MicroRolls: Expanding Touch-Screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb

Anne Roudaut[1,2] anne.roudaut@enst.fr
Eric Lecolinet[1] eric.lecolinet@enst.fr
Yves Guiard[1] yves.guiard@enst.fr

[1]TELECOM ParisTech – CNRS LTCI
46 rue Barrault
75013, Paris, France

[2]Alcatel-Lucent Bell Labs France
Centre de Villarceaux, Route de Villejust
91620, Nozay, France

ABSTRACT
The input vocabulary for touch-screen interaction on handhelds is dramatically limited, especially when the thumb must be used. To enrich that vocabulary we propose to discriminate, among thumb gestures, those we call MicroRolls, characterized by zero tangential velocity of the skin relative to the screen surface. Combining four categories of thumb gestures, Drags, Swipes, Rubbings and MicroRolls, with other classification dimensions, we show that at least 16 elemental gestures can be automatically recognized. We also report the results of two experiments showing that the roll vs. slide distinction facilitates thumb input in a realistic copy and paste task, relative to existing interaction techniques.

Author Keywords
Mobile devices, touch-screen, interaction, selection techniques, gestures, one-handed, thumb interaction, rolling/sliding gestures, MicroRoll, RollTap, RollMark.

ACM Classification Keywords
H.5.2. User Interfaces: Input Devices and Strategies, Interaction Styles, Screen Design; D.2.2 User Interfaces

INTRODUCTION

The reduction in mass and size that has taken place in the transition from the old desktop computer to the laptop, along with the extension of communication networks, has greatly increased the mobility of computer utilization. Today one may read one's mail virtually anywhere. That change has been possible at essentially no cost from the viewpoint of computer usability: leaving aside the difference between a mouse and a touchpad, everything has been miniaturized but the interface. A modern laptop looks like a normal-sized keyboard attached to a normal-sized screen, with all the rest occupying virtually no room.

With the next qualitative leap, from the laptop to the handheld, computer users no longer need to be seated. This new step in the direction of increased mobility has involved a lot more miniaturization. But with handheld devices some critical scale threshold has been passed as now interface components *are* miniaturized, raising an unprecedented challenge to interaction design.

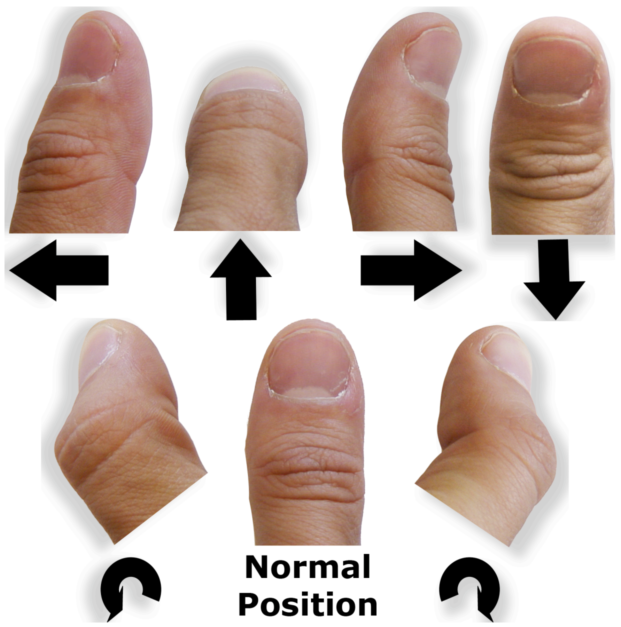

Fig. 1. The six MicroRoll gestures used in this study.

In comparison with the interface of a standard laptop, those of handheld terminals like smartphones or PDAs are dramatically impoverished. Most familiar input devices have been removed because, if scaled down to such an extent, they would no longer match the size of the human hand. The devices that are most noticeably missing are the keyboard and the mouse, or any equivalent contrivance that would deliver the functionalities of the two buttons and the wheel of the standard mouse.

Combined with the very limited amount of screen real estate, these limitations result in a dramatic reduction of interaction bandwidth. We focus here on the fairly extreme case of handheld devices that must be operated with just one thumb [11,17], a restriction that arises quite naturally in a variety of real world situations (e.g. the user is standing in the metro, and one hand is busy full time to ensure upright stance). In such a case not only must the input rely on a single hand, but the hand that provides the input must also hold the device. Given the anatomy of the human hand, in such a case only the thumb can be used to cover the surface area of the screen.

HANDHELD INPUT LIMITATIONS, RELATED WORK

Handheld design is all about compromising in the face of a shrunk and impoverished interface. Smartphones with an extended touch-screen are suitable for the display of images or for Web browsing, but this is at the expense of the physical keyboard. Tricky interaction design problems result from combinations of the following factors: the absence of a keyboard or physical buttons, the intrinsic limitations of passive touch-screens, and the mediocre precision of finger operation on touch-screens.

Hotkeys and Modifiers

The absence of a keyboard deprives the user, by the same token, of hotkeys and modifiers. Although computer users usually know a small number of hotkeys, they use them often [7]. Recourse to hotkeys for the copy, cut, and paste functions is an especially critical instance, considering their high frequency of use. It is noteworthy that nothing equivalent to that crucial resource exists on a sophisticated device like the iPhone.

Interaction States

As often pointed out [3,4,13,14], graphical user interfaces require several different interaction states. The interaction model actually varies with the input technology [4], but virtually all modern computers make it possible for users to specify whether they want to move the cursor, drag an object, activate an interactor, or open a context menu and select one of its items. For instance, with the mouse or touchpad buttons (possibly in combination with modifier keys) one may discriminate *tracking*, *activating/dragging*, and '*menuing'*. Active touch-screens provide similar capabilities, thanks to stylus buttons and a technology that detects whether the stylus is close to or actually touching the screen, plus the possibility of measuring pressure. Unfortunately, common capacitive and resistive touch-screens used in mobile devices do no provide equivalent capabilities. The user cannot specify the desired state, a drawback that results in several ambiguities and limitations, listed below.

