Use the Force Picker, Luke: Space-Efficient Value Input on Force-Sensitive Mobile Touchscreens

Christian Corsten[†] Simon Voelker[†] Andreas Link[††] Jan Borchers[†]

RWTH Aachen University
52074 Aachen, Germany
[†]{corsten, voelker, borchers}@cs.rwth-aachen.de, [††]andreas.link@rwth-aachen.de

ABSTRACT

Picking values from long ordered lists, such as when setting a date or time, is a common task on smartphones. However, the system pickers and tables used for this require significant screen space for spinning and dragging, covering other information or pushing it off-screen. The Force Picker reduces this footprint by letting users increase and decrease values over a wide range using force touch for rate-based control. However, changing input direction this way is difficult. We propose three techniques to address this. With our best candidate, Thumb-Roll, the Force Picker lets untrained users achieve similar accuracy as a standard picker, albeit less quickly. Shrinking it to a single table row, 20% of the iOS picker height, slightly affects completion time, but not accuracy. Intriguingly, after 70 minutes of training, users were significantly faster with this minimized Thumb-Roll Picker compared to the standard picker, at the same accuracy and only 6% of the gesture footprint. We close with application examples.

ACM Classification Keywords

H.5.2 User Interfaces: Input devices and strategies (e.g., mouse, touchscreen)

Author Keywords

Force; pressure; bidirectional; rate-based control; value input

INTRODUCTION

Smartphone users frequently need to pick a value from an ordered list, e.g., to set the date and time of an appointment. However, the UI for this task takes up significant screen space: Apple's default system "picker" occupies 3.3 cm, or 30%, of an iPhone 8's screen height. On a Samsung Galaxy S5 running Android, it is 33%. This space is required not just to display the widget *(display footprint),* but also for the swipe gesture to spin the picker's wheels *(gesture footprint)*, as shown in Fig. 1 (left) and the video figure for this paper. Tables are similar: with many items, such as in the iOS *Settings* App, they may occupy the entire screen, and still require scrolling similar to the picker. Slide-in keyboards take up similar space.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-5620-6/18/04. . . $15.00

DOI: https://doi.org/10.1145/3173574.3174235

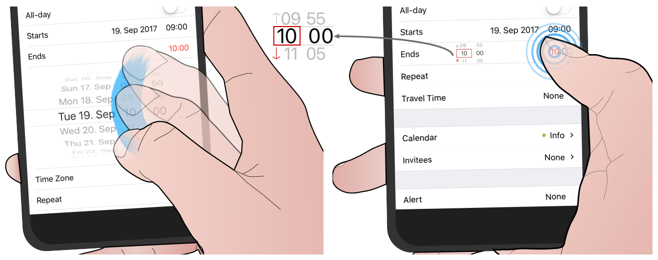


Figure 1. Entering a time on a smartphone. Left: Setting the hour with a picker requires significant screen space in height for dragging and spinning the wheel. Right: Selecting the hour by applying force on the hour digits of the underlying label fades in the *Force Picker.* Rolling the thumb to the left scrolls through the hours—the harder the user presses, the faster. Rolling the thumb to the right scrolls in the opposite direction. Lifting the thumb off the screen sets the value, and the Force Picker disappears. Compared to dragging and spinning, the Force Picker is more compact, so that contextual information is never pushed off-screen.

When these UI elements appear, other content usually disappears: The date picker in the iOS Calendar app, for example, expands from a single row with the date to the equivalent of five rows when activated, pushing content below it down and often off-screen. This can make contextual information needed to pick a value, such as the end time for a meeting, suddenly disappear. Horizontal sliders are another alternative for value input on smartphones that is smaller in height, but they are not recommended for setting precise values [6].

There are alternative interaction techniques that need minimal screen space. Of these, however, speech is socially awkward and time-consuming, tilt sensing makes screens hard to read at an angle and is difficult to use while walking [7, 8, 19], and remapping existing physical controls like volume buttons leads to inconsistent behavior across apps. Bendable interfaces [24] require two hands for good control, and squeezable devices, like HTC's U11, can only sense two different states. All of these require an extra tap to specify where on the screen the input should go. Exploiting simple touch duration instead only allows for changing values at a constant rate, which takes time.

A more powerful option is to use the force-sensitive touchscreen in recent smartphones, like Apple's iPhones since the 6s or Huawei's Mate S, to add force as a third dimension to every touch input. There are two established control mechanisms to map such input to a selected value: *Positional Control (PC)* maps a particular force level directly to a particular value—the harder the user presses, the higher the selected value. Beyond around ten discrete values, however, this technique becomes infeasible [13, 32]. *Rate-Based Control (RC)*, by contrast, maps force to velocity: the harder the user presses, the faster the selected value increases. This allows for efficient selection from larger value ranges, such as dates, times, or percentages.

However, using force for RC has one major drawback: If exerting force increases the currently displayed value at varying speeds, how does the user *decrease* it? One solution are two buttons, one per direction, but this means that the displayed value can no longer serve as the input area itself, since the user needs to pick one of the buttons. This requires adding space for two permanent buttons next to the displayed value. Another alternative is the wrap-around technique: if the user scrolls past the highest value, it wraps around to the lowest value. However, this makes correcting overshoots tedious.

Instead, our *Force Picker* allows for bidirectional RC with force touch to select a value, using a technique to reverse direction that requires no more space than the initial touch.

In summary, this paper makes the following contributions:

- the *Force Picker*, a force-based input technique for value selection that outperforms a standard picker on smartphones with trained users, at a fraction of the size;
- a quantitative study of three techniques for a Force Picker to reverse direction in-place;
- a quantitative study examining the effect of different display sizes for the Force Picker.

RELATED WORK

There is a variety of techniques for entering values on smartphones, such as speech, sensing tilt, or remapping physical buttons, that do not use the touchscreen at all. However, as outlined in the Introduction, they exhibit other drawbacks, and still require the user to tap on the screen to specify which area the input should go to. We instead propose using force input as an alternative, and will focus on this area below.

Related work on force input for selection tasks can be classified into the two groups of control mechanisms introduced above, Positional and Rate-Based Control. They determine how force is used to navigate through the available options.

Positional Control (PC) maps force levels directly to selectable values: applying the same force always selects the same value. In terms of Card et al.'s seminal Design Space of Input Devices [2], this is an "absolute" mapping. While typical force sensors are sampled with 10 bits of resolution for 1024 possible raw sensor values [25], these raw sensor values are usually binned, mapping a range of them to one force level. A transfer function then maps force levels to values. This mapping is usually natural: the higher the force, the higher the value.

The transfer function in PC is often linear, as recommended in [30]. This satisfies the above criteria and is often used for menu selection. However, this approach begins to break down beyond around ten items: Ramos et al. [20] showed that users can control up to six menu items reliably with a force-sensitive pen on a tablet. Mizobuchi et al. [15] found similar results for menu control on a force-sensitive handheld device. Wilson et al. [32] reported that users can control ten levels with their fingers on a handheld device at 85% accuracy with continuous visual feedback. McLachlan et al. [13] added a force sensor to the bezel of a tablet; this enabled the hand holding the device to control a menu of ten items with 89% accuracy. Corsten et al. [4] added force sensors to the back of a smartphone, and found that users could control five levels of force reliably.

Navigating larger ranges with PC requires nonlinear transfer functions. Shi et al. [25] reached 16 controllable levels using a fisheye transfer function with a force-sensitive mouse. Using a similar mouse, Chechanowicz et al. [3] let users first tap on the force sensor multiple times to jump to a coarse level and then press to zoom in on that level at finer granularity, for up to 64 controllable levels. However, for bidirectional control, this technique requires a second force sensor.

In summary, although bidirectional control within the footprint of a single finger is trivial with PC, the limited number of reliably selectable values makes PC inappropriate for setting values of larger ranges, such as times, dates, or percentages.

Rate-Based Control (RC) maps force to the velocity of value change. The transfer function determines how fast values change depending on the force applied. Here, the usual natural mapping means that the harder the user presses, the faster values are browsed. Maintaining a particular force results in change at a constant speed; reducing the force applied slows down browsing. Force RC lets users browse value ranges of any size, but changing direction requires some separate input action.

