports the percentage of each type of event actually performed when instructed to perform one particular event. We observe some significant differences: EVENT has a significant effect on success rate ($F_{9,99} = 2.00$, $p = 0.0475$, $\eta^2 = 0.14$), a large effect size as indicated by the η^2 value. A post-hoc t-test with Holm correction[2] shows (i) that *light-click*(B) has a significantly ($p < 0.05$) worse success rate than *light-click*(A) and *continuous*(B); and (ii) that *strong-click*(A) has a marginally ($p < 0.1$) worse success rate than *bi-click*.

Light-click(B) errors are mostly due to a confusion with *continuous*(B). Participants were not fast enough to release the pressure sensor. However, the cost of this type of error in a real context of use is very low, since a short *continuous*(B) has an effect very similar to that of a *light-click*(B) in SidePress' interaction model. A confusion between, e.g., *light-click*(B) and *max*(B) would have been much more problematic; but this error was very rare.

Participants tended to do a *continuous*(A) or a *light-click*(A) instead of a *strong-click*(A). This also has little practical consequences since users can achieve the effect of a *strong-click*(A) event through a sequence of *continuous*(A) and/or *light-click*(A) events. However, participants sometimes input a *bi-click* instead of a *strong-click*(A) or a *strong-click*(B), which can be more problematic as it could for example bookmark a value instead of navigating within the range.

[2]There are too many conditions to use a Bonferroni correction.

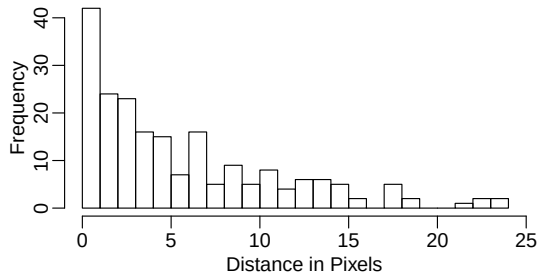


Figure 7. Distance to the target with *continuous* A and B (97.5% of the best data). 1 mm $\sim$ 13 pixels .

Most of the non-marginal errors left in Table 2, e.g., input *continuous*(A) instead of *max*(A), have a low cost in real contexts of use. However, the 4.1% of *switch* events performed instead of *bi-click*s is more problematic, as a mode switch can lead to a very different view and might be disorienting.

Finally, Figure 7 reports the distribution of the distance between target and actual level reached for *continuous* trials. It shows that participants were very accurate, as this distance is less than 1 mm for about 80% of the trials. This is better than what we had expected, especially considering that this is the result of a single *continuous* event without any additional adjustment to get closer to the target level.

This first empirical evaluation shows that side pressure input is a very promising modality. Users are able to control the whole vocabulary of events with an overall success rate of 90%, most errors being low-cost or costless ones. The next experiment tested how users can take benefit of such a new vocabulary for tasks that mainly involve unidimensional or bi-directional navigation, such as scrolling a document or finding a scene in a video.

SIDEPRESS VS. TOUCH IN A SCROLLING TASK

We compared SidePress with the standard interaction technique for one-handed scrolling: drag and flick touch gestures. The experimental task consists in scrolling a 20-page document (page length = screen height = 920 pixels) to bring a target that was initially out-of-screen inside the viewport. The experiment followed a $2 \times 3 \times 2$ within-participant design with the following factors:

- TECH: the scrolling technique, *Touch* or *SidePress*;
- DIST: the distance between the initial viewport's and target's locations, expressed as a percentage of document length, 5% (1 page), 50% (10 pages), and 80% (16 pages);
- HEIGHT: the target's height, as a percentage of one page's height, 25% (230 pixels) and 75% (690 pixels).

Experimental Task

A trial consists in scrolling through the document up or down to make the target fully visible in the viewport. The target is a black rectangle (600-pixel wide) that lies on a white canvas. A blue knob and a small black rectangle show the respective locations of the viewport and target, in a scrollbar located on the right side of the screen. The scrolling direction and the value of factor DIST define the initial viewport and target locations. When scrolling up (resp. down), the viewport is 1/2 DIST below (resp. above) the center and the target is 1/2 DIST above (resp. below) the center of the viewport.

