Leveraging Finger Identification to Integrate Multi-touch Command Selection and Parameter Manipulation

Alix Goguey[a,*], Daniel Vogel[b], Fanny Chevalier[a], Thomas Pietrzak[c], Nicolas Roussel[a], Géry Casiez[c]

[a]*Inria Lille - Nord Europe, 40 Avenue du Halley, 59650 Villeneuve-d'Ascq, France*
[b]*Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada*
[c]*Université de Lille, Cité Scientifique, 59650 Villeneuve-d'Ascq, France*

Abstract

Identifying which fingers are touching a multi-touch surface provides a very large input space. We describe FingerCuts, an interaction technique inspired by desktop keyboard shortcuts to exploit this potential. FingerCuts enables integrated command selection and parameter manipulation, it uses feed-forward and feedback to increase discoverability, it is backward compatible with current touch input techniques, and it is adaptable for different touch device form factors. We implemented three variations of FingerCuts, each tailored to a different device form factor: tabletop, tablet, and smartphone. Qualitative and quantitative studies conducted on the tabletop suggests that with some practice, FingerCuts is expressive, easy-to-use, and increases a sense of continuous interaction flow and that interaction with FingerCuts is as fast, or faster than using a graphical user interface. A theoretical analysis of FingerCuts using the Fingerstroke-Level Model (FLM) matches our quantitative study results, justifying our use of FLM to analyse and validate the performance for the other device form factors.

Keywords: multi-touch; finger identification; shortcuts; command selection; parameter control; direct manipulation

Word count: 11200

1. Introduction

A common belief is that multi-touch input is an excellent match for direct manipulation interaction. Certainly panning and zooming with two-finger pinch-to-zoom, or simultaneously rotating, translating and scaling objects with multiple fingers are better semantic and articulatory translations [1] of real world actions than dragging scrollbars,

*Corresponding author
Email addresses: alix.goguey@inria.fr (Alix Goguey), dvogel@uwaterloo.ca (Daniel Vogel), fanny.chevalier@inria.fr (Fanny Chevalier), thomas.pietrzak@univ-lille1.fr (Thomas Pietrzak), nicolas.roussel@inria.fr (Nicolas Roussel), gery.casiez@univ-lille1.fr (Géry Casiez)

sliders or transformation handles. But, a direct manipulation interface should also provide access to a variety of *commands* to trigger global actions (*e.g.* "undo") and contextual actions (*e.g.* "delete this object"), activate modes (*e.g.* "start drawing rectangles") and modify modes (*e.g.* "draw rectangles from the centre"). With a proportionately small number of possible multi-touch input actions, most commands are represented visually, away from the object of interest in toolbars, palettes or menus which occupy screen space and break interaction flow into a complex and time consuming "back-and-forth" graphical syntax [2]. The restricted display area together with limited input vocabulary result in real-world touch interfaces that tend to be less rich in functionalities than their desktop counterparts[1].

In desktop computing, keyboard shortcuts provide an effective alternative to clicking on graphical buttons when issuing commands [4]. Once mastered, they decrease command activation time, enable redundant palettes to be hidden, and allow for a more fluid flow between command selection and the continuous control of command parameters. For example, pressing the `R` key with one hand could activate a "draw rectangles" mode and then the rectangle location and size parameters can be controlled with the mouse in the other hand. Current multi-touch input primarily uses simultaneous touch points and long presses which provides a far more limited vocabulary than keyboard shortcuts.

No wonder researchers and designers are constantly searching for ways to increase the multi-touch input space using techniques like gestures [5], touch locations [6], touch patterns [7], normal and tangential forces [8], discriminating finger parts [9] and more. Yet, these still treat all fingers equally which confines the input vocabulary – identifying *which* fingers are touching the surface provides a much larger input space. Recent studies show that even for single-touch command selection, finger identification is a promising solution to increase the touch input vocabulary [10]. Robust finger identification sensing is on the horizon [11, 12], but examples of the interaction it enables have only been isolated point designs without cohesive design. For example, Marquardt *et al.*built an exploratory toolkit for tabletops [13] and explored a range of interaction technique ideas to emphasize the toolkit's expressiveness, but they did not provide a consistent way to apply finger identification to direct manipulation interfaces. Our research moves beyond point designs by contributing a cohesive finger identification-enabled interaction technique focused on the classic goal of merging command selection with direct manipulation of command parameters [14, 15].