Context and Marking Menus

For lack of some equivalent for a right-mouse button, menu opening generally relies on a temporal delay. Not only does this degrade performance when using linear menus, it may also seriously hinder interaction techniques, like Marking Menus [12], that rely on an expert mode based on gestures. If one has no control over quasi-modes to specify the meaning of one's gestures, there is no way to state whether the gesture should affect the selected object (e.g. initiating a drag & drop) or trigger a contextual Marking menu. Waiting for a delay before drawing gestures would make little sense.

Precise Positioning

The absence of a *tracking* state not only prohibits interaction techniques based on hovering, such as tooltips, but also constrains users to point directly at the correct screen location without the possibility of finely adjusting the cursor position before actual selection. This problem is particularly unwelcome on handheld devices on which, due to absolute size problems, text and target selection is especially difficult, all the more so when interacting with a thumb.

Efficient techniques for finger pointing have been proposed [9,19,21,24,25] but how to allow the user to activate them is a difficult design problem. Having them permanently available would not only conflict with standard interaction styles but also degrade efficiency when dealing with large enough targets. In an attempt to solve this problem, Thumbspace [9] is activated after pressing a physical button, while Shift [24] relies on temporal delays, taking into account the properties of target objects (the larger the object the longer the delay before Shift is activated). A simpler, and presumably more efficient option would be to let users decide which pointing mode they want to elicit when pressing the screen to select some graphical object. Again, interaction states could allow this, were they available on passive touch-screens.

Avoiding Widgets

Another approach for avoiding precision problems, also liable to save screen real estate, consists in replacing on-screen widgets with alternate interaction techniques. For example, viewports do not have scrollbars on the iPhone, the scrolling being obtained by direct finger dragging. One drawback is that the user can no longer drag an object contained in a viewport or scroll an embedded viewport. This limitation is especially problematic for Web pages which demand scrolling but also often contain graspable or scrollable widgets (e.g. Google Maps). One solution is to take into account the starting point of the drag by checking whether the finger was initially pressed on some object or an empty area. But then precision becomes an issue, especially for finger interaction. As in the previous case, interaction states are needed to allow users to specify which action they actually want to perform.

Gestures as Substitute

Gestures conceal rich resources that can be used to solve many of the above problems. For instance Lift-and-Tap gestures [13] can serve as a substitute for clicking on a touch-screen. A small rocking motion of the finger is used in SimPress [3] to differentiate a tracking vs. a dragging state. This technique, developed for vision-tracked tabletops, simulates a pressure-sensitive device by analyzing the finger contact area on the screen. In a different context, Rollpad [15] was proposed as an alternative to Multitap for inputting characters onto a 12-key soft keypad. Instead of tapping a key several times, the user presses the key and then performs a rolling motion of the finger to select one individual character. Gestures also provide an efficient means for selecting commands, as with Marking Menus [12] and derived techniques [2], or for improving navigation [10].

Gestures can also be used to set continuous values as in Flow menus [8] and Control menus [18]. Rubbing techniques such as Rub-Pointing [14] have been recently proposed to allow view zooming with a single hand. Rubbing, which relies on diagonal to-and-fro movements of the finger, can be seen as a natural substitute for the mouse wheel on touch-screens. Of course, multi-touch gestures provide other ways of achieving such operations but that resource is out of reach when a single hand has to both hold and operate the device.

Analyzing the geometrical path or the space-time kinematics of gestures can also provide valuable information. For example Curve Dial [23], which focuses on the curvature of user motion, was proposed for eyes-free scrolling through documents. Swipe or Flick gestures [6], easy to recognize automatically thanks to their specific acceleration pattern, are now used on a number of commercial systems, including the iPhone, where they serve to control scrolling.

Thus gestural interaction appears to be one of the most promising approaches for offsetting the input limitations of passive touch-screens, especially in the case of small handheld devices. In the next sections, we introduce a coherent set of gestures that seems well suited for thumb interaction. We first explain why one classification criterion, corresponding to the slide vs. roll distinction, seems particularly robust and easy to leverage for users as well as recognition algorithms. We will then turn to experiments that give support to this statement and some possible applications.

SLIDES VS. ROLLS: DISTINGUISHING TWO MOTION REGIMES TO HELP DISAMBIGUATE THE INPUT

As mentioned above, some interaction techniques have incorporated the roll more or less incidentally [3, 15]. To our knowledge, however, no principled justification of the distinction between slide and roll gestures has been offered so far. The distinction in question, which is grounded in the physics of friction, probably enjoys a broader scope of application than one might believe at first.

Whenever the contact zone of the user's fingertip shifts on a touch-screen, a stream of events is registered by the system and a *move* is identified. But according to elementary physics [e.g., 20], two categories of moves, the *slide* and the *roll* of the fingertip, should be easily distinguishable. When a finger touches a solid surface, the contact forces may be decomposed into two perpendicular forces, N and T, whose orientations are normal and tangential, respectively, to the mutual surface. When the two contacting surfaces move relative to each other, the tangential sliding force is opposed by *kinetic friction* $T_k = \mu_k * N$, where μ_k is the characteristic coefficient of kinetic friction of the combination of materials under consideration. Now suppose the system is at rest and the tangential effort is gradually increased: so long as the tangential effort is insufficient to trigger motion, the blocking force is known as *static friction* $T_s \leq \mu_s * N$, where μ_s is the coefficient of static friction. The reason why it usually takes a greater force to start the motion than to keep it going is because typically $\mu_s > \mu_k$.

The fingertip has a spheroid shape, and so its motion on a touch-screen may be likened to that of a ball on a flat surface. A ball will either *slide* or *roll* depending on whether or not the horizontal component of the contacting forces overcomes static friction [5]. The fingertip must have just the same two possible regimes of motion on a touch-screen. Finger motion being detected on the touch-screen, if the skin moves relative to the screen surface, then the fingertip is sliding, otherwise it is rolling.