Shi et al. [26] used a force-sensitive mouse to rotate objects by mapping force to angular velocity. Users tapped the sensor to adjust by single degrees, and rotated counter-clockwise using a second force sensor. The system was faster to use than its PC counterpart. Wilson et al. [31] compared RC to PC for controlling ten menu items on a smartphone using a single force sensor. For RC, they used wrap-around instead of bidirectional navigation. Using RC was faster and more accurate. Using one force sensor per direction, Ng et al. [17] report similar results for menu control in a car. Spelmezan et al. [28] attached force and proximity sensors to the side of a smartphone for bidirectional continuous thumb scrolling. Their revised technique [29] used two force sensors for bidirectional scrolling and zooming by using either the index and ring finger or thumb and palm. While touch input was faster for small scrolling distances, RC prevailed for longer distances. Holman et al. [9] attached two force sensors to the side of a smartphone to use with the fingers holding the phone while thumbing, and found that gestures augmenting the natural thumb touch worked best. Pelurson et al. [18] attached a force sensor to a smartphone. Using RC to scroll large information spaces horizontally while changing direction by swiping on the touchscreen was faster and preferred over the default drag & flick interaction. Antoine et al. [1] used RC on a force-sensitive iPhone to scroll a viewport vertically down while simultaneously dragging with the thumb. Users were faster and had fewer errors compared to baseline edge-scrolling.

PreSense II [22] and GraspZoom [14] describe bidirectional RC using force touch. Similar to our approach, they support immediate change of direction with a minimal gesture footprint. In PreSense II, the user tilts her index finger up vertically to navigate in the opposite direction while force-touching on a desktop touchpad. In GraspZoom, swiping across the screen selects the direction of scrolling and zooming with RC, while a sensor on the back of the smartphone captures continuous force input. Both techniques, however, were not evaluated. Hence, it is not clear how fast and how accurate users perform with such techniques, especially for discrete value input, and whether such techniques let us minimize both gesture footprint and widget display footprint on the mobile touchscreen.

BIDIRECTIONAL FORCE INPUT TECHNIQUES

Below, we present three interaction techniques we designed for bidirectional value selection using force input that minimize the gesture footprint. Since we want to change values across a large range, e.g., 1–31 to set a date, all our techniques require RC, as our review of related work has illustrated.

To let standard touch input and RC using force touch coexist on the same widget without accidental activation, we classify any force input below a *resting threshold* of 20% (based on Apple's guidelines) of the force sensor range as an ordinary touch that does not start changing the displayed value. Between 20% and 80% of the sensor range, our force-to-value mapping iterates over values at a velocity depending on a linear transfer function, as used in [31]: The more force is applied, the faster the selected value changes. To efficiently cross large value distances, crossing 80% of the sensor range activates a *boosted* transfer function. We focus on one-handed interaction in portrait mode, using the thumb of the dominant hand. Lifting the thumb off the screen was chosen to confirm the currently selected item, removing the need for an extra tap or an additional confirmation button.

By default, RC only supports changing values in one direction. Our designs (Fig. 2) address this shortcoming, each with its own benefits, within the footprint of a single force touch:

In **Pulse**, the user changes direction by quickly increasing force significantly above and back below the resting threshold. This technique has the advantage of being identical for both directions. To correct for overshoots in either direction, the user reduces force below the resting threshold, issues a Pulse, and increases force again to scroll towards the desired value.

Press-Through maps force increase to value increase, and force decrease to value decrease: To reverse direction, the user quickly applies the maximum detectable force, starting from below the resting threshold. While she maintains that maximum force, value change pauses, like in the resting force zone. When she reduces the force, the value starts decreasing, reversing the transfer function—the more she reduces the force, the faster the value decreases. Re-applying more force decelerates value decrease, up to the maximum detectable force, which pauses value changes. Within 40% to 20% of the force sensor spectrum, the boosted transfer function is applied. Below 20%, resting force is reached, which pauses value change and reverts back to the normal transfer function. To correct overshoots,

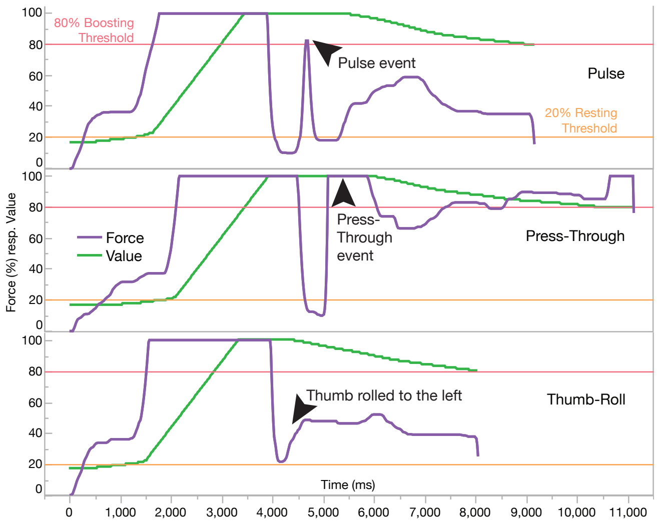


Figure 2. Force input and corresponding selected values for our three force techniques over time, when changing a value from 17 to 80 while overshooting to 100 in between. Note how values begin to decrease when the corresponding change of direction (black arrows) is detected.

the user quickly reduces force beneath the resting threshold, and then issues the Press-Through gesture to reverse direction. The advantage of Press-Through is its natural mapping of force increase/decrease to value increase/decrease.

In **Thumb-Roll**, the user gently rolls her thumb to the left to decrease, and to the right to increase values. The applied force sets the speed in either direction. To correct for overshoots, she rolls her thumb to the other side. This results in a natural decrease and therefore decelerated navigation while the thumb is rolling through its neutral, flat position. Since the thumb has to be rolled and maintained in that position while navigating, Thumb-Roll represents a temporary "quasi-mode" [21] that ensures the user is always aware of the active direction from her thumb position. Like in Pulse, the transfer function is identical for both directions. Thumb-Roll is similar to a rocker switch, or two small force buttons right next to each other. Although rolling the thumb up and down would result in a more natural mapping for increasing and decreasing values, left-right roll is ergonomically easier to perform and leaves a much smaller footprint compared to rolling the thumb up and down. Roudaut et al. [23] investigated thumb-roll gestures as input modifiers for traditional mobile touchscreens. Their technique MicroRolls gives users faster access to toolbars and menus on PDAs through four straight and two circular thumb roll gestures. However, that work did not explore force input.

STUDY 1: FORCE INPUT VS. DRAGGING & SPINNING

Having defined these candidate techniques to reverse direction for the Force Picker, we wanted to understand how they each compare to a standard picker that is controlled by dragging and spinning, in terms of speed, accuracy, and gesture footprint. 16 participants (21–31y, $M = 26.19$, $SD = 2.71$, all right handed, five females) used a Force Picker with each candidate technique, and a standard picker as baseline, on a force-sensitive iPhone 6s Plus. Fig. 3 shows our application to display instructions and capture data.

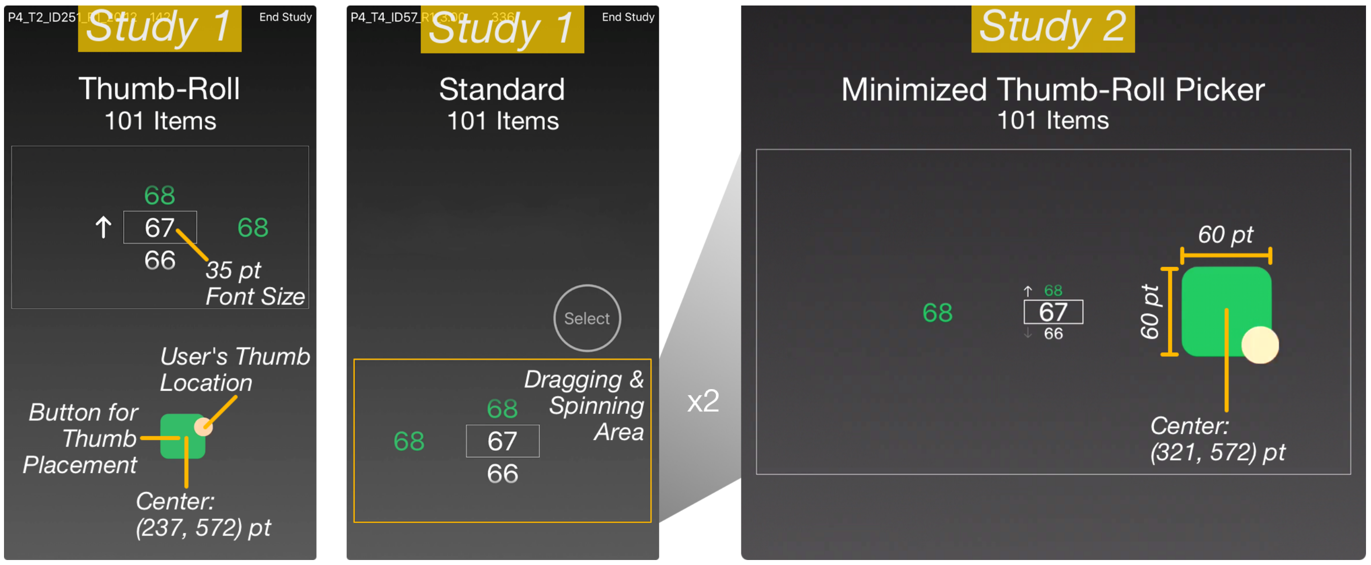


Figure 3. UI of our study prototype showing the picker and instructions. Left: Force Picker and System Picker from study 1. Right: Minimized Thumb-Roll Force Picker, 2× magnified. The iPhone 6s Plus screen measured 414×736 pt, or 68×122 mm; the origin is located in the top left corner.