We contribute FingerCuts, an interaction technique inspired by keyboard shortcuts that leverages finger identification for direct manipulation. In desktop computing, experts typically use keyboard shortcuts in three steps: 1) the non-dominant hand triggers a command with a key; 2) the dominant hand manipulates with a pointing device; and 3) the non-dominant hand optionally tunes manipulation with a modifier key. With finger identification-enabled touch, a similar three-step pattern can be adopted and made universal so that subsequent command selection and manipulation are perfectly merged: 1) one set of fingers selects a command mapping for another set of fingers (*e.g.* the thumb and index finger chord in Figure 1-a shows a set of possible drawing

[1]Wagner *et al.*[3] note that the desktop version of Adobe Photoshop (CS6) has 648 menu commands while the tablet version (Express) has only 35 commands.

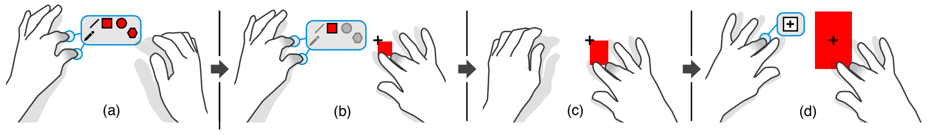

Figure 1: A bimanual tabletop example of how FingerCuts merges command selection with direct manipulation of command parameters with feed-forward: (a) specific finger chords with non-dominant hand displays feed-forward showing possible commands, *e.g.* thumb and index finger chord shows different drawing modes; (b) dominant hand selects command using specific corresponding finger, *e.g.* middle finger triggers mode activation command to "start drawing rectangle"; (c) dominant hand movement provides continuous control of command parameters, *e.g.* specifies the size of the rectangle; (d) other specific fingers with non-dominant hand further tunes the command parameters, *e.g.* middle finger modifies draw rectangle mode to "draw rectangle from the centre". See Section 6 for a restricted bimanual example with a tablet and unimanual example with a smartphone.

commands); 2) another finger triggers a command and immediately begins direct manipulation of command parameters (*e.g.* the middle finger draws and adjusts the size of a rectangle in Figure 1-b,c) — at this point, the fingers used to to select the command mapping can be released; and 3) the first set of fingers optionally tunes the manipulation (*e.g.* the middle finger of the non-dominant hand specifies the mode of drawing, namely from the centre, in Figure 1-d). For novices, the progression of actions provides an opportunity to display simple feed-forward [16] for discoverability. For experts, this progression of overlapping finger actions forms a tightly coupled phrase that can be chunked [17].

We perform a systematic analysis of the finger identification interaction space to motivate practical design considerations for FingerCuts. These argue for maintaining backward compatibility with existing multi-touch techniques like pinch-to-zoom, and guide designers when applying the technique to different device form factors. To demonstrate how FingerCuts works in different device form factors, we implemented it in a vector drawing application for a tabletop, a document annotation application for a tablet, and a messaging application for a smartphone[2]. We evaluate the tabletop version in two user studies. A qualitative study suggests that with some practice, FingerCuts is expressive, easy-to-use, and increases a sense of continuous interaction flow. A quantitative study shows that FingerCuts can be as fast, or faster than a traditional graphical user interface, primarily due to the lower number of operations required. A Fingerstroke-Level Model (FLM) analysis supports the results of the tabletop quantitative study and enables us to generalize these results to the tablet and smartphone form factors.

[2]Video demonstration at: https://www.dropbox.com/s/u22lc0l71c13ry7/FingerCuts-ijhcs.mp4?dl=0

2. Background and Related Work

A direct manipulation interface must provide a way to issue *commands* to the system. There are commands for global actions (*e.g.* “undo”) and contextual actions to be performed on a specified object (*e.g.* “delete this rectangle”). There are also commands to activate *modes*, where the system should start interpreting input in a particular way (*e.g.* “start drawing rectangles”). Modes are conceptually associated with real world *tools*. Commands can also modify the current mode (*e.g.* “keep drawing rectangles, but draw them from the centre”). These *modifiers* are often temporarily maintained by a kinesthetic action, such as holding a key down. At a command level, this means a key down sends a command to start modifying the current mode and a key up stops modifying the current mode. This same temporary kinesthetic action can be applied to modes, which are then called ‘quasimodes’ [18].