Physical transducers placed underneath the screen would probably allow errorless automatic discrimination of finger rolls from slides, but it turns out that the two categories of moves are actually easy to discriminate in a purely kinematic approach (Fig. 2). As a matter of fact, the recognition algorithm we used in Exp. 1, to be described below, was able, after little training, to perform the binary classification with virtually no errors.

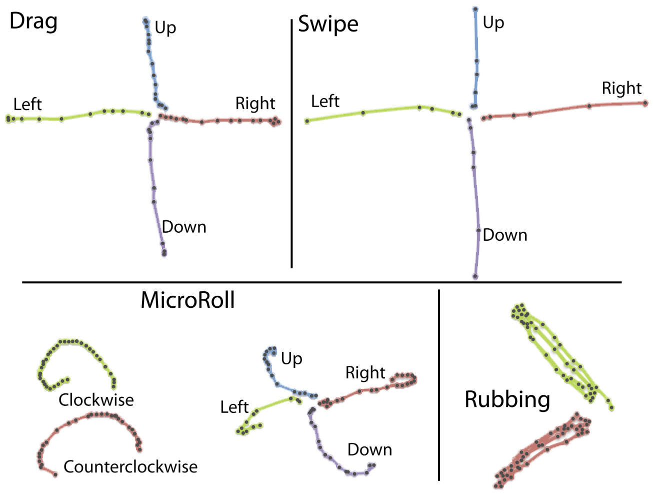

Fig. 2. Typical instances of the kinematic traces produced on the touch-screen by the 16 elemental gestures of our repertoire. Shown are the successive positions of the fingertip barycenter ($\Delta t \approx 25$ ms).

Note that finger rolls, like finger slides, may be modulated in direction and, to some extent, in amplitude. Obviously the range of amplitudes that can be covered with a roll of the thumb is limited, hence the label *MicroRoll*. Note also that a MicroRoll, just like a slide, can serve to trace curves. Finally, for users it is quite a different experience to produce a roll (where motion of the hand is supported by a stationary fingertip) and a slide (where it is the fingertip that moves), meaning that the latter cannot be confused with the former. In sum we do seem to have at our disposal a reliable and convenient extra bit of information for the design of gestural vocabularies.

APPLICATION OF THE DISTINCTION: A SIXTEEN-GESTURE REPERTOIRE FOR THE TOUCH SCREEN

Leveraging the above facts we designed a set of 16 elemental gestures (Fig. 3) that fall in two main categories:

- Six *MicroRoll* gestures (Fig. 1): four straight gestures (in all cardinal directions) and two circular gestures (clockwise and counterclockwise);
- Ten Slide gestures: four *Drag* and four *Swipe* gestures (in all cardinal directions) plus two diagonal *Rubbing* gestures, small repetitive diagonal motions as in [14].

We then developed an algorithm to automatically recognize that 16-gesture set. This gesture recognizer uses a simple but efficient supervised-learning algorithm based on K-nearest neighbors, in which the gestures are characterized by a set of 10 features from Rubine's algorithm [22].

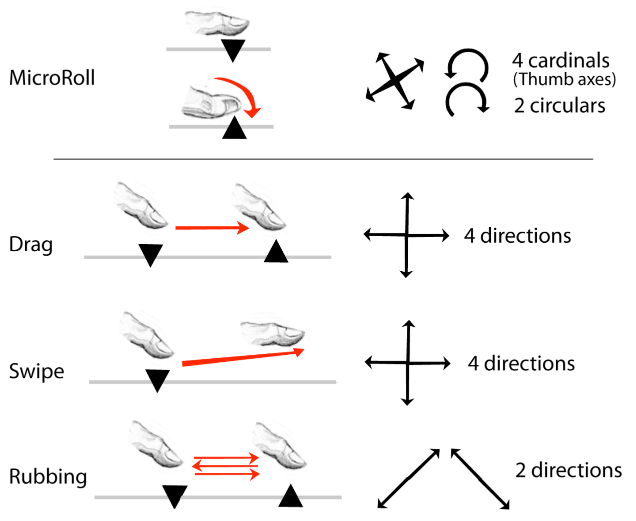

Fig. 3. The 16 elemental gestures of our repertoire viewed from the side and from above.

EXPERIMENT 1

Our goal was to check whether Drags, Swipes, Rubbings and Rolls could be actually recognized. To this end, we simply asked a sample of participants to draw all 16 elementary gestures of Fig. 3 at different locations on a small handheld screen. Half of the gesture database (odd-numbered trials) was used for training the recognition algorithm, and the other half for evaluating its accuracy.

Methods

Task. The participants were asked to draw the gestures, while seated, with the thumb of their preferred hand, that hand also serving to hold the device. Each trial started with a printed message specifying the requested movement, followed by the appearance of a circle 60 pixels in diameter indicating where to make the gesture on the screen. Each of the 16 gestures was performed four times at nine screen locations. These locations were determined by dividing the screen into nine 80x106 pixel areas, with the location mark appearing in the center of the area. The trial ended when the thumb was lifted from the screen.

Apparatus. All software was developed in C# using the .Net Compact Framework. This experiment and the next two were performed on a HTC P3600 running Windows Mobile 5.0 with a QVGA 320x240 pixel resistive touch-screen.

Participants. Ten right-handed volunteers (3 female) aged 23-28, none of whom were experienced with touch-screen handhelds, were recruited from our institution and received a handful of candies for their participation.

Design. Participants made each gesture four times in each area, yielding a total of 16 x 9 x 4 = 576 gestures per participant. Latin squares were used to counterbalance order within participant. For each gesture, participants performed four trials in each area. The experiment was run in a single session, which lasted about 40 minutes, the first 10 minutes being dedicated to instructions and warm up.

Results

Recognition Rates. The overall recognition rate was 95.3%, despite the fact that some of the gestures we asked were rather hard to make in some locations. A Drag or a Swipe to the right is unlikely to be spontaneously started close to the right border, because that would constrain gesture amplitude. In some systems, a Slide can serve to drag an object outside the viewport, in which case it can be performed ins the direction of a border and started close to that border. Such a gesture, however, has a different kinematic signature, involving a delay (once the border is reached, the user must keep the finger pressed to scroll the viewport). A more elaborate algorithm could certainly detect these specific gestures, but they were out of the scope of the current study.