Picker Design

For our baseline condition, we designed a **standard picker** similar to an iOS or Android system picker. We reserved the same 414×216 pt (65×36 mm, 30% screen height) space for the dragging and spinning footprint that Apple recommends to use for the iOS 10 system picker [10]. Note that the unit 'pt' used throughout this paper does not refer to the unit measure used in typography, but relates to Apple's measurement for display points on iPhone 6/7/8 Plus (1pt ≈ .16 mm). We displayed a rectangle with slightly narrower sides around this area, such that users could easily perceive the picker boundaries. Although the iOS system picker displays seven values at a time, of which the two upper and lower values are in a vertically compressed font (Fig. 4, left), we chose to display only three values at a time, like the Android system picker does (Fig. 4, right). This was done since we anticipated shrinking the display space for our Force Picker in study 2 and 3 (Fig. 3, right), which would be difficult for displaying seven values at a time, and we wanted to be able to do a fair comparison with our results across all studies.

Dragging the current value down with the thumb anywhere on the picker area increased the value displayed and vice versa. We used the standard iOS 10 accelerations for dragging and spinning from the `UIScrollView` class. The target value was shown to the left so that it would not be covered by the thumb while dragging. The standard picker was placed at the bottom of the screen to make sure that it was easily accessible by the thumb (Fig. 3, center). Since liftoff happens frequently on standard pickers, it cannot be used for confirmation; users had to tap a 'select' button placed within thumb reach above the picker to confirm the selection. Despite continuous scrolling, selection would always snap to the closest discrete value.

For our **Force Picker** (Fig. 3, left), we used the same visualization as in the standard picker, but instead of controlling it by dragging and spinning, users were asked to use our force techniques to navigate to the target value shown next to the picker. Users held the device in their right (dominant) hand, and placed their thumb on a green 60×60 pt button. When they applied force, the values started scrolling up resp. down. The arrow indicated the current direction. Upon releasing the thumb, the picker snapped to the closest value and selected it, completing each trial.

For both our standard picker and our Force Picker, we displayed the higher value above, and the lower value below the current item. Although the iOS and Android pickers display values in reversed order, we wanted to achieve a more natural mapping for force input: An increase (↑) in force increased (↑) the value more quickly by scrolling more quickly.

Transfer Function and Implementation of Force Techniques

According to Apple's API documentation [11], the iPhone force sensor API delivers unitless force values between 0 and $\frac{400}{60} \approx 6.67$ in steps of $\frac{1}{60}$, with force sensitivity set to the "medium" default. Values around 1.0 should be interpreted as an ordinary touch; higher values as intentional force input. Although Apple does not state how these values translate to Newtons, experiments [16] suggest a 4 N range and a linear transfer function. In the remainder of the paper we report both the iOS-specific 0–6.67 range and their generic relative values.

According to the design of our techniques, we used RC for mapping force to the Force Picker scrolling speed. Forces below 1.35, the 20% resting threshold, were ignored to let ordinary touch input coexist and to let users rest their thumb on the touchscreen without affecting value input. We set $Speed(x) = (24.812x - 33.496)\frac{mm}{s}$, $x \in [1.35; 5.34)$, following [31]. For forces beyond 5.34, the 80% boost threshold, we set $Speed(x) = (38.824x + 46.588)\frac{mm}{s}$, $x \in [5.34; 6.67)$, which we determined through pilot tests. Pressing harder than 6.67 resulted in a plateau of 305 $\frac{mm}{s}$ scrolling speed.

We implemented our force techniques according to our specifications above. To issue a Pulse, users had to start below the resting threshold of 20% of the sensor range, and cross it

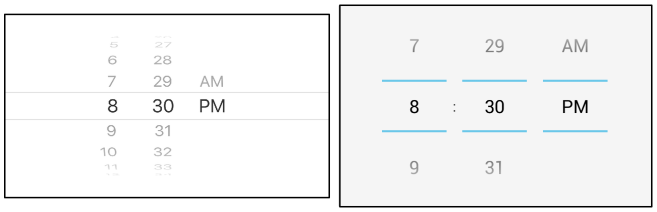
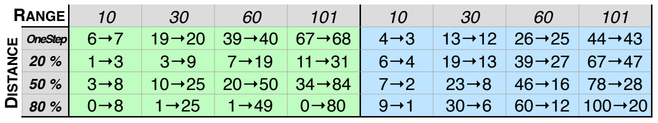


Figure 4. Default iOS 10 system picker (left) and Android 7 system picker (right). The iOS picker measures 414×216 pt on a 414×736 screen (taking up 30% of the screen height) and shows up to seven values at a time, whereas the Android picker measures 620×424 px on a 720×1280 px screen (taking up 33% of the screen height) and displays three values per picker wheel at a time.



RANGE	10	30	60	101	10	30	60	101
DISTANCE *OneStep*	6→7	19→20	39→40	67→68	4→3	13→12	26→25	44→43
20 %	1→3	3→9	7→19	11→31	6→4	19→13	39→27	67→47
50 %	3→8	10→25	20→50	34→84	7→2	23→8	46→16	78→28
80 %	0→8	1→25	1→49	0→80	9→1	30→6	60→12	100→20

Figure 5. Start and target values per RANGE×DISTANCE combination.

clearly, reaching 3.34 (50% of the sensor range, determined in pilot tests) at a rate of change of at least 10% between two digitizer frames captured every 16 ms, and then drop quickly below the resting threshold again. Detection for Press-Through was similar, except that users had to reach the maximum measurable force of 6.67 without reducing force quickly afterwards. For Thumb-Roll, we captured the location of the thumb's touch point at resting force. When the touch moved to the right or left, direction was set to up or down, respectively. Since the user's thumb could drift during rolling, we re-calibrated its location whenever force dropped below the resting threshold.

Independent Variables were TECHNIQUE (Baseline, Pulse, Press-Through, and Thumb-Roll), RANGE (10, 30, 60, and 101 items), and DISTANCE (OneStep, 20%, 50%, 80%) in both directions. RANGE determined the number of available picker values, representing typical use cases: selecting a digit (0–9), a day of the month (1–30), minutes for a timer (1–60), or a percentage (0–100). DISTANCE denoted how many values lay between start and target value. We chose relative DISTANCEs to allow comparison across RANGEs. Only OneStep was absolute to test single increments and decrements, representing typical off-by-one corrections. For all force techniques, applying force navigated up by default. Fig. 5 lists all values users had to navigate from and to in both directions. To prevent shortcut strategies by navigating in the opposite direction, we disabled wrap-around. Instead, DISTANCE included OneStep and 20% as short distances. We did not include the extreme values as targets since this would have simplified the task, as scrolling stopped at these values, and since it would not be feasible for pickers that support wrap-around.

We recorded 4 TECHNIQUEs $\times$ 4 RANGEs $\times$ 4 DISTANCEs $\times$ 2 directions $\times$ 3 repetitions $=$ 384 trials per participant. We also screen-captured all trials to investigate potential outliers later. TECHNIQUE and RANGE were each counterbalanced using a Latin Square, and DISTANCE in both directions was randomized. Once all three repetitions were done, the user continued with the next RANGE, until all RANGEs had been tested. As each TECHNIQUE was presented to the user, she was allowed to explore it until she felt more familiar with it.

