Extending the Vocabulary of Touch Events with ThumbRock

David Bonnet
bonnet@lri.fr

Caroline Appert
appert@lri.fr

Michel Beaudouin-Lafon
mbl@lri.fr

Univ Paris-Sud & CNRS (LRI) — INRIA
F-91405 Orsay, France

ABSTRACT

Compared with mouse-based interaction on a desktop interface, touch-based interaction on a mobile device is quite limited: most applications only support tapping and dragging to perform simple gestures. Finger rolling provides an alternative to tapping but uses a recognition process that relies on either per-user calibration, explicit delimiters or extra hardware, making it difficult to integrate into current touch-based mobile devices. This paper introduces ThumbRock, a ready-to-use micro gesture that consists in rolling the thumb back and forth on the touchscreen. Our algorithm recognizes ThumbRocks with more than 96% accuracy without calibration nor explicit delimiter by analyzing the data provided by the touch screen with a low computational cost. The full trace of the gesture is analyzed incrementally to ensure compatibility with other events and to support real-time feedback. This also makes it possible to create a continuous control space as we illustrate with our MicroSlider, a 1D slider manipulated with thumb rolling gestures.

Keywords: Touch event, Gesture vocabulary, Finger rolling, Incremental recognizer, Touchscreen, Mobile device.

Index Terms: H.5.2 [User Interfaces]: Graphical user interfaces (GUI)—Input devices and strategies, Interaction techniques;

1 INTRODUCTION

Touch-enabled mobile devices support content manipulations mostly through finger taps and drag gestures. Since drag-operations often default to navigation, taps often trigger modes in which subsequent drags are dedicated to other actions. A mode-switching tap can be distinguished from a regular one based on a delay, such as tap-and-hold to drag an item, or a specific touch location, such as Bezel Swipe [19] to apply a command (e.g. select, cut or paste) to an object. However, with such techniques, users can perform only one action at a time since lifting the finger both triggers the command and deactivates the mode. This makes repeated tasks such as selecting several objects or dragging them to distant locations cumbersome. Allowing users to perform several actions in a row fluidly without leaving a mode requires the use of in-drag events, i.e. events that can occur while the finger is touching the screen.

Finger rolling has been introduced as a way to input in-drag events with SimPress [2]. As soon as the contact area of the index finger goes beyond a predefined threshold, a SimPress event is detected. However, this event is highly dependent on the user and the finger that interacts with the surface. Not only does the recognizer require per-user calibration for setting the threshold but it also always selects the single point that is right above the contact area, which can vary especially when the user interacts with the thumb. Wang et al. [25] describe an algorithm that removes per-user calibration for SimPress, but it is not robust when used with the thumb because it does not account for its high variability.

Graphics Interface Conference 2013
29-31 May, Regina, Saskatchewan, Canada

The ThumbRock gesture introduced in this article builds on this previous work by considering finger rolling as *ready-to-use events*. ThumbRocks consist in rolling the finger back and forth on the touch screen, so that the start and end locations of the gesture are the same, similar to a tap. The algorithm for recognizing ThumbRocks is incremental and does not require training nor explicit delimiters. It also detects the start of the gesture early enough to quickly disable any conflicting drag action, and captures the full trace of the rolling gesture to convey more than a single bit of information to the system. For example, ThumbRocks with different amplitudes and/or durations can be interpreted as different *discrete* events. Moreover, the gesture trace can be used as a control space for setting a *continuous* value, as demonstrated with our MicroSlider. Our algorithm supports both thumb and index finger rolling gestures. The recognized in-drag event can be easily integrated into existing touch-enabled mobile devices, supporting rich and fluid interactions.

After an overview of related work, this article describes the ThumbRock gesture and the recognizer we developed. It then presents a controlled experiment that assesses the robustness of the recognizer and the usability of the technique. Finally, it describes several applications that can benefit from ThumbRock; In particular, it presents the MicroSlider technique along with an empirical evaluation of its precision.

2 RELATED WORK

This section reviews techniques aimed at augmenting the expressivity of single-point input on touchscreens.

Small finger movements Figure 1 shows how the ThumbRock builds on the SimPress [2] and MicroRolls [20] techniques that have already used finger rolling. SimPress [2], initially introduced for tabletop interaction, was adapted to mobile devices with the Fat Thumb [4] technique that assigns a different meaning to a tap or a drag depending on the size of the contact area. As for SimPress, Fat Thumb uses hard-coded absolute thresholds so that both techniques require per-user calibration to handle inter-user variability. Also, because the size of the contact area depends on the thumb posture (intra-user variability), Fat Thumb is more effective and reliable when used in the center of the screen. While there is some evidence that some locations are more comfortable than others for thumb-based interactions [10], only considering the center of the screen is quite restrictive since the thumb can access most screen locations with good accuracy [16]. Also, dragging with a higher contact area generates more friction and fatigue. MicroRolls [20] introduces a set of simple unidirectional and circular rolls performed

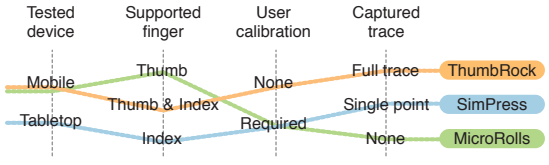

Figure 1: Review of finger rolling techniques characteristics.

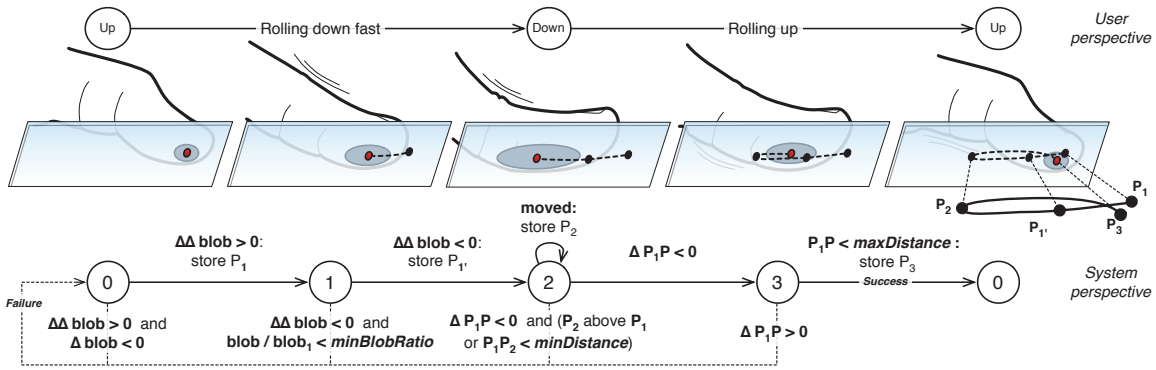

Figure 2: Top: Postures of the thumb during a ThumbRock. Bottom: The ThumbRock recognizer (see text for the notations).

with the thumb. The recognizer for these gestures requires a learning phase and an explicit segmentation of the gesture with down and up events. MicroRolls therefore cannot be used while dragging a finger. Finally, we note that neither MicroRolls, SimPress nor Fat Thumb were tested to assess their recognition rates when compared with small drag movements.