In the *Touch* condition, participants scroll by performing the default drag and flick gestures provided by the iOS UIScrollView class. In the *SidePress* condition, they can use any of the ten SidePress events, mapped to a scrolling task:

- *light-click*(A) (resp. *light-click*(B)) scrolls up (resp. down) by 12.5% of one page's height (115 pixels);
- *strong-click*(A) (resp. *strong-click*(B)) scrolls up (resp. down) by one page (920 pixels);
- *max*(A) (resp. *max*(B)) jumps to the top (resp. bottom) of the document;
- *continuous*(A) (resp. *continuous*(B)) scrolls up (resp. down) at a speed that depends on the amount of pressure applied. We use a generalized logistic transfer function to ensure a smooth transition between the lowest pressure level that moves the document slowly (7 $pixel.s^{-1}$) and the strongest level that moves it very fast (13000 $pixel.s^{-1}$);
- *bi-click* and *switch-on/off* are enabled, but not mapped to any meaningful action. Both events make the document's background red for $500\,ms$, notifying participants that they made an error.

The audio feedback used in the first experiment is also enabled here, for each SidePress event. The availability of all events from the *SidePress* vocabulary enables participants to choose among different navigation strategies to complete the scrolling task. The operator does not instruct participants to use any specific navigation strategy. Indeed, indicating strategies such as, e.g., *"use five* strong-click *when* DIST=*5"* or *"jump to the top or bottom of the document using* max *to get closer to the target when* DIST=*80"*, might have introduced a bias in favor of *SidePress* as this would likely have reduced the cognitive load associated with choosing a strategy.

Procedure

At the start of the experiment, the operator introduces all SidePress events, while participants freely practice the *tank filling* task (Figure 5) for ten minutes maximum. During this familiarization phase, participants also choose their preferred holding posture, *Fingers* or *Palm* (see Experiment 1), that they will use in all the following phases. The operator then introduces the scrolling task and lets participants practice *SidePress* and *Touch* for a maximum of ten minutes each.

Data collection starts after this training session. Trials are blocked by TECH, half of the participants starting with the *Touch* condition. A TECH block is split into four sub-blocks containing three repetitions of each of the six DIST x HEIGHT conditions in both up and down directions (36 trials) presented in random order. The first sub-block is for training in actual experimental conditions. To summarize, analyses reported below consist of:

$$\begin{aligned} & 2 \text{ TECH conditions} \\ \times\ & 6 \text{ HEIGHT} \times \text{DIST conditions} \\ \times\ & 3 \text{ repetitions} \\ \times\ & 3 \text{ sub-blocks} \\ =\ & 108 \text{ trials per participant.} \end{aligned}$$

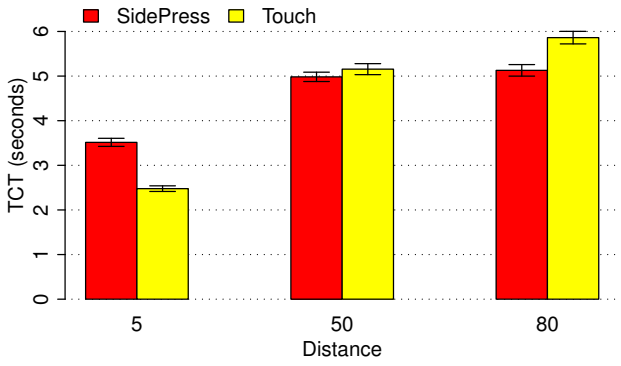


Figure 8. Trial completion time as a function of DIST for each TECH.

At the end of the experiment, participants told the operator which technique they prefer. The experiment lasted one hour.