Dependent Variables were *Time* [ms], *Crossings* [$i \in \mathbb{N}$], and *Success* [0,1]. For each trial, *Time* denoted the time from first contact with the touchscreen until value selection by liftoff (force techniques) or tapping the 'select' button (Baseline). *Crossings* counted how often the user overshot the target value, as in, e.g, [20], such that she had to change direction and navigate back. *Success* indicated whether the user selected the correct target value (1) or not (0). While we also recorded how far off users were from the target value in those cases, they missed by more than one in only .05% of all trials. We also logged touch positions every 16 ms to capture gesture starting points and *footprints*. At the end, users were asked to *rank* the four techniques by preference (1: most, 4: least).

Results

A total of 32 outliers were identified by applying the Tukey Method for Extreme Outliers on *Time* (18–63 s). Looking at the screen recordings for these trials revealed that users had their thumb already placed on the touchscreen, which activated time counting, but they were still asking questions before actually performing the task. Hence, these trials were not representative and we therefore excluded them from the analysis. Since we were interested in how performance for each TECHNIQUE was affected by RANGE and DISTANCE, we concentrate on these results. Hence, although significant for *Time* and *Crossings* (p <.001, each), we will not discuss main effects for RANGE and DISTANCE, since these mix data from all TECHNIQUEs. We log-transformed *Time* for repeated measures ANOVA. Since *Success* was dichotomous, we conducted Cochran's Q and McNemar tests. We analyzed the count for *Crossings* with Friedman and Wilcoxon Signed Rank tests.

TECHNIQUE had a significant main effect on ***Time*** ($F_{3,6065} = 955.73$, p <.001). Tukey HSD post hoc pairwise comparisons were all significant (p <.001, each). At 2,547 ms, users were fastest for Baseline. The fastest force technique was Thumb-Roll, yet 900 ms slower than Baseline, followed by Pulse and Press-Through (Fig. 6, right).

There was also a significant TECHNIQUE×DISTANCE interaction effect ($F_{9,6065} = 10.90$, p <.001). Tukey HSD post hoc pairwise comparisons revealed that, for each TECHNIQUE, users were significantly fastest for OneStep, followed by 20%, 50%, and 80% DISTANCE (p <.001, each). This was to be expected, as navigating farther distances takes more time. For each DISTANCE, pairwise post hoc tests showed that users were always fastest for Baseline (Fig. 6, left), then Thumb-Roll, Pulse, and Press-Through (Pulse vs. Thumb-Roll with 50% and 80% DISTANCE: both $p = .01$, all others: p <.001).

TECHNIQUE had a significant main effect on ***Crossings*** ($\chi^2(3) = 121.41$, p <.001). Post hoc pairwise comparisons revealed that users made significantly more *Crossings* for Baseline compared to all force techniques (p <.001, each), but the difference was small (Fig. 7, left).

There was also a significant TECHNIQUE×RANGE interaction effect ($\chi^2(15) = 154.27$, p <.001). Post hoc tests revealed that users made significantly fewer *Crossings* with Pulse for RANGE 10 compared to Baseline ($p = .001$), and they made significantly fewer *Crossings* with Thumb-Roll for RANGE 30

		RANGE				DISTANCE			
		10	*30*	*60*	*101*	*OneStep*	*20%*	*50%*	*80%*
Baseline	**M**	1,435	2,148	3,004	3,606	855	2,151	3,180	4,009
	SD	674	1,204	1,857	2,279	604	961	1,561	1,996
Pulse	**M**	2,685	3,764	4,691	5,561	1,863	3,899	4,967	5,964
	SD	2,148	2,530	2,950	3,599	1,627	2,714	2,870	3,115
Press-Through	**M**	4,542	5,699	3,764	8,705	3,967	5,703	7,809	9,163
	SD	4,635	4,391	5,863	6,213	4,665	4,812	5,414	5,876
Thumb-Roll	**M**	1,983	2,963	4,021	4,863	1,320	2,988	4,343	5,187
	SD	1,185	1,815	2,496	3,101	1,018	1,694	2,514	2,539

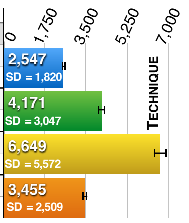


Figure 6. ***Time*** **(ms) for study 1. Left: M and SD for TECHNIQUE split by RANGE and DISTANCE. Right: Mean** ***Time*** **for each TECHNIQUE, all significantly different from each other. Error bars denote 95% CI.**

compared to Baseline ($p = .006$) (Fig. 8, left). Within each TECHNIQUE, RANGE did not affect *Crossings*.

There was also a significant TECHNIQUE×DISTANCE interaction effect ($\chi^2(15) = 485.17, p < .001$). Post hoc pairwise comparisons revealed that users made significantly fewer *Crossings* for all force techniques compared to Baseline for all percentual DISTANCEs (Baseline vs. Thumb-Roll with 50% and 80% DISTANCE: $p = .004$, Baseline vs. Pulse with 80% DISTANCE: $p = .002$, all others: $p < .001$). Only for OneStep, users made significantly more *Crossings* with Press-Through compared to all other techniques ($p < .001$, each) (Fig. 8, right). Also, these tests revealed that for Baseline, Thumb-Roll, and Pulse, users always made significantly fewer *Crossings* for OneStep compared to all other DISTANCEs (Baseline vs. Thumb-Roll with 20% DISTANCE: $p = .012$, Baseline vs. Pulse with 50% and 80% DISTANCE: both $p = .003$, all others: $p < .001$).

TECHNIQUE had a significant main effect on ***Success*** ($Q(3) = 61.69, p < .001$). Post hoc pairwise comparisons were not significant between Pulse and Thumb-Roll, but between all other pairs ($p < .001$, each): Users performed with 100% *Success* for Baseline, followed by Pulse (98.7%), Thumb-Roll (98.6%), and Press-Through (96.6%), (Fig. 7, right). Although Pulse and Thumb-Roll had significantly lower *Success* compared to Baseline, the difference of less than 1.4% is small.

There was also a significant TECHNIQUE×RANGE interaction effect ($Q(15) = 78.40$, $p < .001$). Post hoc pairwise comparisons revealed that users had always significantly higher *Success* with Baseline (100%) compared to Press-Through for each RANGE (10: 97%, $p = .02$; 30: 95%, $p < .001$; 60: 96%, $p < .001$; 101: 98%, $p = .021$). In addition, for RANGE 30, users had significant lower *Success* for Press-Through (95%) compared to Baseline (100%), Pulse (99%), and Thumb-Roll (98%), ($p < .001$, each). Also, for RANGE 60, users had significantly higher *Success* for Thumb-Roll (99%) compared to Press-Through (96%), ($p = .035$). Within each TECHNIQUE, however, RANGE had no effect on *Success*.

There was also a significant TECHNIQUE×DISTANCE interaction effect ($\chi^2(15) = 69.16, p < .001$). Post hoc pairwise comparisons revealed that users had always significantly higher *Success* for Baseline (100%) compared to Press-Through for each DISTANCE (OneStep: 97%, $p = .004$; 20% DISTANCE: 96%, $p < .001$; 50% DISTANCE: 96%, $p < .001$; 80% DISTANCE: 97%, $p = .007$). In addition, for 20% DISTANCE, users had significantly lower *Success* for Press-Through (96%) compared to all techniques (Baseline: 100%, $p < .001$; Pulse:

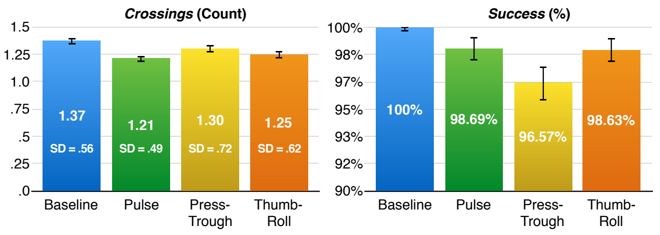


Figure 7. ***Crossings*** **(left) and** ***Success*** **(right) for study 1. Users made significantly more** ***Crossings*** **for Baseline, but its** ***Success*** **was significantly higher than for all force techniques. Error bars denote 95% CI.**

	RANGE (M, SD)								DISTANCE (M, SD)							
	10		*30*		*60*		*101*		*OneStep*		*20%*		*50%*		*80%*	
Baseline	1.35	0.55	1.34	0.52	1.36	0.57	1.42	0.60	1.02	0.14	1.54	0.59	1.47	0.60	1.45	0.61
Pulse	1.16	0.44	1.19	0.46	1.22	0.51	1.27	0.54	1.06	0.27	1.28	0.60	1.24	0.47	1.26	0.52
Press-Through	1.34	0.82	1.27	0.70	1.29	0.68	1.30	0.68	1.30	0.72	1.27	0.63	1.32	0.78	1.31	0.75
Thumb-Roll	1.19	0.59	1.21	0.57	1.27	0.62	1.32	0.70	1.07	0.37	1.25	0.58	1.32	0.72	1.34	0.73

Figure 8. ***Crossings*** **for study 1 split by RANGE and DISTANCE.**

99%, $p = .035$; Thumb-Roll: 99%, $p = .005$) Also, for 50% DISTANCE, users had significantly lower *Success* for Press-Through (96%) compared to Pulse (99%), ($p = .003$). Still, within each TECHNIQUE, DISTANCE had no effect on *Success*.