Finger taps variations Several techniques augment the expressivity of a simple touch. Some use additional sensors: TapSense [5] uses a microphone attached to the casing of the device, while Sensor Synaesthesia [8] uses acceleration data in combination with touch location to distinguish a "hard tap" from a "soft tap" by analyzing the vibration caused by the touch event.

Holz and Baudisch [9] use fingerprints to capture the three-dimensional finger posture with high precision. However, such technology is not available on current handheld devices, which only provide an elliptical approximation of the finger contact area, defined by its center and by the size of its major axis. All the above tap variations can be used while dragging using a technique such as Lift-and-tap [13]. However, this introduces an ambiguity with the common sequence of touch/release events produced while clutching. It also creates a precision problem because users may have difficulty landing their finger exactly where they lifted it.

Mode switching techniques Several techniques change the semantics of a drag without leaving the dragging quasi-mode and its associated kinesthetic feedback [23], but they are not well suited to mobile devices. For example, iconic gestures such as a pigtail [7] or other gesture [24] are usually performed at the end of the drag, which is too late to affect its default action. Furthermore iconic gestures use more space than finger rolling gestures, which is an issue for mobile devices. Using a time-out ("dwelling") is a simple alternative but is already overused in current interfaces. It is also time-consuming [7] and prone to accidental activations in case of hesitations. Tilt input is used in, e.g., Tip-to-select [8], but is also noisy so that exaggerated gestures such as DoubleFlip [21] should be used to avoid false triggers. Finally, pressure is used in Force Gestures [6], but the necessary hardware is not present on current commercial devices. Note also that applying a force orthogonal to the surface while dragging can cause fatigue because the increased contact area generates more friction.

The rest of this article describes and evaluates ThumbRock, a reliable in-drag event that does not conflict with other events.

3 DESCRIPTION

From a user perspective, performing a ThumbRock consists of quickly rolling the thumb back and forth so that the thumb knuckle is successively pushed down and moved back up (top of Figure 2). This specific movement acts only on the mid-joint of the thumb while keeping it in contact with the surface so that it is easy to get back to the initial location. Also, even though it is a micro movement that does not require much effort, it is a conscious action that should not be prone to accidental activations.

From a system perspective, rolling the thumb affects the data captured by the touchscreen [2, 20]: the contact area (blob size) increases and its center (touch location) moves slightly (Figure 2). Using either one of these two features alone is not sufficient to reliably detect a ThumbRock. In particular, using only the touch trajectory leads to potential confusion between rolls and small drags [9]. We also found that using hard-coded thresholds for the blob size [2] requires calibration to account for intra-user variability when using the gesture in different contexts, and for inter-user variability. The latter is especially important given the high amount of variability that we observed among thumbs, from almost straight thumbs to very bendable ones. To overcome these limitations and increase robustness, the ThumbRock recognizer (i) tracks both the touch trajectory and the blob size[1] and (ii) uses relative thresholds such as percentage increase or sign changes.

Our goal when designing the ThumbRock recognizer was to identify a system description of the different physiological states and transitions. To that end, we captured typical touch trajectories and blob size changes that occur during a ThumbRock. We asked six participants whose thumbs differ in size (from 58mm to 82mm, 69 ± 8.9mm average) and flexibility (defined as the bending angle when the thumb is in full extension, from 14° to 56°, $29.8 \pm 17.5°$ average) to perform both random drag movements and ThumbRocks at different locations on the screen. The setting was similar to that of the experiment described in the next section. We used the collected data to identify potential signatures of a ThumbRock gesture and to design the corresponding recognizer. To accurately detect the transitions between the different states, we considered the set of measures shown in Figure 3. We used a trial-and-error approach to optimize both segmentation and recognition accuracy. Namely, segmentation should be as precise as possible, and recognition should maximize hits while minimizing false positives. Our final algorithm is described in Figure 2 and detailed below.

To detect the starting point, P_1: the recognizer analyzes the evolution of the blob size. Figure 3 shows that simply detecting an increase in the blob size (Δ blob > 0) would accept too long a sequence of points. Detecting an *acceleration* of the increase in blob size instead ($\Delta\Delta$ blob > 0) is more accurate. At this point the recognizer stores P_1 and enters state 1. When a deceleration of the increase in blob size ($\Delta\Delta$ blob < 0) occurs in state 1, the recognizer enters state 2, where the confidence of capturing a ThumbRock is higher, only if the relative increase in blob size is large enough ($blob/blob_1 > minBlobRatio = 1.29$, where $blob$ is the current size of the blob and $blob_1$ its size at P_1).

[1] We use the length of the major axis of the contact ellipse as blob size.

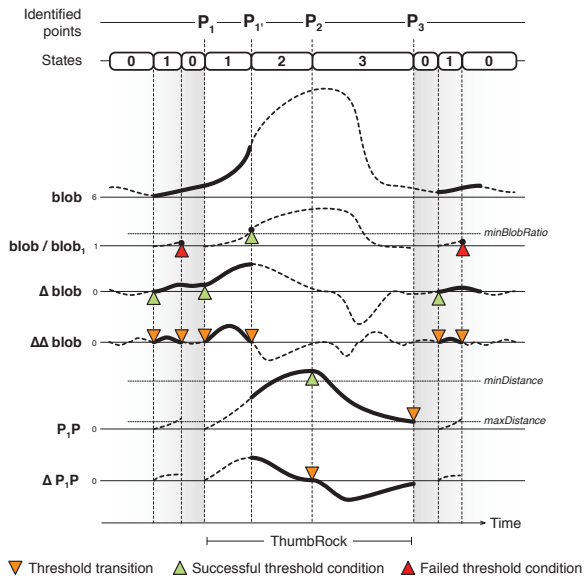

Figure 3: Raw and computed measures during a ThumbRock. Bold parts show the measures considered at each step of the movement.