	Drag	MicroRoll	Swipe	Rubbing
Drag	**95,1**	3,5	1,4	0
MicroRoll	3,9	**96,1**	0	0
Swipe	2,6	0	**97,4**	0
Rubbing	0	0	0	**100**

Table 1. Recognition rates with unlikely gestures discarded.

Leaving aside the unlikely Drag and Swipe gestures that started near a boundary and were directed toward it, the average recognition rate of our algorithm was raised to 97.1% (Table 1).

Note that that result was obtained without any feedback to the participants about the possible ambiguity of their gestures (which provided the training material for the recognition algorithm). Real-life users, with feedback, would probably adapt their gestures to maximize recognition performance. Moreover, the recognition algorithm can easily be made to adapt to each individual user, just by adding new samples. This can be done automatically, in an incremental way, without the user even knowing.

Time. The durations of the 16 gestures were measured from the first screen contact to the thumb lift. The Swipe, unsurprisingly, was the fastest gesture (129ms). On average over the four directions it took participants 230ms to complete a cardinal MicroRoll, and it took them 339ms to complete a circular MicroRoll. Note that these durations are appreciably shorter than the 938ms and 458ms we measured for Rubbing and Drag.

IMPLEMENTING OUR NEW GESTURAL VOCABULARY

Exp. 1 showed that Drag, Swipe, Rubbing and MicroRoll gestures can be reliably distinguished on handheld touch-screens. As shown by the experiments to be reported, they are also easy to discriminate from standard "Tap" gestures, which do not involve motion on the screen. Rubbing and MicroRoll gestures can thus be used to augment standard interaction styles without conflict. As stated in previous sections, the absence of hotkeys, modifiers and mouse buttons seriously hinders interaction on mobile devices. Rubbings and MicroRolls can serve as substitutes for these missing interaction resources. While Rubbing [14] looks like a natural substitute for mouse wheeling (for zooming or scrolling), MicroRoll gestures can be used in a variety of ways. Let us introduce one intuitively appealing way of using them for performing useful and frequently used functions (Fig. 4b):

1. The *Bottom*, *Left* and *Right* MicroRolls replace the familiar cut, paste and copy hotkeys. Although crucial for many applications, this functionality is currently unavailable on many mobile platforms, including the iPhone.
2. The *Top* MicroRoll activates a precision mode. Targets on handhelds are often too small for selection with a fingertip.
3. The *Clockwise* MicroRoll gesture replaces the right mouse button for opening context menus. This option makes it possible to activate Marking menus in the expert mode, or likewise to open context menus with no delay.
4. The *Counterclockwise* MicroRoll gesture triggers a quasi-mode that will control the effect of a subsequent Drag (e.g., specifying that a drag-and-drop, rather than a view scroll, is required).

Learnability and the RollMark Menu Concept

One problem that can affect the learnability of gesture-based interfaces is that the commands are invisible. Marking menus solve this problem by displaying the commands if the user keeps the mouse pressed for a short delay. The same principle can be applied to MicroRoll gestures. As shown in Fig. 4a, a *RollMark* menu recalling which MicroRolls trigger which commands would appear only if the finger was kept immobile on the screen for 300ms. As for traditional Marking menus, MicroRoll gestures may remain exactly the same in novice and expert modes, the only thing being that a novice, but not an expert, will wait for the RollMark menu display.

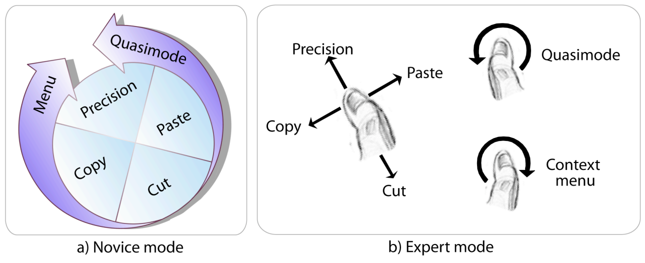

Fig. 4. The RollMark Menu.

Such a solution precludes recourse to temporal delays for opening context menus but this is not a major concern in an implementation like ours, which uses another (clockwise) MicroRoll gesture to control the opening of context menus. We mentioned that a MicroRoll takes about the same time as a usual menu-opening timeout, on the order of 300ms (in fact, it should be realized that timeouts often last more than their programmed duration, being restarted by any inadvertent twitch of the finger). In sum, delays should probably be reserved for novice usage, with experienced users allowed to avoid them.

While MicroRoll gestures could be used in many sophisticated ways, we found it important, as a first step, to evaluate their efficiency in a simple realistic case. The two experiments we performed compared MicroRolls with two conventional ways of performing copy and paste sequences on mobile devices.

EXPERIMENT 2

The aim of Exp. 2 was to evaluate the efficiency of MicroRoll (MR) gestures relative to two conventional techniques, the toolbar (TB) and the context menu (CM), which represent two sensible but different compromises in the face of space scarcity. A TB, whose main drawback is its permanent consumption of screen real estate, offers directly clickable buttons and thus should allow fast interaction. The CM seems better suited for small screens as it pops up only when needed, hence presumably its high incidence in the Windows Mobile environment, but it has the drawback of imposing an activation timeout delay on users, which can only slow down the interaction.

Experimental Task

We used a fairly realistic copy and paste task with the constraint that the pasting area was located away from the viewport, thus requiring the participant to perform a *pan* between the initial copy and the final paste (Fig. 5a). Participants had to perform five consecutive operations: (1) select the object to be copied (materialized by a red circular target 60 pixels in diameter appearing at screen center); (2) activate the copy command; (3) pan the view to reveal the pasting zone, also a 60 pixels round target, but green in color; (4) select that second target; and (5) activate the paste command. Any of these five operations had to be repeated if missed (for instance in the case of an empty or wrong selection). Thus, some copy and paste sequences involved more than five operations, but all five of the above list, by construction, had to be successfully performed once. Thanks to this feature, task completion time will be considered below a safe net measure of performance.