Users' ***Ranking*** ($\chi^2(3) = 29.25, p < .001$) revealed no significant differences between Baseline ($M = 1.56$, $SD = .90$), Thumb-Roll ($M = 1.88$, $SD = .89$), and Pulse ($M = 2.06$, $SD = .68$) ($p < .001$, each). Press-Through ($M = 3.63$, $SD = .50$), however, was significantly least preferred over Baseline ($p < .001$), Pulse ($p = .003$), and Thumb-Roll ($p < .001$).

Fig. 9 visualizes the ***Gesture Footprint*** from all users for each TECHNIQUE. For our force techniques, the green square marks where users were asked to place their thumb. Red data points represent first contact with the screen (*Touch Began*), blue data points all other touches (*Touch Moved*). As can be seen, the footprint for the force techniques is much smaller than for Baseline. Based on all first touches, we also calculated the effective width×height (yellow overlays in Fig. 9) capturing 96% of these data points, multiplying the SD in x and y by 4.133 [27]. These rectangles denote the space that needs to be reserved for detecting the start of an interaction with the Force Picker. During interaction, users may drift (blue dots), and touch input outside the rectangles should not be passed on to other widgets until thumb liftoff ends this modal interaction. Note that content beneath the blue dots will be occluded by the thumb. With the standard picker, users do multiple liftoffs while spinning and dragging, whereas with our force techniques, each selection requires only one liftoff. Thumb-Roll required only 12%, Pulse and Press-Through only 5% of the effective area (width×height) required for Baseline.

Discussion

For our force techniques, users performed fastest (3.5 s) and most accurate (99%) with Thumb-Roll. Although users' *Success* for Pulse was the same, they were a significant 700 ms slower. For Baseline, users were 900 ms faster than for Thumb-Roll, and achieved 100% *Success*. However, this was to be anticipated, since all users were familiar with standard picker interaction, but not with force control, and post-corrections after liftoff are part of the interaction with standard pickers.

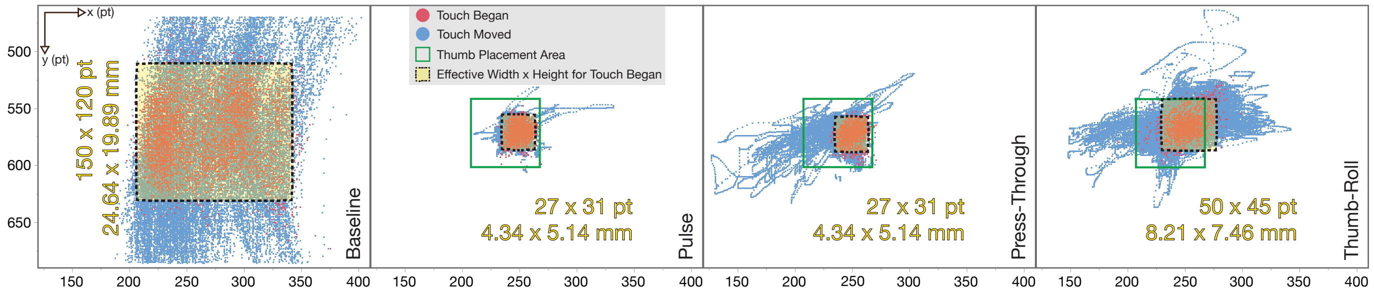


Figure 9. Gesture footprint from study 1. Our three Force Picker techniques (right) have much smaller footprints than standard dragging and spinning (left). Yellow overlays denote 96% of all initial touch points. Compared to Baseline, the force techniques require only 5–12% of the space.

Users' slower task completion time for our force techniques could also be explained by *Crossings*. Users reported that overshoots felt easier to correct with dragging and spinning than by applying force, and therefore navigated more carefully with the force techniques. For single value increase or decrease, however, users could gently tap using our force techniques, as this would cause the picker to scroll a little but then snap to the next value. This could explain why users made significantly fewer *Crossings* compared to further DISTANCES. For Press-Through, however, tapping for a one-step change was not possible when decreasing values. This technique was also problematic for users when decreasing values because of sensor limitations: If they had applied force beyond what the sensor could detect, as they were beginning to release, the system could not provide continuous feedback, although this is essential for force input [31]. Users then tried to reduce force more quickly, which resulted in overshooting, hence an increase in *Crossings* and *Time* needed for corrections. Furthermore, users were rather confused that the force-to-value mappings reversed depending on the direction, and did not consider the natural mapping between force increase/decrease and value increase/decrease a benefit. Consequently, users ranked Press-Though lowest among all techniques.

As expected, the gesture footprint for all force techniques was drastically smaller than for dragging and spinning. Naturally, Thumb-Roll had the largest footprint among force techniques, since rolling requires more space than stationary Pulse or Press-Through gestures. Nevertheless, Thumb-Roll accounted for only 12% of the effective width×height by Baseline. In all, since Thumb-Roll achieved the best *Time*, *Success*, and user *Ranking* of our force techniques at a fairly small gesture footprint, we further pursue this technique in the rest of the paper. If the footprint needs to be even smaller, Pulse is an alternative to consider, but at the cost of *Time*.

The promising reduction in *gesture* footprint from Thumb-Roll led us to investigate whether we could now effectively save screen space by also reducing the *display* footprint of the Force Picker. Therefore, we next compared user performance for Thumb-Roll in our standard-sized Force Picker to using it in a minimized Force Picker.

STUDY 2: MINIMIZING THE FORCE PICKER

We modified study 1, using only Thumb-Roll as technique and adding SIZE as independent variable to represent the 414×216 standard-sized picker and a 44 pt squared minimized Force Picker (7.3×7.3 mm) that fits within the height of an iOS table row (Fig. 3). We adjusted the transfer function for the minimized Force Picker to $Speed(x) = 8.271x - 11.165\frac{mm}{s}, x \in [1.35; 5.34)$ (normal) and $Speed(x) = 12.941x + 15.529\frac{mm}{s}, x \in [5.34; 6.67)$ (boosted) to achieve the same scrolling for both pickers. Based on the gesture footprint from study 1, the thumb placement area was shifted a little more to the right (Fig. 3, right), such that the thumb would not occlude the picker during interaction. Levels for RANGE and DISTANCE were identical to study 1. Again, *Time*, *Crossings*, *Success*, and *Gesture Footprint* were recorded. We also asked users whether they found the minimized Force Picker hard to read (7-point Likert scale, 7 = totally agree).

Of our eight participants (24–40y, $M = 31.13$, $SD = 6.51$, all right-handed, three females, none from study 1), four started with the minimized Force Picker. Counterbalancing, randomization, and presentation order of conditions were identical to study 1. We recorded 2 SIZES × 4 RANGES × 4 DISTANCES × 2 directions × 3 repetitions = 192 trials per user, excluding two test trials for each SIZE. Users also had the opportunity to briefly familiarize themselves with both pickers beforehand.

Results

Data analysis was performed similar to study 1, directly contrasting effects for each SIZE. We again log-transformed *Time* for a repeated-measures ANOVA.

SIZE had a significant main effect on ***Time*** ($F_{1,1514} = 9.86$, $p = .002$): Users needed $M = 2{,}698$ ms to complete a trial for the standard-sized Force Picker, and $M = 2{,}988$ ms for the minimized version (Fig. 10, left). There were no interaction effects on *Time*.