To detect the point where the rolling back sub-movement starts, P_2: the recognizer focuses on the touch trajectory rather than the blob size. We found this to be more reliable because of the variability in thumb anatomy: when in full extension, some users have most of the distal phalange in contact with the surface while others actually lift the tip of the thumb so that the blob size temporarily shrinks. Collected data revealed that a decrease in the distance between the current point and the starting point ($\Delta P_1P < 0$, where P is the current touch location) was a good criterion for recording P_2. If this decrease occurs during a movement with a large enough amplitude ($P_1P > minDistance = 8.0$ mm) and that is consistent with the thumb anatomy (P_2 is below P_1), the recognizer enters state 3.

To detect the end point, P_3: the current touch location must get close enough to the starting point ($P_1P < maxDistance = 3.6$ mm). If P_1P increases before this threshold is reached, the recognizer stops with a miss. The user can thus cancel a ThumbRock by simply moving away from P_1 a bit during the rolling back movement. Otherwise, a ThumbRock is recognized.

Designing an incremental recognizer is fundamental to introducing finger rolling as a new event. First, the recognition fails as early as possible to *avoid conflict with other drag-based events*. Second, *real-time feedback* can assist the user while performing the gesture and reinforce interface responsiveness to ThumbRock events. As in previous work on ambiguous input [22], the probability of recognizing a ThumbRock is higher each time the recognizer reaches a new step. Figure 4 illustrates the visual and audio feedback we implemented to guide the user along these steps. In the middle of the first rolling back sub-movement (P_1' point on Figure 2), a *tick* sound occurs and/or a circle is displayed. The circle then grows and shrinks according to the distance P_1P while staying large enough to be visible while the thumb is in contact. When a ThumbRock is recognized, the circle turns into a bursting bubble. If audio feedback is enabled, the user hears a *tack* sound. Visual and audio feedback can be used together or independently, and can be complemented or replaced by view-specific feedback such as buttons that look like physical switches (Figure 11).

Introducing a new basic event must also incur a *small computational cost*. Each time a new drag event occurs, our algorithm compares it only to the previous drag event, decomposing the whole movement into small steps that are analyzed in real time. This approach is also used for recognizing "corner events" [1], Rubbing [15] and CycloStar [14]. It differs from other approaches such as the one used for MicroRolls [20], where the whole touch trace is sent to a training-based recognition algorithm after the final up event, or the one used for Pigtail delimiters [7], which uses a continuous sliding time interval over which a pattern is analyzed.

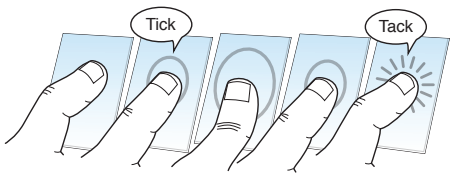

Figure 4: Visual and audio feedback during a ThumbRock.

Finally, our algorithm does not require per-user training, making the ThumbRock a *ready-to-use* event. It relies on a definition of the ThumbRock gesture in system terms to avoid using reference templates that are vulnerable to inter- and intra-user variability. Here, the algorithm captures a generic profile of the gesture by computing relative differences between successive touch positions using first and second degree derivatives (e.g. $\Delta blob$ and $\Delta\Delta blob$), or between the current position and a fixed reference point (e.g. $blob/blob_1$ and P_1P). We now report on an experiment that demonstrates the robustness of this approach.

4 EVALUATION

To be valuable as a new event on small touch screens, the ThumbRock must be (i) **usable**, i.e. easy to learn and fast to perform, (ii) **accurate**, i.e. reliably recognized by the system, and (iii) **compatible**, i.e. not confounded with other existing events. To evaluate these three aspects, we conducted an experiment divided into two sessions: *true positives* and *false positives*.

4.1 True Positives

The goal of this session is to evaluate how easily users can perform ThumbRocks at different screen locations (*usability*) within a sequence of other events (*compatibility*) and how many of them are recognized (*accuracy*).

Task and Stimuli Participants must perform ThumbRocks at 15 different locations[2] (*Location* factor, Figure 6-left) on the touchscreen with different prior and subsequent touch actions (*Sequence* factor): TOUCH/RELEASE, DWELL, DRAG (Figure 5). Although drags can be performed in many directions, we considered small straight movements along the cardinal orientation (NS, EW, SN, WE) in order to keep the duration of the experiment reasonable. For the same reason, we chose pairs of identical pre/post touch actions. We end up with six values for *Sequence*: TOUCH/RELEASE, DWELL, DRAG-NS, DRAG-SN, DRAG-EW and DRAG-WE.

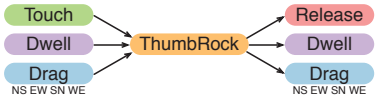

Figure 5: Sequences of events in the true positives session.

For all *Sequence* conditions, participants must perform a ThumbRock as close as possible to the center of a crosshair appearing at the target location (Figure 7-(a)). If the thumb is close enough to

[2]Following our observations in a pilot study, the D4 location in the lower right (resp. left) corner was automatically discarded for right-handed (resp. left-handed) participants, as this location does not provide enough amplitude to perform a ThumbRock.

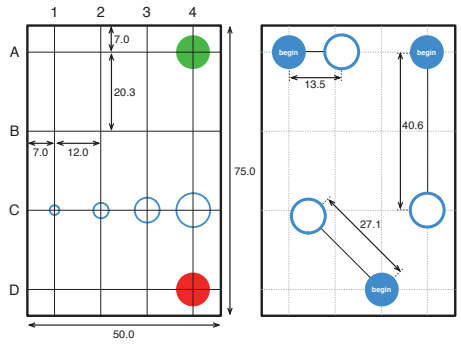

Figure 6: Left: The grid of locations used in the experiment. In the training session, the radius of a circle is a function of the number of ThumbRocks performed successfully at this location. A circle turns red when there were 10 unrecognized gestures at this location or green after 5 successful attempts. Right: Visual guides for the false positives session showing the 3 different segment lengths for the targets A1, A4 and D3 used in the drag tasks. (Lengths in mm.)