Panning was done by means of Drag gestures. The location of the pasting area was invariably in the East direction, at a constant horizontal distance of 240 pixels from screen center (Fig. 5a). Depending on the technique condition, the copy and paste commands were activated by tapping on one of four buttons of the TB, by activating one of four items of the CM, or by making one among four possible cardinal *MR* gestures. Thus in all three conditions, the participants were offered four possible options, only two of which had to be used.

To activate the CM, participants had to press the (copy or paste) target, wait 300ms, and then select the appropriate button by sliding their thumb to the appropriate location and lifting it up. While commonly used in menu systems, this pointing strategy differs from that of Windows Mobile, which requires users to first open the menu, then lift the finger and finally tap on the appropriate item. Thus our implementation of the CM technique required one less pointing act. In the MR condition no visual indication, such as icons or text labels, was provided to participants, who had learned the simple rule that left = copy and right = paste.

We were aware that the efficiency of our two reference techniques depended on several parameters. Especially critical was the size of the graphical buttons, whether tool icons or menu items. We resolved to consider two button heights that correspond to two established standards: 20 pixels (about 3 mm on the test device) is commonly used for buttons in Windows Mobile applications; 60 pixels (about 9 mm) approximates the size of icons in the iPhone main window and corresponds to Parhi et al.'s recommendations for thumb operation on handhelds [16].

Button width was chosen accordingly: a typical toolbar button being square, so were ours, with the copy and paste buttons located at the left-most and second left-most locations respectively (Fig. 5b). Menu items had to be rectangular because of their text material; we chose a convenient aspect ratio of 3.

Apparatus and Participants

We used the same equipment as in Exp. 1. Twelve right-handers volunteers from our institution (aged 23-31, two female) participated.

Experimental Design

This experiment involved a 3 x 2 within-participant design, the factors being the technique (TB, CM, MR) and the level of difficulty for button acquisition (easy, 60px-high buttons vs. difficult, 20px-high buttons). The order of presentation of the conditions was counterbalanced within participants with Latin squares. For each technique, participants performed the easy task first so as to familiarize themselves with the technique (first block of trials) before handling the more difficult condition (second block). Eight trials were performed for each difficulty level. The button size factor is obviously irrelevant to the MR technique, which does not rely on any button. Therefore the MR technique was tested in the same conditions in two consecutive blocks, yielding the same total number of copy and paste sequences as for the other two techniques. In total, the experiment involved 12 participants × 3 techniques (TB, CM, MR) × 2 levels of task difficulty × 8 trials = 576 copy and paste sequences. We performed the same 3 *Technique* x 2 *task difficulty* within-subject analysis of variance (ANOVA) on several dependent variables.

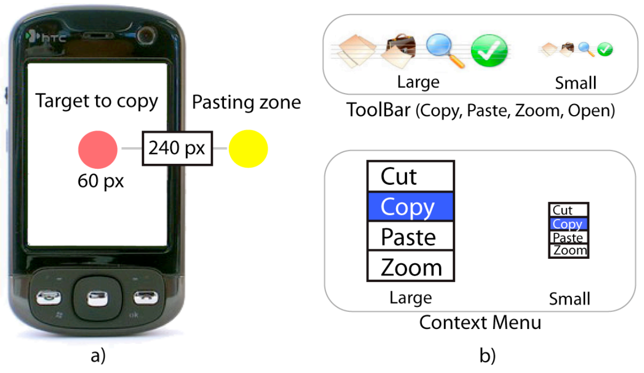

Fig. 5. (a) Layout of the copy and paste task; (b) The TB and CM displays.

Results

Task completion time (Fig. 6) was measured from the first thumb-screen contact to the thumb lift that followed a successful paste. Recall that the full sequence involved at least five operations and that panning could involve several drag gestures. Significant effects were found for *Technique* ($F_{2,22}$=12.54, p<.001), *Task difficulty* ($F_{1,11}$=24.74, p<.001), and for the *Technique* x *Task difficulty* interaction ($F_{2,22}$=4.44, p<.016). Post-hoc multiple means comparison tests showed that for the condition with the rather large 60px buttons, TB (4.0s on average) was faster than CM (7.0s), MR falling in between (5.4s), not significantly different from TB or CM. However, for the more realistic condition with the 20px buttons of Windows Mobile applications, MR (5.8s) was much faster than both TB (8.1s) and CM (10.1s). The difference between the two blocks for MR performance was quite small and non significant.

The ANOVA showed a significant effect of *Technique* on panning time, measured from the first screen contact to the last lift of the thumb ($F_{2,22}$=6.27, *p*<.004): panning took less time with TB (0.68s) than MR (1.07s) and CM (1.17s). One explanation is that the final *x*-location to which the paste target was panned was technique dependent ($F_{2,22}$=17,27, *p*<.0001): paste targets were released closer to the right border of the 320x240px screen, that is, panned over a smaller amplitude, for TB (20px) than for CM (43px) and MR (53px). Presumably the participants found it more comfortable using CM and MR to operate not too close from the border (we will return to the screen location issue in the next experiment).

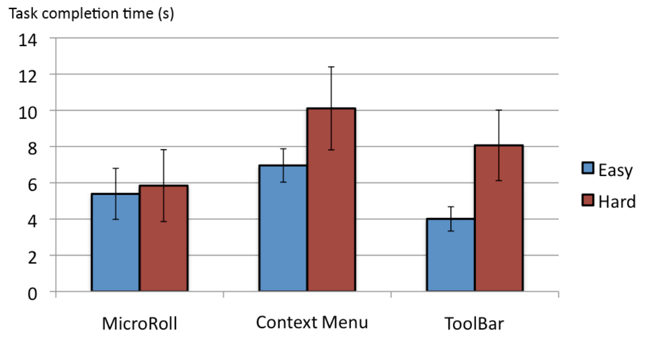

Fig. 6. Mean total time. Bars represent 95% confidence interval based on between-participant standard deviations.