There was a significant TECHNIQUE×RANGE interaction effect on ***Crossings*** ($\chi^2(7) = 30.21$, $p < .001$), but post hoc tests were not significant. There was also a significant TECHNIQUE×DISTANCE interaction effect ($\chi^2(7) = 61.92$, $p < .001$). Post hoc tests showed that users made significant fewer *Crossings* for OneStep ($M = 1.06$, $SD = .67$) compared to 50% DISTANCE ($M = 1.34$, $SD = .27$) for the standard-sized, and for the minimized Force Picker, users made significant fewer *Crossings* for OneStep ($M = 1.08$, $SD = .35$) compared to 80% DISTANCE ($M = 1.50$, $SD = .97$) (both $p < .001$).

There were neither significant main effects nor interaction effects for ***Success***. Users correctly selected values with a *Success* of 96.6% for the standard-sized and 96.1% for the minimized Force Picker (Fig. 10, right).

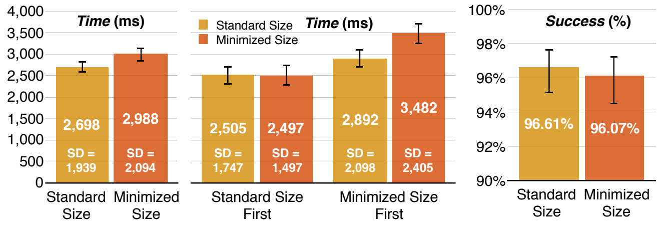


Figure 10. ***Time*** **and** ***Success*** **for study 2. Users were 290 ms faster with the standard-sized Force Picker (left), but starting with the standard size also sped up using the minimized version afterwards (center).** ***Success*** **was the same for both SIZEs (right). Error bars denote 95% CI.**

The effective width×height based on the ***Gesture Footprint*** from first touch contact measured 43×53 pt for the standard-sized Force Picker (Fig. 11, left), which was slightly larger than the 43×42 pt for the minimized version (Fig. 11, right). These results are similar to study 1.

Overall, users disagreed that the minimized Force Picker was hard to read ($M = 3.38$, $SD = 1.51$); only one user explicitly stated reading difficulties.

Discussion

Users were 290 ms slower with the minimized Force Picker than with the standard-sized Force Picker. Nevertheless, for both SIZEs, users from study 2 were 470–750 ms faster compared to Thumb-Roll performance from study 1.

We wondered whether this was due to a training effect, since users had to perform twice as many Thumb-Roll trials in study 2. Therefore, we split the *Time* data: Users who tested the standard-sized Force Picker first performed trials equally fast for both SIZEs (Fig. 10, center): 2,505 ms for the standard-sized Force Picker and 2,497 ms for the minimized Force Picker—which was actually as fast as dragging and spinning in study 1. However, at 3,482 ms, users testing the minimized Force Picker first were about 1 s slower than the other group, but improved their performance while testing the standard-sized Force Picker afterwards ($M = 2{,}892$ ms). It seems that users testing the minimized Force Picker first navigated more carefully because they were confronted with two new situations at once, Thumb-Roll force input and the small picker visualization, while those testing the standard-sized Force Picker first encountered them one by one, increasing confidence.

Crossings and *Success* were similar for both SIZEs, although *Success* was slightly worse (96%) than with Thumb-Roll in study 1, possibly due to a time–accuracy trade-off.

Based on these findings, we wondered whether users could become faster with further training, since users from both studies so far had known neither Thumb-Roll force input nor the minimized Force Picker visualization before. Therefore, we conducted a third study with trained users comparing the minimized Thumb-Roll Force Picker against a standard picker.

STUDY 3: TRAINED USER PERFORMANCE

We recruited four participants (29–35y, $M = 32.50$, $SD = 2.65$, all right-handed, one female, none from previous studies) to complete ten training sessions at home over five days to master Thumb-Roll for the minimized Force Picker. We modified study 2 to capture DISTANCEs from 10–90% (in 10% steps) as well as OneStep, again for both directions. We picked the same RANGE sizes. Users performed 4 RANGEs × 10 DISTANCEs × 2 directions× 10 sessions = 800 trials. On average, the training took 1:10 h. Afterwards, users did a final session at our lab, using the tasks from studies 1 and 2, but testing the Baseline standard picker from study 1 against the minimized Thumb-Roll Force Picker from study 2.

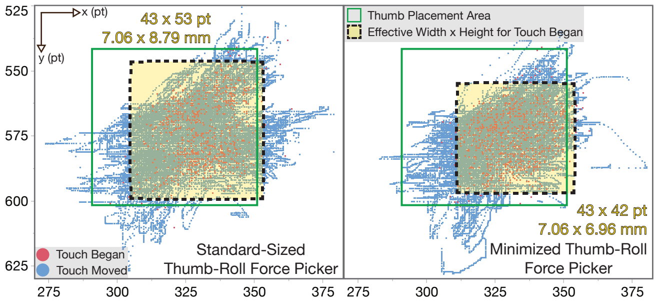


Figure 11. Gesture Footprint for study 2. Left: Footprint for the standard-sized Thumb-Roll Force Picker. Right: Footprint for minimized Thumb-Roll Force Picker. The yellow overlays denote 96% of all first touch contact points and were similar for both pickers.

Results

Fig. 12 shows how users' *Time* decreased over the ten training sessions, while *Success* remained at about 98%. For the analysis, we will focus on the results from the final test, again by directly contrasting effects for each TECHNIQUE. *Time* was again log-transformed for repeated-measures ANOVA.

TECHNIQUE had a significant main effect on ***Time*** ($F_{1,751} = 143.20$, $p < .001$): Users needed $M = 2{,}300$ ms to complete a trial for Baseline, but only $M = 1{,}796$ ms for Thumb-Roll (Fig. 13, left).

TECHNIQUE also had a significant main effect on ***Crossings*** ($Z = 3062.50$, $p < .001$). As seen in study 1, users made significantly fewer *Crossings* using Thumb-Roll ($M = 1.16$) than with Baseline ($M = 1.35$), but the difference was small (Fig. 13, center). There was also a significant TECHNIQUE×RANGE interaction effect ($\chi^2(7) = 35.27$, $p < .001$), but post hoc tests were not significant. There was also a significant TECHNIQUE×DISTANCE interaction effect ($\chi^2(7) = 107.98$, $p < .001$). Post hoc tests revealed that within TECHNIQUE, users made significantly fewer *Crossings* for OneStep (each TECHNIQUE: $M = 1.01$, $SD = 1.02$) compared to all other DISTANCEs (Baseline: $M = 1.44$–1.50, $SD = .54$–$.63$, Thumb-Roll: $M = 1.19$–1.24, $SD = .40$–$.43$), ($p < .001$, each).

There were neither significant main effects nor interaction effects for ***Success***: Although Baseline had again 100% *Success*, this was not significantly different from the 99% for Thumb-Roll (four errors were made in total).

Fig. 14 shows ***Gesture Footprint*** and effective width×height based on first touch contacts for both TECHNIQUEs. The minimized Thumb-Roll Force Picker takes up only 40×25 pt, 6% of the 131×134 pt dragging and spinning requires. While the low number of users leads to lower variance, Fig. 14 shows a very consistent gesture footprint among these trained users.

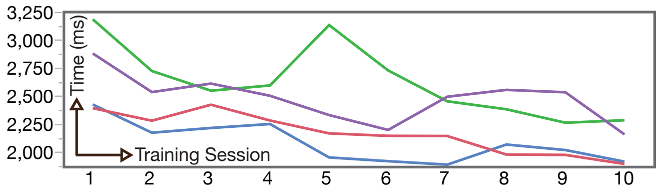


Figure 12. ***Time*** **(ms) for each training session from four users before performing a final test in study 3. Users became faster over ten sessions.**

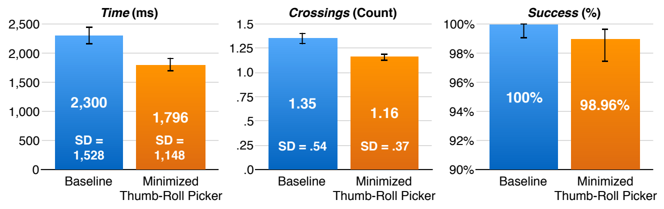


Figure 13. ***Time*** **(ms),** ***Crossings*****, and** ***Success*** **(%) for study 3. Overall, users were faster and made fewer** ***Crossings*** **with Thumb-Roll compared to Baseline, while** ***Success*** **was not significantly different between both TECHNIQUES. Error bars denote 95% CI.**

Discussion

The results indeed show that, with some training, users can become faster with the minimized Thumb-Roll Force Picker than with the standard picker. Task completion time for dragging and spinning the standard picker was the same as in study 1, but it was now outperformed by our minimized Thumb-Roll Force Picker, for which users were 500 ms faster. All users reported that they developed a "pumping" strategy to get faster: Instead of applying a strong force over a long time, they pressed hard, then released a bit, and then pressed harder again, to navigate far distances. While not significant, the differences in *Time* between both TECHNIQUEs increased with RANGE, from 177 ms for RANGE 10 to 240 ms for RANGE 101, and with DISTANCE, from 0 ms for OneStep to 240 ms for 80% DISTANCE. Hence, Thumb-Roll paid off especially when navigating larger RANGEs or DISTANCEs, apparently due to the speed boost that applying strong force triggered. *Success* was not significantly different between both TECHNIQUES ($\chi^2(1) = 2.25$), and was again similar to study 1.