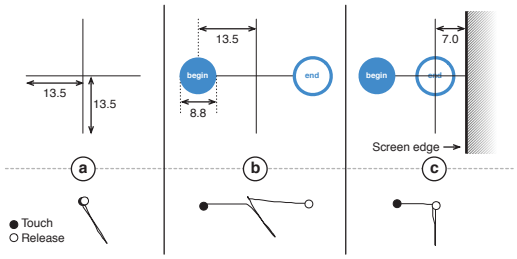

Figure 7: Visual guides for the true positives session and a corresponding example of a ThumbRock trace for the TOUCH/RELEASE or DWELL condition (a), the DRAG-WE condition on columns 2 and 3 (b) and near the screen edge on column 4 (c). (Lengths in mm.)

that crosshair (at most 4.7 mm from its center), the white background turns green and a beep occurs to announce that a ThumbRock can be performed (*go signal*). The action before and after the ThumbRock depend on the *Sequence* condition:

- TOUCH/RELEASE: the participant touches the center of the crosshair which triggers the *go signal*, performs a ThumbRock and immediately releases his thumb.
- DWELL: after touching the center of the crosshair, the participant must stay still (dwell) for 500 ms within a radius of 4.7 mm (30 px). When the dwell timer expires, the *go signal* instructs the participant to perform a ThumbRock. He must then dwell again and release the thumb as soon as he hears a *done signal* and sees the screen turning white.
- DRAG-{NS, EW, SN, WE}: two disks surround the crosshair: a blue and a white one (Figure 7-(b)). The participant touches the blue disk and drags to the middle of the crosshair. At this point the *go signal* occurs. The participant must perform a ThumbRock and immediately drag to the white disk, where the *done signal* indicates that he can release the thumb. When a target is too close to a bezel to display the begin or end disk, the disk is positioned on the target. If it is the begin disk, the participant must perform a ThumbRock immediately after his thumb hits the surface. If it is the end disk, the participant must lift the thumb immediately after performing the ThumbRock (Figure 7-(c)).

Presentation To avoid asking participants to perform too many uncomfortable gestures, we introduce a testing phase where participants must perform 5 successive ThumbRocks at each location. All the target locations are displayed and the participant is free to go over them in any order (Figure 6-left). At each successful ThumbRock, a blue circle grows until it turns into a green disk at the 5th successful ThumbRock. In case of repeated failures (10 unrecognized gestures), a red disk appears and the location is discarded for the rest of the session.

The rest of the experiment is divided into two *Feedback* conditions, AUDIO and VISUAL (Figure 4). Within a *Feedback* condition, a participant performs six series of trials, one per *Sequence* (TOUCH/RELEASE, DWELL, DRAG-{NS, EW, SN, WE}). Presentation order for *Feedback* and *Sequence* is counterbalanced with a Latin square. A series is divided into four blocks containing up to 15 trials each, one per target location that was not excluded after the testing phase, presented in a random order. The first block is a practice block, the three others are testing blocks. With such a design, we collect up to 540 trials per participant.

After a series is completed, the participant reports the perceived difficulty of performing a ThumbRock at each location using a Likert scale (from 1:very easy to 5:very difficult). Participants are allowed to take a break between each series.

4.2 False Positives

In the previous session, the *compatibility* was only assessed for touch events performed in sequence with a ThumbRock. In this session, the goal is to test the robustness of the recognizer against dragging gestures that are most likely to be confounded with ThumbRocks. We consider back-and-forth drags that mimic the shape of the typical trajectory captured by the touch screen during a ThumbRock. Such gestures occur, e.g., in the *Rubbing* [15] and *CycloPan* [14] techniques. We also consider simple taps or *HardTaps* [8] where the thumb, when hitting the surface, may produce a blob increase similar to the beginning of a ThumbRock.

Task and Stimuli The session is divided in two parts, *drags* and *taps*. In both parts, the ThumbRock recognizer is active with both visual and audio feedbacks enabled.

In the *drags* part, participants have to perform reciprocal drags, i.e. dragging back and forth between two disks without lifting the thumb. We vary the amplitude and orientation of reciprocal drags because it may affect the surface of the contact area. For example, when a right-handed user performs high-amplitude reciprocal drags along the NE-SW diagonal, the contact area is fairly large in the NE area while it is much smaller in the SW area.

At the beginning of each trial, a blue disk and a white disk appear to guide the participant (Figure 6-right). The participant must reach the blue disk without lifting the thumb. When the thumb reaches the blue disk, a *tick* sound occurs and the blue disk turns white while the white disk turns blue, becoming the next disk to reach. The trial ends when the user has acquired 7 blue disks[3]. The begin disk is located at one of the 16 target locations on the grid (Figure 6-left) and the location of the ending disk depends on two factors: *Orientation* ({N, S, E, W, NW, NE, SW, SE}) and *Amplitude* ({SMALL (13.5 mm), MEDIUM (27.1 mm), LARGE (40.6 mm)}). Because all the *Orientation* × *Amplitude* combinations are not always available according to the position of the target on the grid, we only consider the meaningful combinations.

In the *taps* part, participants have to perform taps, i.e. touch then release as close as possible to the center of the crosshair appearing at each of the 16 target locations. The participants perform blocks of SOFT (i.e. regular) and HARD taps.

[3]Participants can therefore trigger at most three false ThumbRocks.

Presentation For the *drags* part, participants start with a practice session of 16 trials with randomly selected target *Locations*, *Directions* and *Amplitudes*. Then, they perform eight blocks, one per *Direction*, presented in a random order, i.e. a total of 180 trials per participant. For the *taps* part, participants perform two series (SOFT and HARD tap) of one training block and two recorded blocks, i.e. a total of 64 trials per participant.

4.3 Participants and Presentation

Twelve unpaid volunteers (11 male, 1 female), 25 to 35 years old (average 29.3, median 27), participated in this two-session study. None had taken part in the pilot study for calibrating the ThumbRock recognizer. The study lasted about one hour and the two sessions were presented one after the other in an order that was counterbalanced across participants. For the one left-handed participant, the grid was mirrored so as to keep the target location constant relatively to the hand. A preliminary questionnaire revealed that 8 participants used smartphones daily.