Participants & Apparatus

Twelve unpaid volunteers (2 females), age 23 to 39 year old (average 29.3, median 28), all right-handed, participated in the experiment using the prototype described earlier. Nine of them had participated in the first experiment. Eight participants were daily users of smartphones, the remaining four only used one from time to time.

Results

The main measure is trial completion time, *TCT*. The timer starts when the program displays the target stimulus and stops when the target is inside the screen, with a stabilized viewport (no inertial movement), and no touch or pressure applied[3]. We removed 5 outliers (two for *Touch* and three for *SidePress*), and studied our collected data using the model: *TCT* $\sim$ TECH $\times$ DIST $\times$ HEIGHT $\times$ Rand(PARTICIPANT).

As expected, an ANOVA reveals a significant effect of DIST ($F_{2,22} = 399, p < 0.0001, \eta^2 = 0.78$) and HEIGHT ($F_{1,11} = 166, p < 0.0001, \eta^2 = 0.22$) on *TCT*. Acquiring a target in a scrolling task takes significantly longer when the target is larger or far away. However, the effect of TECH ($F_{1,11} = 0.07, p = 0.7916, \eta^2 < 0.01$) on *TCT* is not significant. Mean *TCT* for *SidePress* and *Touch* are very close ($4.54\,s$ and $4.50\,s$ respectively).

Significant differences between the two techniques can be observed with interaction TECH $\times$ DIST on *TCT* ($F_{2,22} = 38.2, p < 0.0001, \eta^2 = 0.29$). A post-hoc t-test with Holm correction[4] supports the observations reported in Figure 8: *Touch* is faster than *SidePress* for DIST = 5 ($p < 0.0001$), while *SidePress* is faster than *Touch* for DIST = 80 ($p = 0.004$). For DIST = 50, the difference between the two techniques is not significant. The post-hoc test also shows a significant difference between DIST = 50 and DIST = 80 for *Touch*, but not for *SidePress*.

The only other observed significant interaction comes from TECH $\times$ HEIGHT ($F_{1,11} = 25.6, p = 0.0004, \eta^2 = 0.02$). We tentatively attribute this to a difference in *TCT* between both target heights that is larger for *SidePress* ($4.14\,s$ vs. $4.94\,s$) than for *Touch* ($4.28\,s$ vs. $4.72\,s$). However, the effect size is very small, as indicated by the η^2 value.

Qualitative results indicate that six participants preferred *SidePress* while the other six did not express any preference. Among these latter six, four said that they preferred *Touch* for small distances and *SidePress* for long distances. This is in accordance with the quantitative results reported above. Interestingly, among all twelve participants, nine chose to control SidePress with the fingers vs. three with the thumb and palm.

Discussion

The most interesting observation is the performance difference between the two techniques, depending on target distance: *Touch* is faster than *SidePress* for small distances, and conversely *SidePress* is faster than *Touch* for long distances.

With *Touch*, users can acquire a target that is close to the current position with a simple flick gesture followed by a touch and possibly a small drag gesture to stabilize the viewport on the target. For distant targets, users have to chain several flick gestures, and the optimal way to take advantage of inertial movements induced by these gestures might not be obvious: we observed that some users tended to perform numerous flicks, while others minimized them, performing a new flick only after the viewport had been translated far way via a large inertial movement.

With *SidePress*, users may adopt two main strategies. First, they can rely on continuous control. In this case, there is an incompressible delay of $500\,ms$ before the view actually starts moving (Figure 3). This may be penalizing in comparison with a simple flick for short distances, but this cost can be counterbalanced for long distances if the transfer function between pressure level and scrolling speed is easy to control. This interpretation is supported by the fact that *TCT* is almost the same for DIST = 50 and DIST = 80 in the *SidePress* condition, suggesting that participants were able to apply different pressure levels to optimize scrolling speed depending on movement amplitude. The second strategy consists in jumping to the top or bottom for distant targets and then using continuous control to travel a shorter distance. The data collected revealed that, in the DIST = 80 condition, *max* had been used in approximately 63% of all trials (vs. 6% for DIST = 50 and 0.4% for DIST = 5). The only other discrete event that was used a significant number of times (more than 1% of all trials) is *light-click*. Participants used *light-click* for fine adjustments, especially for large targets, as those are difficult to fit within the viewport (21% of HEIGHT = 75 trials contained *light-click* events vs. only 6% for HEIGHT = 25 trials).