Finally the ANOVA was run on total less panning time. Similar results were obtained, with significant effects for *Technique* ($F_{2,22}$=13.56, *p*<.001), *Task difficulty* ($F_{1,11}$=28.42, *p*<.001), and *Technique* x *Task difficulty* ($F_{2,22}$=5.5, *p*<.01). Post hoc tests gave similar results: for the easy task (60px buttons), TB (3.4s on average) was faster than CM (5.9s), with MR in between (4.4s), not significantly different from TB or CM. For the difficult task (20px buttons), MR (4.7s) was faster than both TB (7.4s) and CM (8.8s). Again, there was no significant change in MR performance from the first to the second block.

Discussion

One should not be too impressed by the fact that the best performance for this experiment was obtained with the TB technique in the easy task condition. We did manipulate the size of graphical objects involved in the TB and the CM techniques to acknowledge the fact that this variable is critical to both of these techniques, but one should keep in mind that a TB with 60px icons would be rather problematic on the small screens of real-life handhelds. The really interesting finding of this experiment is that the MR technique outperformed not only the CM technique unconditionally (with a 23% or 43% time saving, depending on menu-item size) but also our *realistic* implementation of the TB technique, that which used the sort of graphical icons used in the Windows Mobile environment (a 28% time saving).

EXPERIMENT 3

The goal of this experiment was to see how the techniques of interest fate for different screen locations in the case, often met in the real world, where objects are too small for direct selection. Participants again had to copy and paste objects but since in real-life applications the object to be copied may well be a text element, here we used 20x20px copy and paste areas (instead of 60x60px in Exp. 2). Another difference with Exp. 2 is that the object to be copied could appear in four different screen locations.

The TB technique was used as our baseline condition. The CM, found in Exp. 2 to elicit far slower performance than the two other techniques, was dismissed. Three commands were used instead of two: in addition to the copy and the paste buttons (for TB) or gestures (for MR), a *Precise-selection* command was provided for selecting the very small copy and paste targets. Because of their small size, the targets were selected by using a variant of *TapTap* [21], an automatic zooming technique for handhelds that improves over Zoom-Pointing [1].

TapTap and RollTap

The original TapTap technique requires tapping the screen twice to select a small target: (1) the first tap serves to select an area of interest; (2) this area is then automatically zoomed and displayed at the middle of the screen; (3) the second tap serves to select the target in the magnified view. An advantage of this design is that the zoomed target is generally pretty close to screen center when the second tap is performed. This second tap is hence executed in a remarkably small amount of time, which makes this technique especially fast [21].

TapTap was combined with MicroRolls by replacing its first tap with a top MicroRoll gesture. A nice feature of *RollTap,* this new precise-selection technique, is that it can be activated on request, just when needed. It will not interfere with usual interaction styles: tapping on a target does just what is expected.

Task

The task was essentially the same as in Exp. 2 but the number of operations was doubled (leaving the pan aside). Whether for copying or for pasting participants had to: (1) press the screen to select the area; (2) trigger the modified version of TapTap; (3) press the screen to select the magnified target; and (4) activate the copy or paste command. Modified TapTap was either triggered by performing a top MR, as explained above, or by tapping on the third button of the TB. Participants always had to perform all these steps and were not allowed to attempt to select targets directly (this would have caused numerous errors and made results hard to compare).

Apparatus and Participants

We used the same equipment as in Exp. 1. Twelve right-handers volunteers from our institution (aged 23-31, three female) participated.

Experimental Design

We used a 2 x 2 x 4 within-participant design using as factors the *technique* (TB, MR), the *level of difficulty* for button acquisition (easy 60px-high buttons vs. difficult 20px-high buttons), and the *screen location* of the to-be-copied object (obtained by dividing into four identical rectangular zones the space available under the large icon version of the toolbar). The order of presentation of conditions was counterbalanced within participants using Latin squares. For each technique, however, participants performed the easy task in a first block of trials before experiencing the more difficult condition in a second block.

Note that here, as in Exp. 2, difficulty was an irrelevant factor for the MR technique, where commands were activated without recourse to any graphical object. Thus for the MR technique, the first and second blocks of trials were identical replications. The justification for this design option is that it would have been unfair to allot half practice time to our novel technique, however easy to grasp, in its evaluation relative to a far more familiar technique.

In summary the experiment involved 12 participants × 2 techniques (TB, MR) × 2 levels of task difficulty × 4 target locations x 4 trials = 768 copy and paste sequences.

Results and Discussion

As any error had to be immediately corrected, in this experiment again the error rate was 0%, allowing task-completion time (measured in the same way as for Exp. 2) to be used as a safe net measure of performance.

Exp. 3 delivered two main findings concerning total task-completion time (Fig. 7). The first was that MRs generally outperformed the TB technique ($F_{1,11}$ = 10.60, p < .02). While using the TB technique it took participants on average 12.06s to complete the copy and paste sequence, they needed only two-thirds of that time (8.04s) when allowed to use the MR technique.

This conclusion obviously must be qualified by taking into account the considerable dependence of TB performance upon icon size (actually revealed in our ANOVA by a strong and highly significant technique × difficulty interaction ($F_{1,11}$ = 21.15, p < .002). As visible in Fig. 7, performance with the MR technique was roughly twice as fast as with the small-icon version of the TB (15.71s, a value that differs very significantly from 8.04s), and MR did at least as well as the easy, large-icon version of the TB (8.41s). Keeping in mind that this research is aimed at designing novel techniques for use on devices that all suffer from space scarcity, the fact that MRs actually allowed very fast performance at zero real-estate expense is a potent argument for its consideration.

The same pattern of results was obtained for total less panning time, unsurprisingly since panning occupied a modest and little variable proportion of total time (1.16s and 0.97s on average for MR and TB, corresponding to 14.4% and 8.0% of total completion time, respectively).