4.4 Apparatus

We conducted the experiment on a 2^{nd} generation Apple iPod Touch running iOS 4.2.1 with a touch precision of 163 PPI. The touchscreen captures touches at a maximum frequency of 62.5 Hz. Participants sat on a chair and had to interact only with the thumb of the dominant hand holding the device and were not allowed to rest the dominant hand anywhere (table, lap). Thus, the study is limited to the use of ThumbRocks in a static setting, leaving a study in mobility conditions to future work.

4.5 Results

4.5.1 Recognition rate

To assess the recognition rate in the *true positive* session, we consider the measure *Recognized*, which is True if the participant performed the right sequence of events with exactly one ThumbRock recognized at the right location. If 0 or more than 1 ThumbRocks were recognized, the trial was validated but *Recognized* was set to False. Otherwise, if there was a mistake in the sequence of events (e.g. dwell time too short or touch outside the begin disk), the background flashed red and the task was presented again.

At the end of the training session, location D3 was discarded for two participants while location C4 was discarded for one participant (Figure 6), leading to 360 measures for location D3, 396 for location C4 and 432 for all other locations. Figure 8-left shows the recognition rates for each target location by counting those excluded trials as misses. Without those misses, mean recognition rate is 98% ±2% for location D3 and 88% ±12% for location C4. In both cases, the mean recognition rate is very high: between 96% and 98%. *This shows that in most cases, the participant and the system agree on what a ThumbRock is.*

	Recognition rate (%)				Mean duration (s)				Perceived difficulty (1-5)			
	1	2	3	4	1	2	3	4	1	2	3	4
A	97 ±5	99 ±1	99 ±1	98 ±3	0.32 ±0.06	0.31 ±0.06	0.30 ±0.05	0.30 ±0.06	2.20 ±1.19	1.67 ±0.79	1.60 ±0.74	2.02 ±0.79
B	99 ±2	98 ±3	99 ±2	97 ±2	0.30 ±0.06	0.30 ±0.05	0.30 ±0.05	0.31 ±0.05	1.75 ±0.83	1.26 ±0.58	1.33 ±0.62	2.06 ±0.67
C	98 ±3	98 ±2	99 ±2	90 ±28	0.30 ±0.05	0.31 ±0.05	0.32 ±0.05	0.32 ±0.05	1.82 ±0.83	1.42 ±0.57	1.67 ±0.60	2.72 ±0.94
D	98 ±2	98 ±3	73 ±36		0.34 ±0.06	0.35 ±0.07	0.37 ±0.07		3.09 ±1.22	3.36 ±0.91	3.69 ±0.77	

Figure 8: Mean values and standard deviation per target location for the main collected measures. (Excluded trials are counted as misses)

Because the robustness of a recognition algorithm is assessed by both its recognition accuracy and the number of false positives it accepts, we counted the number of ThumbRocks that were recognized in the *false positives* session: 9 ThumbRocks were recognized in the dragging part (0.1%) and none in the tapping part (0%). We also assessed whether the feedback mechanisms we envision in real contexts of use could be disturbing. We counted how many times a feedback was triggered without recognition of a ThumbRock. This occurred 12 times in the *dragging* part (0.2%), never in the *taps* part (0%). These low numbers suggest that the first part of a ThumbRock could be recognized as an independent event, i.e., our algorithm could recognize SimPress events [2] performed with the thumb without any per-user calibration.

For the rest of the analyses, we exclude the misses (*Recognized*=False). There were 150 misses out of 6372 trials (2.4%), evenly distributed across the two *Feedback* conditions and the six *Sequence* conditions. Location D3 had by far the most misses: 12% of the trials at this location, 29% of all misses.

4.5.2 Speed and pointing accuracy

The *ThumbRock Duration* (TRD) is the time in seconds for performing a successful ThumbRock. Participants were fast: mean time for TRD was 0.32 ± 0.1s, compared with 0.5s for dwelling and more than 0.6s for tilting [17]. There was no effect of *Feedback* on TRD while we could have expected that the animated visual feedback could be a bit more disturbing than the more discrete audio feedback. However, there was an effect of *Sequence* on TRD ($F_{(5,55)}$=29, $p < 0.0001$): Participants were significantly faster in the TOUCH/RELEASE sequence (0.25s). Figure 8-center reports the mean duration of a ThumbRock by target location, illustrating the significant effect of *Location* on TRD (Welch ANOVA test: $F_{(14,146.8)}$=15, $p < 0.0001$). These differences in duration between locations match the perceived difficulty reported on Figure 8-right. The target location also has a significant effect on the perceived difficulty ($F_{(14,144)}$=23, $p < 0.0001$). In summary, ThumbRocks are fast and easy to perform on average but a bit more difficult and longer when the thumb is at its maximal extension (target location A1) and when it is parallel to the pinky (targets on row D).

To get a fair evaluation of the time taken by participant to perform a ThumbRock, we must include the failed attempts before performing a successful one. Note that the low number of misses supports the fact that the feedback mechanisms are explicit enough to let participants know whether their ThumbRock was recognized or not. By relaxing the algorithm to accept very degenerate cases of ThumbRocks (i.e., setting *minBlobRatio* to 1.15), we can extract the time interval between successive attempts to perform a ThumbRock, denoted as *Fail Duration* (FD). Figure 9 reports this time and the time for performing a successful ThumbRock by target location, which is the only factor that has a significant effect on FD ($F_{(14,144)}$=15, $p < 0.0001$). Even when adding TRD and FD to include unsuccessful attempts in the overall time to perform a ThumbRock, the duration remains small: 0.34 ± 0.22 s.

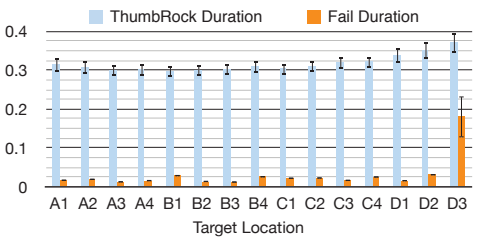

Figure 9: Mean time for performing a successful ThumbRock compared with mean time of a failure prior to a successful attempt.

The power of our recognizer lies not only in its accuracy but also in its ability to incrementally analyze the gesture in real time. To evaluate this aspect, we extracted the *begin* and *end* points of a ThumbRock in the DWELL and TOUCH/RELEASE conditions and measured their distance to points P_1 and P_3 captured by the recognizer: $distance(begin, P_1) = 0.7 \pm 1.2$ mm and $distance(end, P_3) = 3.6 \pm 1.8$ mm. (As expected, the latter is close to the *maxDistance* threshold used to recognize P_3.)