In summary, our studies show that Thumb-Roll was the most promising technique for a force-based picker on force-sensitive smartphones, while requiring only a fraction of the gesture footprint (6%) and display footprint (20% of the height) of a standard picker as used on these systems today. Taking *all* touch points from thumb-rolling into account, trained users will cover an area of 10.2×5.3 mm, such that the Force Picker should be placed farther than this to avoid occlusion by the thumb. After 1:10 h of training, users were able to select values faster with the minimized Thumb-Roll Force Picker than with the standard picker without significant loss of accuracy (*Success* rates of 99% for the minimized Thumb-Roll Force Picker vs. 100% for the standard picker).

Below, we present some application examples that could benefit from using the minimized Thumb-Roll Force Picker instead of the system picker.

APPLICATION EXAMPLES

Space-Efficient In-Row Value Selection

As illustrated in Fig. 1 and the video figure, our Force Picker can be used to pick values without moving context information off-screen like standard pickers: To add a new calendar entry, for example, the user needs to specify a date, start time, and duration. With standard pickers, the user taps on, e.g., the start time, and the associated picker slides in, moving contextual information like the end time or invitees down and often off-screen to make space for the picker widget.

Using our Force Picker, however, we can keep all that information in place. When the user places her thumb on the hour digits of the start time, the Force Picker appears in the white space next to the label in the same row. It shows the hour currently selected, with the next values above and below it. It needs no more vertical space than the value label it is controlling, so nothing else on the screen moves around, which keeps important contextual information on screen and makes the interaction more serene. Using Thumb-Roll, the user now adjusts the hour through the picker, starting at the value the label displayed. Lifting the thumb fades out the Force Picker again, and updates the label to show the selected value. Minutes or dates can be selected similarly. Any mobile apps that work with times and dates, whether for searching flights and hotels, retrieving credit card statements, using public transport, or planning a drive, can benefit in a similar way.

Parameterized Shortcuts

Apple's recent smartphones let users apply a "force touch" on certain icons to display a list of context-specific shortcuts. However, since Apple only distinguishes two force levels in its UI, choosing from that list then requires either dragging or a secondary tap. The list also covers any underlying content that may have been useful for context, and the length of the list is limited by the size of the screen to avoid scrolling. An example is setting a countdown timer via force-touching the timer icon in the iOS Control Center: the user can only pick a timer for one, five, 20, or 60 minutes from the list that appears.

Using the Force Picker instead, force-touching on the countdown timer icon can fade in the Force Picker above the thumb, letting the user pick a value between 1 and 60 minutes directly by using force and the Thumb-Roll technique. Once

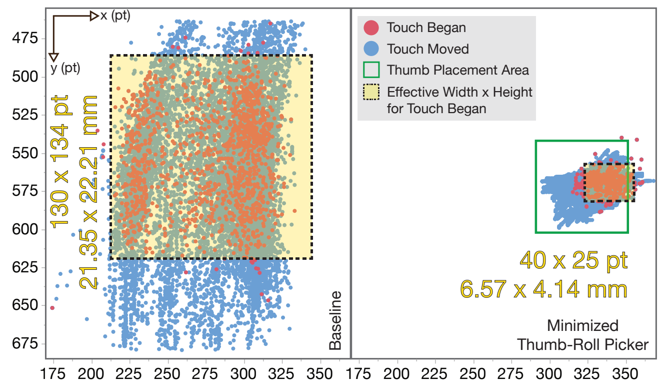


Figure 14. Thumb gesture footprint for study 3 for the Baseline picker (left) and the minimized Thumb-Roll picker (right). The footprint for controlling the minimized Thumb-Roll Force Picker was much smaller than dragging and spinning. The yellow overlays denote 96% of all first touch contact points: Thumb-Roll required only 6% of the space that users needed for dragging ans spinning the Baseline picker.

she reaches the desired countdown time, she lifts her thumb off the screen to start the timer (see video figure). Similarly, our Force Picker can be used in home control apps, e.g., to quickly set the volume of a stereo between 0 and 100%.

Self-Paced Browsing in Immersive Applications

Immersive applications like photo browsers or video editing apps should use the screen real estate for content, and reduce the footprint of widgets and other "debris" to a minimum. Therefore, browsing through a set of photos usually requires swiping left or right on the screen repeatedly. However, this still occludes the content and can be tedious and slow.

Using the minimized Thumb-Roll Force Picker instead, the user can browse photos at her desired pace to find the picture she is looking for, while occluding only the smallest possible area (a touch point) of the picture. Similarly, the technique can be used to flip through pages in an ebook or PDF. Indeed, this may remind the user of "thumbing" through pages in a physical book, because both the rolling thumb movement and the application of measured force are somewhat similar. Trimming a video clip can be simplified in the same way: When placing the thumb in the lower left corner of the screen, the Force Picker lets the user find the frame to set the first cut mark using Thumb-Roll. She selects it by lifting her thumb off the screen. The same gesture in the lower right corner of the screen defines the end frame of the video clip.

While these examples assume displaying the currently selected photo, page, or frame in real time, the Force Picker can also display their numbers above the user's thumb.

LIMITATIONS AND FUTURE WORK

In our studies, we only tested input for ordered values, and using only the picker as the form of visualization. However, we expect similar results for related visualizations, such as tables, and also for picking alphabetically ordered strings, like selecting a country name from a drop-down list when entering a shipping address on a website, as long as the user can approximately anticipate how far she has to navigate from the current value to the target value.

Unlike most standard pickers, our picker visualizations ordered values bottom-up. We used this reversed order to achieve a more natural mapping for force input. Somewhat surprisingly, this turned out not to be an issue, although all participants were familiar with standard pickers. Six users launched discussion after the study, and when asking them whether they noticed the reversed mapping, they all denied.

We only displayed three values at a time, like the Android system picker, to allow us to minimize the picker for studies 2 and 3, which would not have worked with the seven items the iOS 10 system picker shows. The physical size of our standard picker, and the spinning gestures, were using the iOS system defaults. Interestingly, Apple also seems to have noticed that their system picker required a significant amount of screen space, since for iOS 11, its height has been reduced to 126 pt (20.89 mm). This matches the effective height that we derived from studies 1 and 3 for our standard picker. Hence, the iOS 11 picker can only show five items at a time for each picker wheel, with the highest and the lowest value vertically compressed.

For our standard picker, users had to tap on a 'select' button to confirm value input, but existing pickers also require the user to tap somewhere outside the picker to confirm the value and close them. Tapping accounted for 275 ms ($SD = 40$ ms) of the trial completion time. Note, however, that our force techniques also included trial selection time. According to [5], confirming force input by lifting the thumb requires 240 ms.

The iPhone touch sensor only reports the center of each touch point, not the entire ellipsis. Actual footprints are therefore slightly larger, depending on thumb size.

Using Thumb-Roll directly at the edge of the smartphone screen will not work when the thumb moves beyond the digitizer sensing area. Therefore, the input area for the Thumb-Roll gesture should be placed accordingly.

In future work, we would like to reevaluate Press-Through with a stronger force sensor that the user cannot drive into saturation. This way, we could provide immediate feedback when reducing force from any attainable force level. Finally, we would also like to examine whether our results also scale to elderly people: Force input is more difficult for this group to control [12], and the minimized Thumb-Roll Picker might turn out to be too small to read.