The second main finding was that performance was more dependent on the screen location of the gesture for the MR than the TB technique, as revealed by a significant technique × location interaction ($F_{3,33}$ = 7.29, p < .002). The pattern of data in Fig. 7 shows that MRs were more quickly performed in the left, rather than right half of the screen. Such a dependency of thumb operation is not very surprising given the high degree of motor idiosyncrasy of the thumb. But the data of Exp. 3 do show that MR gestures can be made successfully in various screen locations.

Our experience during the experimental sessions was that the MR technique, new to participants, was quite easy to learn. That impression was supported by the data of Exp. 3. Had the technique required substantial training, we would presumably have observed an improvement in its utilization from the first to the second block of trials. In fact, virtually no change was observed (from 8.16s to 7.93s, a difference far from statistical significance).

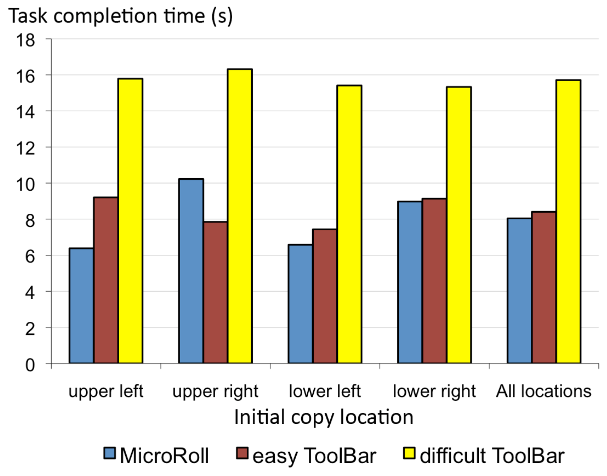

Fig. 7. Mean total task completion time for the four locations and the two techniques of Exp. 3.

SUBJECTIVE PREFERENCES

Several participants spontaneously complained about fatigue when using the CM and TB techniques with small buttons. It turns out that no reservation was expressed by the participants about the MR technique. We asked the participants to rank the techniques by order of global preference. MR was ranked first by a majority of participants, but the result fell short of significance. We also recorded from our participants subjective evaluations of the MR, TB, and CM techniques, using a set of Likert scales to quantify judgments on the techniques' merits, but no significant patterns emerged from the data (Fig. 8).

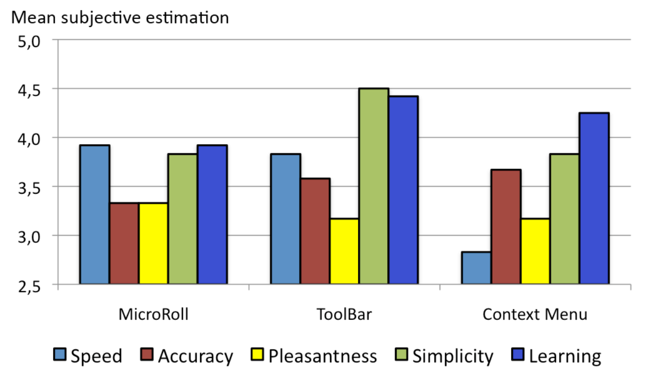

Fig. 8. Subjective Preferences (Experiments 2) on a Five-points Likert scale.

GENERAL DISCUSSION

According to the preceding, MicroRoll gestures are quite promising for selecting commands on passive touch-screens. Screen real estate is obviously a scarce resource on handheld devices. Context menus seem well suited as they allow to save that resource. Unfortunately, however, their speed performance is limited: Exp. 2 shows that MicroRolls are nearly twice faster than context menus in the realistic case of 20px buttons. MicroRolls can thus be deemed a better technique when the number of commands is limited.

Toolbars and similar interactors are the other common technique for issuing commands on handhelds with a GUI. Compared with context menus, toolbars have opposite advantages and drawbacks: they allow faster performance but they consume a substantial amount of screen space. The larger the buttons, the more serious this problem, with the consequence that in general applications (most notably Windows Mobile) only use small 20px buttons. Acknowledging that toolbar buttons, unfortunately, have to be reduced to that sort of size on handhelds, one is led to conclude from our data that for these devices MicroRolls are definitely more efficient than a toolbar: not only was performance 28% faster in Exp. 2, and twice as fast in Exp. 3, but this result was obtained with a technique that does not waste any screen real estate.

The halving of task-completion time with MicroRolls that we observed in Exp. 3 is especially worth considering as that was certainly our most realistic experiment, both the buttons and the copy and paste targets having small, most commonly observed, sizes. Finally, although MicroRolls were of course not designed to serve as a substitute for large-buttoned toolbars, it is interesting to note that they compared well even in this case, being slower, but not significantly so, in Experiment 2 and significantly faster in Experiment 3.

Of course there are situations where other techniques than MicroRolls are more appropriate. For instance, in contrast with menus, MicroRolls can only support a limited number of commands. However, an important property of MicroRolls is that they enjoy a high degree of compatibility with other techniques because they leverage an unoccupied input channel. For instance MicroRolls can be used together with menus, as these two techniques will not interfere with each other.

Besides, as already suggested, using a MicroRoll gesture instead of a temporal delay for opening context menus would probably further improve their performance. This design would also make it possible to use delays for triggering the MicroRoll novice mode: as for traditional Marking menus, a *RollMark* menu could be made to appear after a certain timeout to display the available commands to the user. Such a design would solve the well-known learnability problem characteristic of gesture-based interfaces. It would also facilitate the implicit learning of the expert mode, gestures remaining the same in novice and expert modes.

Several reasons can explain the efficiency of MicroRoll gestures. First, unlike context menus, they do not involve delays. Second, they require less information on the part of the user than does the pointing act required by the context-menu and toolbar techniques. A MicroRoll gesture, just like a normal Marking-menu gesture, requires the specification of just one direction: if its orientation must be contained in a given, usually fairly tolerant, angular sector, its amplitude suffers no maximum constraint thanks to the natural stop provided by the hand biomechanics. Moreover, techniques that require pointing movements are handicapped by the problem, quite inevitable in miniaturized interfaces, of a rather high ratio of fingertip size to screen size. The foregoing provides a general argument in favor of techniques based, like MicroRolls, on directional rather than pointing gestures.