Capturing the rolling gesture in real time lets us provide feedback as soon as the recognizer identifies point $P_{1'}$. In the collected trials, feedback was triggered after 0.07s on average, or 20% of the overall ThumbRock duration. In terms of distance, this represents a movement of 7.8 mm (4.5 mm along the x-axis and 5.3 mm along the y-axis), or 60% of the overall ThumbRock amplitude (13 mm). For example, performing a ThumbRock on an item in a vertical list would move the view by 5.3 mm (34 pixels) along the y-axis before the feedback is triggered and navigation disabled. Similarly, performing a ThumbRock on an horizontal list (such as the iOS home screen) would move the view along the x-axis by only 4.5 mm (29 pixels). Because this is a small offset that takes place over a very short time interval, we found the disturbance to be minimal in the applications that we describe in the next section. These observations support the fact that the ThumbRock recognizer is fast enough to detect the rolling movement in order to minimize its effect on the ongoing drag action. The responsive feedback lets users quickly correct their gesture in case the beginning of a ThumbRock was not recognized.

Regarding the accuracy of acquiring a target with a ThumbRock, we consider the distance between the target location and the first point of the ThumbRock detected by the recognizer. There are neither simple nor interaction effects of *Location* and *Sequence* on this measure. The average distance is 4.1 ± 6.7 mm. By comparison, the mean distance between the target center and the captured location of a tap in the tapping part of the *false positives* session is 4.2 ± 7.0 mm for regular taps, and 4.7 ± 6.9 mm for hard taps. This supports the fact that users have similar or better accuracy when acquiring a target with a ThumbRock compared to a tap.

4.5.3 Index finger in two-hand usage

Even though single-handed use is critical for mobile devices, two-handed usage must also be considered as users often interact with their index finger while holding the device with the other hand. To check that our recognizer also effectively captures *IndexRocks* (ThumbRocks performed with the index finger), we conducted a follow-up study where we asked six participants (5 male, 1 female, mean age 33.3 years old) to take part in the exact same experiment as described above using their index finger.

The results are very similar to those observed when users interact solely with their thumb, with even higher accuracy (98% on average, Figure 10-left), higher speed (0.27 s on average, Figure 10-center) and lower overall perceived difficulty (Figure 10-right).

	Recognition rate (%)				Mean duration (s)				Perceived difficulty (1-5)			
	1	2	3	4	1	2	3	4	1	2	3	4
A	98 ±2	99 ±1	99 ±1	99 ±1	0.27 ±0.03	0.27 ±0.04	0.27 ±0.04	0.27 ±0.05	1.33 ±0.52	1.33 ±0.52	1.33 ±0.52	1.56 ±0.50
B	98 ±3	99 ±1	100 ±1	99 ±3	0.26 ±0.04	0.27 ±0.03	0.27 ±0.04	0.27 ±0.04	1.36 ±0.50	1.17 ±0.41	1.17 ±0.41	1.67 ±0.52
C	98 ±1	98 ±1	99 ±2	99 ±2	0.27 ±0.04	0.28 ±0.05	0.29 ±0.05	0.28 ±0.05	1.39 ±0.49	1.17 ±0.41	1.17 ±0.41	1.75 ±0.42
D	96 ±2	98 ±1	92 ±7		0.27 ±0.05	0.27 ±0.05	0.29 ±0.06		3.03 ±0.83	3.31 ±0.83	3.72 ±0.98	

Figure 10: Mean values and standard deviation per target location when using the index finger in two-handed use.

4.5.4 Summary

Our experimental data shows that our algorithm reliably recognizes ThumbRocks and efficiently distinguishes them from other events (96-98% overall accuracy). Furthermore, they are performed quickly (0.34 s in the worst case) and their pointing accuracy is at least as good as that of a tap. Finally, the incremental nature of the recognizer captures the finger rolling trace with high fidelity and in real time, therefore supporting immediate and continuous feedback.

5 APPLICATIONS

We developed several prototypes, illustrated in the accompanying video, to demonstrate a variety of applications of the ThumbRock.

5.1 Finger rolling as a discrete event

Tap alternative A ThumbRock can act as a complement to a tap to avoid resorting to additional widgets. For example, there is currently no way to select items in a hierarchal list or collection of thumbnails without using an additional "edit" button to enter a mode where taps select the items instead of navigating to them. For compatibility with the default mode, we use taps for navigation and ThumbRocks for selection. ThumbRocks can also be used to invoke frequent commands. For example, they can be used to drop a pin on a map instead of using a time-consuming dwell.

A ThumbRock can also replace a tap event for "dangerous" buttons. For example, the "Send" button is often very close to alphanumerical keys on a soft keyboard so that the user may accidentally hit it while typing a message. To avoid such errors, our "Send" button can only be invoked with a ThumbRock. An interesting variant is to automatically turn "tap buttons" into "ThumbRock buttons" when the tilt value of the device indicates a situation where continuous contact of the thumb with the surface is required to maintain a good grip, such as when the user is lying down.

To show that a widget such as a button or tab can receive ThumbRock events, we propose to depict it as a mechanical switch (Figure 11). A more general approach is to provide feedforward when the user hesitates, as with marking menus [11]: When the thumb is in contact with the surface at a location where a ThumbRock is meaningful, a small mechanical switch appears close to the thumb tip after a short delay. The appropriate location for displaying this visual clue can be computed using the blob size and orientation.

In-drag event Some techniques use the dragging quasi-mode, e.g. the magnifying glass for positioning the text caret or the *Bezel Swipe* for applying a tool to an object. As mentioned in the introduction, these techniques exit the quasi-mode when lifting the finger, making them cumbersome to use several times in a row. ThumbRocks avoids this repetition: in the text selection example, the user switches between the caret positioning and selection modes with a ThumbRock, without loosing the cursor position (Figure 12-a). After a *Bezel Swipe*, the activated tool can be applied to multiple objects in a row, each selected with a ThumbRock.

Since the thumb can remain on the screen right after a ThumbRock, it can be used to grab an object and then drag-and-drop it, or to invoke and then navigate a hierarchical menu such as *SAM* [3]. During a drag-and-drop, a ThumbRock can also be used for, e.g., switching between views. For example, to move an application icon to an off-screen location, a ThumbRock invokes an "Exposé" view of the different home screens (Figure 12-b) to let users select the

Figure 11: ThumbRock widgets look like mechanical switches that follow the thumb's movement.