One last interesting observation relates to overshooting. There were more overshoots with *Touch* than with *SidePress*, especially for large distances. For DIST = 5, participants overshot the target (at least once) while scrolling in about 10% of those trials in both *Touch* and *SidePress* conditions. For DIST = 50, the numbers are 10% for *SidePress* vs. 31% for *Touch*; for DIST = 80, results are 18% for *SidePress* vs. 41% for *Touch*. This latter finding corroborates the observation that *SidePress* is more efficient than classic *Touch* when scrolling long distances.

[3] Starting the timer at the first input event leads to the same results.
[4] Bonferroni correction yields the same results.

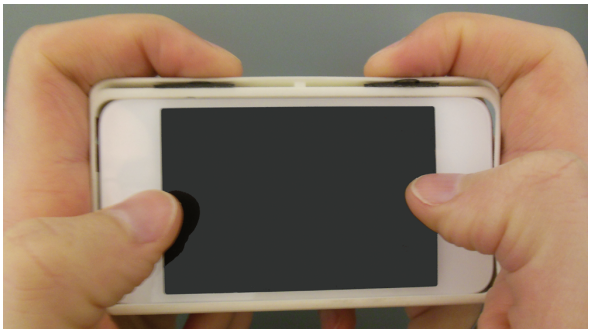


Figure 9. SidePress used in landscape mode.

DISCUSSION AND FUTURE WORK

We investigated the use of pressure sensors on one side of a hand-sized mobile device for occlusion-free unidimensional navigation. We presented a working prototype and a rich vocabulary of events that can be mapped to many different actions in a variety of applications. We reported on two laboratory studies: the first study provides empirical evidence that users can control side pressure sensors to input the right events, with a high level of accuracy; the second study shows that SidePress is more efficient than classic flick-style touch gestures when scrolling large documents.

While SidePress works well for a range of actions such as scrolling, navigating a video, precisely adjusting sliders, or selecting text, it is less relevant for one-handed tasks that require frequent switches between navigation and selection gestures performed on the touchscreen. For example, setting the center of zoom on a map, or selecting a link in a web page may require some users to slightly adjust their hand grip to comfortably touch the screen with their thumb. In such contexts, one hand can be dedicated to navigation with the pressure sensors while the other hand performs selections using touch gestures.

As future work, we plan to study pressure-dependent control in other navigation contexts. For example, zooming could have an effect on performance and subjective preferences because the axis of movement is perpendicular to the axis along which the two sensors lie. We also plan to refine our prototype to reach an even higher level of accuracy by fine-tuning pressure and time thresholds. While we focused on the frequent one-handed usage of mobile devices, we are also interested in studying two-handed usage, as illustrated in Figure 9, where the device is used like a game controller in landscape mode. This allows the user to control the two pressure sensors with her index fingers, leaving both thumbs free to interact with the touch screen.