Providing an efficient way to trigger all these commands is challenging with multi-touch. Multiple points of contact provide some expression, but nowhere near the discrete input space of keyboard shortcuts. Techniques to increase the multi-touch input space have focused on spatial information, dwell timeouts, motion to enrich the interaction vocabulary, and even cross device interactions [19, 20]. However, these also increase time (*e.g.* dwell) or space offset (*e.g.* gestures) and they move the user’s focus away from the primary object of interest [2].

2.1. Finger Identification

Finger identification — associating specific fingers and hands with touch points — is a way to increase the multi-touch input space. Significant previous work has tackled the technical sensing problem with approaches like: inferring finger identity based on geometric relationships between touch contact points [21, 22, 23, 24, 25, 3]; using an overhead camera to track bare hands [26] or fingers with coloured rings [27]; wearing gloves with fiducial markers [13]; recognizing fingerprints [28, 11]; forearm electromyography [12]; and using RFID attached to fake plastic nails [29]. It seems inevitable that one day finger identification will be a standard feature of consumer multi-touch devices. Our interest is interaction, not sensing, so we focus our review on interaction techniques enabled by finger identification rather than the underlying technologies.

2.1.1. Invoking Commands with Fingers and Chords

Independent of sensing approach, a common interaction technique to demonstrate the potential of finger identification is associating a different command with each finger. Sugiura *et al.* [28] used the index, middle, ring, and little fingers to operate a music player and add web browser bookmarks. Marquardt *et al.* [13] mapped the middle and little fingers to cut and copy commands. However, with only 10 fingers, a limited set of commands can be triggered.

By recognizing *chords* — the simultaneous contact of two or more fingers against the interactive surface — the input space can be significantly increased. Tapping with any two fingers to open a context menu can be considered a simple type of chord-activated command. By counting the number of fingers of each hand in contact with a surface, Bailly *et al.* [30] invoke up to 25 commands. Their technique is used for global action commands like *undo*, where chords are performed without touching any object. Gutwin *et al*.'s FastTap [31] encourages experts to use chords to simultaneously select multiple command buttons displayed in a hand-sized grid. HandMark [32] is a related two-handed method where the chord pattern selects among different grids of command buttons. The grids are spatially arranged around the chording hand so the other hand taps commands exploiting proprioception. Given the area required to perform most chords, using these three techniques for contextual commands on a specific object would be difficult, and using them for modal commands with parameters (*e.g.* "start drawing rectangles, and begin the first one here") does not appear to be possible.

Other chording techniques assume some level of finger identification, but they take the same basic approach of mapping chords directly to commands. Benko *et al*. [12] mapped chord-like thumb-index and thumb-middle pinch gestures to copy and cut commands respectively. Harrison *et al*.used the same principle associating different part of the finger tip to colours [9] The Palm Menu [21] and Arpège [33] both display on-screen place holders for specific fingers or chords to trigger commands. Arpège uses an interesting method enabling chord-to-command discoverability: commands are not triggered until all fingers are lifted so that chords can be entered progressively with feed-forward displaying the current and related chord mappings. Although delaying command triggering until all fingers are lifted enables feed-forward, it also prevents immediate continuous manipulation of the command's parameters. Lepinski *et al*. [25] go beyond mapping one command to one chord by triggering different commands by swiping all chorded fingers in one of eight directions, but this also makes subsequent direct manipulation difficult.

2.1.2. Merging Command Selection and Direct Manipulation

Merging command selection and direct manipulation reduces articulatory and semantic distances [1]. The articulatory distance is reduced because the command occurs where it is expected to be used. The semantic distance is reduced because things can be expressed in a straightforward fashion with no intermediate action. As an example in another input context, Guimbretière *et al*. [14] argue that the important aspect of Toolglass [15] is that command selection and direct manipulation of the command's parameters occur in a single stroke. Merging command selection and this direct manipulation is difficult with multi-touch chords because multiple fingers have an ambiguous interaction point and hand occlusion makes a ToolGlass-like strategy hard to realize.