(a) Text selection (b) Exposé switch (c) Zoom

Figure 12: Application prototypes using the ThumbRock. Top row shows the ThumbRock being performed and bottom row the subsequent action.

destination screen with another ThumbRock or by releasing the finger. This avoids the slow and error-prone auto-scrolling mechanism for browsing multiple home screens sequentially.

A ThumbRock can also be used right before or right after a gestural command : it can act as a start delimiter so that subsequent drag events are interpreted as a gestural command, e.g. to activate a GestureSearch [12] query. It can also be used at the end of a gestural command as a modifier or a way to repeat the command. For example, we use a ThumbRock at the end of a swipe gesture to navigate to the next or previous page.

5.2 Finger rolling as a continuous event

The ThumbRock trace captured by our algorithm can also be used to retrieve additional information. A ThumbRock defines a segment (P_1P_2 or P_2P_3) whose *orientation* and *length* encode information. In particular, the orientation provides information about the hand posture and can be used to reduce occlusion. For example, if a ThumbRock is used to invoke a marking menu, we offset the location of the menu to ensure its full visibility.

We also investigated using the ThumbRock segment as a control space for setting a continuous value. We designed a **MicroSlider** that lets users select a value by rolling their thumb (or index) over an interval that is mapped onto segment P_2P_3. While the thumb is down, after performing a ThumbRock, P_2 and P_3 are mapped to the bounds of the interval. We chose P_3 instead of P_1 because the activation takes place once P_3 is identified, ensuring that the slider starts at 0. Physiologically, users can internalize the bounds as the two extreme positions of the thumb (tip *up* and tip *down*) and can use kinesthetic and visual feedback to control the value of the slider by rolling their thumb between these two positions.

To stop using the MicroSlider, users simply release their thumb. However, we cannot simply consider the slider value at release time because users tends to perform unintentional drags when lifting their finger. This is especially true when the thumb is completely rolled down: upon release, the thumb tends to roll up, moving the

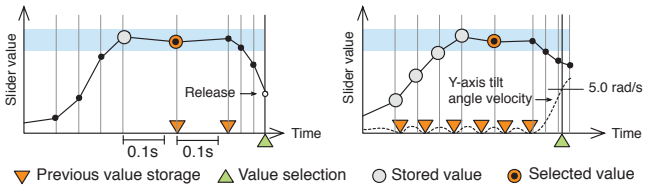

Figure 13: QuickRelease (left) and QuickTilt (right) selection techniques. The light blue band shows the target range to select.

cursor by at least half the slider range. To solve this, we designed two selection mechanisms:

- QUICKRELEASE stores the slider value when two successive drag events are more than 0.1s apart, indicating a slight pause. The most recent such value is selected when the up event occurs (Figure 13-left). This filters out accidental drags that occur when lifting the thumb, which typically create drag events separated by less than 0.02s.
- QUICKTILT uses the absolute angular velocity around the Y-axis provided by the gyroscope. Each time it starts increasing, the current slider value is stored. The most recent such value is selected when a tilt faster than 5 rad/s occurs (Figure 13-right). With this technique, the user can select a value without disabling the slider since the thumb can stay in contact with the surface.

We have used the MicroSlider to navigate a document and to control the scale of a map (Figure 12-c). The incremental nature of the ThumbRock recognizer makes this micro gesture usable within a drag interaction, creating even more possibilities. For example, combined with the QUICKTILT selection mechanism, the user can easily interleave pan and zoom actions without lifting the finger. In order to inform the design of techniques using such a continuous control, we conducted an experiment to assess the precision and usability of the MicroSlider on an item selection task.

5.3 MicroSlider Experiment

Our experimental task is inspired by Ramos et al.'s study of users' precision when controlling pressure via a pen [18]. When a trial begins, a target (a dark-red screen-wide rectangle) appears. The participant then performs a ThumbRock anywhere on the screen. This creates a MicroSlider that lets the participant control the vertical position of a cursor (a thin black horizontal line) on the screen. The participant is instructed to move the cursor into the target area as fast and as accurately as possible. While the cursor is outside the target, the area containing the cursor is light red. It turns white once the cursor enters the target (Figure 14-left). The ThumbRock visual and audio feedbacks are also enabled.

To assess the limits of the MicroSlider in terms of precision and usability, we tested several combinations of target widths and distances and considered the two selection mechanisms:

- *Width* is expressed as a percentage of screen height: $\{10\%, 5\%, 2.5\%, 1.25\%, 0.625\%\}$;
- *Distance* from the top is expressed as a percentage of screen height: $\{20\%, 40\%, 60\%, 80\%\}$;
- *Selection* $\in$ {QUICKRELEASE, QUICKTILT}.

Six unpaid participants, 23 to 29 years old (average 26.0, median 26.0), participated in the study. One was left-handed. The apparatus was the same as the one used in previous experiments, except that we used a 4^{th} generation Apple iPod Touch running iOS 5.1.1, which has a higher touch precision (326 PPI).

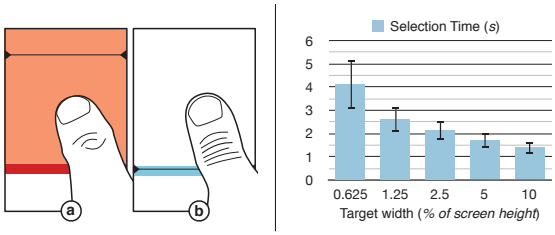

Figure 14: Left: Feedback of the experimental task while off target (a) and on target (b). Right: Selection time by target width.

Participants start with a training session (about 10 minutes) where they test both selection techniques for the four *Distance* values with *Width*=2.5%. The rest of the experiment is divided among the two *selection technique* conditions, QUICKRELEASE and QUICKTILT, counterbalanced across participants with a Latin square. With each selection technique, participants perform five blocks of 20 trials, one for each *Distance* $\times$ *Width* combination in random order. The first block is a practice block, the remaining four are testing blocks, resulting in 200 trials per participant. Once the experiment is over, participants tell which selection technique they found most comfortable and which one they felt was faster.