REFERENCES

1. Alexander, J., Cockburn, A., Fitchett, S., Gutwin, C., and Greenberg, S. Revisiting read wear: analysis, design, and evaluation of a footprints scrollbar. In *Proc. CHI '09*, ACM (2009), 1665–1674.
2. Baudisch, P., and Chu, G. Back-of-device interaction allows creating very small touch devices. In *Proc. CHI '09*, ACM (2009), 1923–1932.
3. Brewster, S. A., and Hughes, M. Pressure-based text entry for mobile devices. In *Proc. MobileHCI '09*, ACM (2009), 9:1–9:4.
4. Butler, A., Izadi, S., and Hodges, S. Sidesight: multi-"touch" interaction around small devices. In *Proc. UIST '08*, ACM (2008), 201–204.
5. Cechanowicz, J., Irani, P., and Subramanian, S. Augmenting the mouse with pressure sensitive input. In *Proc. CHI '07*, ACM (2007), 1385–1394.
6. Clarkson, E. C., Patel, S. N., , Pierce, J. S., and Abowd, G. D. Exploring Continuous Pressure Input for Mobile Phones. Tech. Rep. GIT-GVU-06-20, http://hdl.handle.net/1853/13138, 2006.
7. Gibson, J. *The ecological approach to visual perception*. Houghton & Mifflin, Boston, MA, USA, 1979.
8. Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C., and Want, R. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *Proc. CHI '98*, ACM/Addison-Wesley (1998), 17–24.
9. Heo, S., and Lee, G. Force Gestures: augmenting touch screen gestures with normal and tangential forces. In *Proc. UIST '11*, ACM (2011), 621–626.
10. Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction. In *Proc. UIST '00*, ACM (2000), 91–100.
11. Karlson, A., Benderson, B., and Contreras-Vidal, J. *Understanding one handed use of mobile devices*. In Handbook of Research on UI Design and Eval. for Mobile Tech. IGI Global, 2008, ch. 6, 86–101.
12. McCallum, D. C., Mak, E., Irani, P., and Subramanian, S. PressureText: pressure input for mobile phone text entry. In *Proc. CHI EA '09*, ACM (2009), 4519–4524.
13. Miyaki, T., and Rekimoto, J. GraspZoom: zooming and scrolling control model for single-handed mobile interaction. In *Proc. MobileHCI '09*, ACM (2009), 11:1–11:4.
14. Rahman, M., Gustafson, S., Irani, P., and Subramanian, S. Tilt techniques: investigating the dexterity of wrist-based input. In *Proc. CHI '09*, ACM (2009), 1943–1952.
15. Ramos, G., and Balakrishnan, R. Zliding: fluid zooming and sliding for high precision parameter manipulation. In *Proc. UIST '05*, ACM (2005), 143–152.
16. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure widgets. In *Proc. CHI '04*, ACM (2004), 487–494.
17. Roth, V., and Turner, T. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proc. CHI '09*, ACM (2009), 1523–1526.
18. Shi, K., Irani, P., Gustafson, S., and Subramanian, S. Pressurefish: a method to improve control of discrete pressure-based input. In *Proc. CHI '08*, ACM (2008), 1295–1298.
19. Shi, K., Subramanian, S., and Irani, P. PressureMove: Pressure input with mouse movement. In *Proc. INTERACT '09*, Springer-Verlag (2009), 25–39.
20. Stewart, C., Rohs, M., Kratz, S., and Essl, G. Characteristics of pressure-based input for mobile devices. In *Proc. CHI '10*, ACM (2010), 801–810.
21. Vogel, D., and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. In *Proc. CHI '07*, ACM (2007), 657–666.
22. Wilson, G., Brewster, S. A., Halvey, M., Crossan, A., and Stewart, C. The effects of walking, feedback and control method on pressure-based interaction. In *Proc. MobileHCI '11*, ACM (2011), 147–156.
23. Wilson, G., Hannah, D., Brewster, S., and Halvey, M. Investigating one-handed multi-digit pressure input for mobile devices. In *Proc. CHI EA '12*, ACM (2012), 1727–1732.
24. Wilson, G., Stewart, C., and Brewster, S. A. Pressure-based menu selection for mobile devices. In *Proc. MobileHCI '10*, ACM (2010), 181–190.
25. Wimmer, R., and Boring, S. HandSense: discriminating different ways of grasping and holding a tangible user interface. In *Proc. TEI '09*, ACM (2009), 359–362.
26. Wobbrock, J. O., Myers, B. A., and Aung, H. H. The performance of hand postures in front- and back-of-device interaction for mobile computing. *IJHCS 66*, 12 (2008), 857–875.