CONCLUSION

In this paper, we presented a way to reduce the screen real estate for picking values from long, ordered ranges on smartphones. While existing controls, like pickers or tables, require a significant amount of gesture footprint for dragging and spinning, we exploited the force sensors on modern smartphones to reduce the gesture footprint to the size of the thumb. We conducted three experiments to find a force-based technique that allows for bidirectional control of such a picker, while still fitting inside a standard table row. Our first experiment identified Thumb-Roll, a technique that changes direction upon gently rolling the thumb to the left or right before applying force, as the fastest and most accurate technique to control a standard-sized picker by force input. Although users were slower, accuracy was almost identical, and Thumb-Roll reduced the gesture footprint by 88%. We then showed that this allows the Force Picker to be shrunk down to a minimum size, without affecting accuracy, although it slightly slowed down untrained subjects. Our final study showed that trained participants can actually be significantly faster using the minimized Thumb-Roll Force Picker than with a standard spin & drag picker, without significant loss of accuracy. With Thumb-Roll, these users needed only 6% of the gesture space required for dragging and spinning, and the minimized Force Picker takes up only 20% of the height of the iOS 10 system picker. We provided application examples for value input on smartphones that benefit from the much smaller footprint and in-place touch interaction of the minimized Thumb-Roll Force Picker. We hope that our findings inspire other researchers and practitioners to further improve our key daily interactions with our smartphones through their force-sensing capabilities.

ACKNOWLEDGMENTS

We thank Paulina Reijsmeijer for helping with the figures. This work was funded by the German B-IT Foundation.

REFERENCES

1. Axel Antoine, Sylvain Malacria, and Géry Casiez. 2017. ForceEdge: Controlling Autoscroll on Both Desktop and Mobile Computers Using the Force. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3281–3292. DOI: http://dx.doi.org/10.1145/3025453.3025605

2. Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. 1990. The Design Space of Input Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. ACM, New York, NY, USA, 117–124. DOI: http://dx.doi.org/10.1145/97243.97263

3. Jared Cechanowicz, Pourang Irani, and Sriram Subramanian. 2007. Augmenting the Mouse with Pressure Sensitive Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1385–1394. DOI: http://dx.doi.org/10.1145/1240624.1240835

4. Christian Corsten, Bjoern Daehlmann, Simon Voelker, and Jan Borchers. 2017a. BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4654–4666. DOI: http://dx.doi.org/10.1145/3025453.3025565

5. Christian Corsten, Simon Voelker, and Jan Borchers. 2017b. Release, Don't Wait! Reliable Force Input Confirmation with Quick Release. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 246–251. DOI: http://dx.doi.org/10.1145/3132272.3134116

6. Nielsen Norman Group. 2015. Slider Design: Rules of Thumb. https://www.nngroup.com/articles/gui-slider-controls/. (2015). [Online; accessed September 19, 2017].

7. Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. 1998. Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 17–24. DOI: http://dx.doi.org/10.1145/274644.274647

8. Ken Hinckley and Mike Sinclair. 1999. Touch-Sensing Input Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 223–230. DOI: http://dx.doi.org/10.1145/302979.303045

9. David Holman, Andreas Hollatz, Amartya Banerjee, and Roel Vertegaal. 2013. Unifone: Designing for Auxiliary Finger Input in One-Handed Mobile Interactions. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13)*. ACM, New York, NY, USA, 177–184. DOI: http://dx.doi.org/10.1145/2460625.2460653

10. Apple Inc. 2017a. Apple Human Interface Guidelines. https://developer.apple.com/design/. (2017). [Online; accessed September 8, 2017].

11. Apple Inc. 2017b. iOS 10 UIKit API Reference. https://developer.apple.com/reference/uikit/uitouch. (2017). [Online; accessed September 19, 2017].

12. Hiroshi Kinoshita and Peter R. Francis. 1996. A Comparison of Prehension Force Control in Young and Elderly Individuals. *European Journal of Applied Physiology and Occupational Physiology* 74, 5 (01 Nov 1996), 450–460. DOI: http://dx.doi.org/10.1007/BF02337726

13. Ross McLachlan, Daniel Boland, and Stephen Brewster. 2014. Transient and Transitional States: Pressure as an Auxiliary Input Modality for Bimanual Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 401–410. DOI: http://dx.doi.org/10.1145/2556288.2557260

14. 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

15. Sachi Mizobuchi, Shinya Terasaki, Turo Keski-Jaskari, Jari Nousiainen, Matti Ryynanen, and Miika Silfverberg. 2005. Making an Impression: Force-Controlled Pen Input for Handheld Devices. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 1661–1664. DOI: http://dx.doi.org/10.1145/1056808.1056991

16. R. Kevin Nelson. 2015. Exploring Apple's 3D Touch. https://medium.com/@rknla/exploring-apple-s-3d-touch-f5980ef45af5#.pijkgrkw0. (2015). [Online; accessed September 19, 2017].

17. Alexander Ng and Stephen A. Brewster. 2016. Investigating Pressure Input and Haptic Feedback for In-Car Touchscreens and Touch Surfaces. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (Automotive'UI 16)*. ACM, New York, NY, USA, 121–128. DOI: http://dx.doi.org/10.1145/3003715.3005420

18. Sebastien Pelurson and Laurence Nigay. 2016. Bimanual Input for Multiscale Navigation with Pressure and Touch Gestures. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction (ICMI 2016)*. ACM, New York, NY, USA, 145–152. DOI: http://dx.doi.org/10.1145/2993148.2993152

19. Mahfuz Rahman, Sean Gustafson, Pourang Irani, and Sriram Subramanian. 2009. Tilt Techniques: Investigating the Dexterity of Wrist-Based Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1943–1952. DOI: http://dx.doi.org/10.1145/1518701.1518997

20. Gonzalo Ramos, Matthew Boulos, and Ravin Balakrishnan. 2004. Pressure Widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 487–494. DOI:http://dx.doi.org/10.1145/985692.985754

21. Jef Raskin. 2000. *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

22. Jun Rekimoto and Carsten Schwesig. 2006. PreSenseII: Bi-Directional Touch and Pressure Sensing Interactions with Tactile Feedback. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. ACM, New York, NY, USA, 1253–1258. DOI: http://dx.doi.org/10.1145/1125451.1125685

23. Anne Roudaut, Eric Lecolinet, and Yves Guiard. 2009. MicroRolls: Expanding Touch-Screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 927–936. DOI: http://dx.doi.org/10.1145/1518701.1518843

24. Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. 2004. Gummi: A Bendable Computer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 263–270. DOI:http://dx.doi.org/10.1145/985692.985726

25. Kang Shi, Pourang Irani, Sean Gustafson, and Sriram Subramanian. 2008. PressureFish: A Method to Improve Control of Discrete Pressure-Based Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1295–1298. DOI: http://dx.doi.org/10.1145/1357054.1357256

26. Kang Shi, Sriram Subramanian, and Pourang Irani. 2009. *PressureMove: Pressure Input with Mouse Movement*. Springer Berlin Heidelberg, Berlin, Heidelberg, 25–39. DOI:http://dx.doi.org/10.1007/978-3-642-03658-3_7

27. R. William Soukoreff and I. Scott MacKenzie. 2004. Towards a Standard for Pointing Device Evaluation, Perspectives on 27 Years of Fitts' Law Research in HCI. *International Journal of Human-Computer Studies* 61, 6 (Dec. 2004), 751–789. DOI: http://dx.doi.org/10.1016/j.ijhcs.2004.09.001

28. Daniel Spelmezan, Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. 2013a. Controlling Widgets with One Power-Up Button. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 71–74. DOI:http://dx.doi.org/10.1145/2501988.2502025

29. Daniel Spelmezan, Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. 2013b. 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

30. Craig Stewart, Michael Rohs, Sven Kratz, and Georg Essl. 2010. Characteristics of Pressure-Based Input for Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 801–810. DOI: http://dx.doi.org/10.1145/1753326.1753444

31. Graham Wilson, Stephen A. Brewster, Martin Halvey, Andrew Crossan, and Craig Stewart. 2011. The Effects of Walking, Feedback and Control Method on Pressure-Based Interaction. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*. ACM, New York, NY, USA, 147–156. DOI:http://dx.doi.org/10.1145/2037373.2037397

32. Graham Wilson, Craig Stewart, and Stephen A. Brewster. 2010. Pressure-Based Menu Selection for Mobile Devices. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '10)*. ACM, New York, NY, USA, 181–190. DOI: http://dx.doi.org/10.1145/1851600.1851631