Analysis of variance of *Distance* $\times$ *Width* $\times$ *Selection* on selection time (*ST*), i.e. the time between the end of the ThumbRock gesture and the selection, revealed only one significant effect, *Width* ($F_{4,20} = 65$, $p < 0.0001$), illustrated on the right part of Figure 14. *ST* increases linearly from *Width* $= 10\%$ to *Width* $= 1.25\%$ (around +20% each time the target width is divided by two) but then increases sharply between *Width* $= 1.25\%$ and *Width* $= 0.625\%$ (about +57%). This means that the precision limit of the MicroSlider technique lies between these two target sizes. As a design guideline, we recommend using the MicroSlider when the range contains fewer than a hundred values.

Regarding the selection mechanisms, the difference between QUICKRELEASE and QUICKTILT is not significant ($p = 0.2$) and participants did not express a clear preference for either one. However, participants were a bit faster with QUICKRELEASE than with QUICKTILT (*ST* $= 2.3$ s vs. 2.5 s). This may be due to the cognitive cost of combining two modalities (touch and tilt) and to the fact that releasing the thumb is probably a more intuitive selection action for touch-based interfaces. However, QUICKTILT remains interesting for its ability to chain several selections.

6 CONCLUSION

The ThumbRock is an in-drag gesture that consists of rolling the thumb back and forth on a touch surface. With our incremental algorithm, finger rolling becomes a ready-to-use event that is recognized with over 96% accuracy and no per-user calibration. Furthermore, recognition starts early to enable real-time feedback and avoid conflicts with default drag actions. While our recognizer was designed for one-handed use with the thumb, it is as reliable in two-handed use with the index finger. Our prototypes demonstrate that ThumbRocks can improve many applications with fluid interactions that combine selection and navigation. Finally, the trace of a ThumbRock gesture can be used to design richer interactions, such as a MicroSlider to select a continuous value by rolling the thumb.

We consider two areas for future work. First, the ThumbRock recognizer could be even more robust by checking whether the touch trajectory is straight enough or whether the blob orientation is consistent with the movement. The constraints could also be relaxed according to the screen location. Second, we want to extract other information from a ThumbRock (e.g. roll up speed or dwell time before rolling up) to create several discrete events in the spirit of long vs. short mouse clicks.

ACKNOWLEDGEMENTS

We thank Mathieu Nancel and Clément Pillias for many useful discussions. We also thank the study participants for their time.

REFERENCES

[1] P. Agar and K. Novins. Polygon recognition in sketch-based interfaces with immediate and continuous feedback. In *Proc. GRAPHITE '03*, 147–150. ACM, 2003.

[2] H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *Proc. CHI '06*, 1263–1272, 2006.

[3] D. Bonnet and C. Appert. Sam: the swiss army menu. In *Proc. IHM '11*, 5:1–5:4. ACM, 2011.

[4] S. Boring, D. Ledo, X. A. Chen, N. Marquardt, A. Tang, and S. Greenberg. The fat thumb: using the thumb's contact size for single-handed mobile interaction. In *Proc. MobileHCI '12*, 207–208. ACM, 2012.

[5] C. Harrison, J. Schwarz, and S. Hudson. TapSense: enhancing finger interaction on touch surfaces. In *Proc. UIST '11*, 627–636, 2011.

[6] S. Heo and G. Lee. Force gestures: Augmenting touch screen gestures with normal and tangential forces. In *Proc. UIST '11*, 621–626. ACM, 2011.

[7] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proc. CHI '05*, 451–460. ACM, 2005.

[8] K. Hinckley and H. Song. Sensor synaesthesia: Touch in motion, and motion in touch. In *Proc. CHI '11*, 801–810. ACM, 2011.

[9] C. Holz and P. Baudisch. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proc. CHI '10*, 581–590. ACM, 2010.

[10] A. K. Karlson and B. B. Bederson. One-handed touchscreen input for legacy applications. In *Proc. CHI '08*, 1399–1408. ACM, 2008.

[11] G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Proc. CHI '94*, 258–264. ACM, 1994.

[12] Y. Li. Gesture search: a tool for fast mobile data access. In *Proc. UIST '10*, 87–96. ACM, 2010.

[13] I. S. MacKenzie and A. Oniszczak. A comparison of three selection techniques for touchpads. In *Proc. CHI '98*, 336–343. ACM, 1998.

[14] S. Malacria, E. Lecolinet, and Y. Guiard. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces: the cyclostar approach. In *Proc. CHI '10*, 2615–2624, 2010.

[15] A. Olwal, S. Feiner, and S. Heyman. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *Proc. CHI '08*, 295–304, 2008.

[16] K. B. Perry and J. P. Hourcade. Evaluating one handed thumb tapping on mobile touchscreen devices. In *Proc. GI '08*, 57–64. Canadian Information Processing Soc., 2008.

[17] M. Rahman, S. Gustafson, P. Irani, and S. Subramanian. Tilt techniques: investigating the dexterity of wrist-based input. In *Proc. CHI '09*, 1943–1952. ACM, 2009. ACM ID: 1518997.

[18] G. Ramos, M. Boulos, and R. Balakrishnan. Pressure widgets. In *Proc. CHI '04*, 487–494. ACM, 2004.

[19] V. Roth and T. Turner. Bezel swipe: Conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proc. CHI '09*, 1523–1526, 2009.

[20] A. Roudaut, E. Lecolinet, and Y. Guiard. MicroRolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proc. CHI '09*, 927–936, 2009.

[21] J. Ruiz and Y. Li. DoubleFlip: a motion gesture delimiter for mobile interaction. In *Proc. CHI '11*, 2717–2720, 2011.

[22] J. Schwarz, J. Mankoff, and S. Hudson. Monte carlo methods for managing interactive state, action and feedback under uncertainty. In *Proc. UIST '11*, 235–244. ACM, 2011.

[23] A. J. Sellen, G. P. Kurtenbach, and W. A. S. Buxton. The prevention of mode errors through sensory feedback. *Hum.-Comput. Interact.*, 7(2):141–164, June 1992.

[24] R. Vatavu, L. Grisoni, and S. Pentiuc. Multiscale detection of gesture patterns in continuous motion trajectories. In *Gesture in Embodied Communication and HCI, LNCS 5934*, 85–97. Springer, 2010.

[25] F. Wang, X. Cao, X. Ren, and P. Irani. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proc. UIST '09*, 23–32. ACM, 2009.