One important outcome of this study is that MicroRolls were shown to combine successfully with TapTap, giving birth to an intuitive novel technique, which we called *RollTap*. This new precise-selection technique has two interesting properties: it can be activated on request and it does not interfere with usual interaction styles like standard tapping. In fact RollTap could be made to trigger multiple-level zooms, with the magnifying command issued iteratively in case of extremely small targets. This feature makes this new technique quite flexible.

PERSPECTIVES

The application scope of the finger-slide vs. finger-roll distinction in HCI seems fairly large. The one bit of information available in this distinction, which may be especially valuable when it comes to the extreme case of thumb interaction, remains potentially useful too in less constrained cases like multi-touch interaction, where it may provide an extra input channel to be combined with others. The potential usefulness of the slide vs. roll discrimination extends naturally to quite different situations where a keyboard is missing, like for example multi-person interaction on a wall screen and tabletops. In general, we believe the future of MicroRoll is in their association with other techniques with complementary potentialities. The RollMark and RollTap techniques presented in this paper are two steps in this direction. We plan to investigate in future research whether MicroRolls and other gestures can be efficiently concatenated.

ACKNOWLEDGMENTS

This research arose from a close collaboration with Alcatel-Lucent researchers, notably B. Aidant, J. Daigremont, and B. Legat, whom we warmly thank for their uninterrupted advice and encouragements. We gratefully acknowledge the financial support of UBIMEDIA, a joint laboratory of Alcatel-Lucent and Institut TELECOM. We thank the anonymous reviewers of this article for their insightful critiques and recommendations. Many thanks to G. Bailly and S. Malacria, who helped us in multiple ways, and to all our participants.

REFERENCES

1. Albinsson, P-A. and Zhai, S. (2003). High precision touch screen interaction. Proc. CHI'03, ACM Press, 105-112.
2. Bailly, G., Lecolinet, E., and Nigay, L. (2008). Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. Proc. AVI '08. ACM Press, 15-22.
3. Benko, H., Wilson, A., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. Proc. CHI'06, ACM Press, 1263-1272.
4. Buxton, W. (1990). A three-state model of graphical input. Proc. INTERACT '90. Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 449-456.
5. Cross, R. (1999). The bounce of a ball. Am. J. Physics, 67, 222-227.
6. Geißler, J. (1998). Shuffle, throw or take it! Working efficiently with an interactive wall. Proc. CHI'98, Extended Abstracts, 265-266.
7. Grossman, T., Dragicevic, P., Balakrishnan, R. (2007). Strategies for accelerating on-line learning of hotkeys. Proc. CHI'07, ACM Press, 1591-1600.
8. Guimbretiére, F., Winograd, T. (2000). FlowMenu: combining command, text, and data entry. Proc. UIST'00, 213-216.
9. Karlson, A. K., Bederson, B. B. (2008). One-handed touchscreen input for legacy applications. Proc. CHI'08, ACM Press, 1399-1408.
10. Karlson, A. K., Bederson, B. B., SanGiovanni, J. (2005). AppLens and launchTile: two designs for one-handed thumb use on small devices. Proc. CHI'05, ACM Press, 201-210.
11. Karlson, A., Bederson, B., Contreras-Vidal, J. (2007). Understandingon User Interface Design and Evaluation for Mobile Technology, Idea Group.
12. Kurtenbach, G. and Buxton, W. (1991). Issues in combining marking and direct manipulation techniques. Proc. UIST'91, ACM Press, 137-144.
13. MacKenzie, I. S., Oniszczak, A. (1998). A comparison of three selection techniques for touchpads. Proc. CHI'98, ACM Press, 336-343.
14. Olwal, A., Feiner, S., Heyman, S. (2008). Rubbing and Tapping for precise and rapid selection on touch-screen displays. Proc. CHI'08, ACM Press, 295-304.
15. Oniszczak, A., MacKenzie, I. S. (2004). A comparison of two input methods for keypads on mobile devices. Proc. NordiCHI '04, ACM Press, 101-104.
16. Parhi, P., Karlson, A., Bederson, B. (2006). Target Size Study for One-Handed Thumb Use on Small Touchscreen Devices. Proc. MobileHCI'06. 203-210.
17. Pascoe, J., Ryan, N., Morse, D. (2000). Using while moving: HCI issues in fieldwork environments. ACM Trans. Comput.- Hum. Interact. 7(3):417-437.
18. Pook, S., Lecolinet, E., Vaysseix, G., Barillot, E. (2000). Control menus: execution and control in a single interactor. Proc. CHI'00, ACM Press, 263-264.
19. Potter, R. L., Weldon, L. J., Shneiderman, B. (1988). Improving the accuracy of touchscreens: an experimental evaluation of three strategies. Proc.CHI88, ACM Press, 27-32.
20. Resnick, R. & Halliday, D. (1966). Physics, Part I. New York: John Wiley & Sons, Inc.
21. Roudaut, A., Huot, S., Lecolinet, E. (2008). TapTap and MagStick: improving one-handed target acquisition on small touchscreens. Proc. AVI'08, ACM Press, 146-153.
22. Rubine, D. (1991). Specifying gestures by example. SIGGRAPH Comput. Graph. 25, 4 (Jul. 1991), 329-337.
23. Smith, G., schraefel, m. c., Baudisch, P. (2005). Curve dial: eyes-free parameter entry for GUIs. Proc. CHI'05 Extended Abstracts, ACM Press, 1146-1147.
24. Vogel, D., Baudisch, P. (2007). Shift: a technique for operating pen-based interfaces using touch. Proc. CHI'97, ACM Press, 657-666.
25. Yatani, K., Partridge, K., Bern, M., Newman, M. W. (2008). Escape: a target selection technique using visually cued gestures. Proc. CHI'08, ACM Press, 285-294.