Interaction techniques that merge command invocation and direct manipulation are similar to those surveyed above: a one-to-one mapping is determined between fingers or chords and commands, but command invocation is handled in a way that makes subsequent direct manipulation possible. Benko *et al*.associated colours with each finger so that a 'draw red line' mode can be activated and the line drawn in one action [12]. Malik *et al*.used the left and right index fingers to activate panning or pointing modes with immediate continuous control of commands parameters [26]. They also used thumb-index chords to activate a resize mode, where the size is immediately adjusted using the inter-finger distance. A thumb-index-middle chord is also used to activate a zoom mode, where raising the index finger zoomed-in and lifting the thumb zoomed-out. The problem with these interactions is that the command mappings are not easily discoverable like the Arpège feed-forward technique.

Marquardt *et al.*addressed command discoverability without feed-forward using knuckle touches to show finger-to-command mappings [13]. For example, touching the knuckle of the middle finger reveals that the cut command is mapped to that finger tip. However, this discoverability method is not used consistently — knuckles are sometimes used to copy or move piles of items. It is also not clear if a knuckle preview would work well with chords. Of particular relevance to our work is how Marquardt *et al.*use non-dominant chords to mix a colour (by adding blue with the thumb, yellow with index, and red with middle) for the dominant hand to draw with. Using the non-dominant hand to modify the dominant hand command mode is a simple example of the selector step in our three-step interaction technique.

Previous work has primarily used finger identification to map commands to individual fingers and chords. Using finger identification to achieve the classic goal of merging command invocation with direct manipulation has been demonstrated primarily with single fingers, though Marquardt *et al.*'s non-dominant hand colour mixer hints at a way forward for chord activated commands. The challenge is that previous methods for discoverability from Arpège and Marquardt *et al.*are not compatible with merging command invocation with direct manipulation.

3. FingerCuts Interaction Technique

FingerCuts is an interaction technique using finger identification with multi-touch input in a direct manipulation interface. The primary aim is to make command selection discoverable and efficient in terms of time and space, especially when commands can be merged with continuous control. A central idea is to use finger identification to bring the kind of expert command invocation capability of conventional keyboard shortcuts and modifier keys to multi-touch.

3.1. Finger Identification Input Space

Before explaining the interaction technique, we enumerate the finger identification input space. A straightforward application of finger identification is to map different commands to each finger like some techniques above [13, 12, 28]. The problem is that with only 5 fingers on each hand, a single finger input space is limited. Combining multiple fingers into *chords* significantly increases possibilities. In the remainder of this paper, we will refer to single finger contacts and multi-finger chords as the combined space of *contact configurations*, or simply *configurations*.

With finger identification, any chord involving k fingers among 10 can theoretically be performed in $C(10,k)$ ways, triggering $C(10,k)$ different commands[3]. The total number of 10 finger chords is $\sum_{k=2}^{10} C(10,k) = 1013$ (summing all but the first column of the first row of Table 1). The temporal finger order when forming a chord can also be used as a distinguishing feature. In music, any chord whose notes are not played simultaneously is called a *broken chord*. Permuting the finger order in these broken chords would increase the total number of chords available with 10 fingers

[3]$C(n,k)$ means *k-combinations*, the number of distinct subsets of size k in a size n set. It is equivalent to the binomial coefficient, $\binom{n}{k}$.

k	1	2	3	4	5	6	7	8	9	10	Total
$C(10,k)$	10	45	120	210	252	210	120	45	10	1	1023
$C(5,k)$	5	10	10	5	1						31
$C(4,k)$	4	6	4	1							15
$C(3,k)$	3	3	1								7
$C(2,k)$	2	1									3

Table 1: Number of k-combinations of n fingers for selected values of n.

to $\sum_{k=2}^{10} P(10,k)$, nearly 10 million[4]. In addition, any contact configuration can be complemented with existing techniques such as dwell, contextual menus, gestures, pressure, and so forth.

This huge input potential is purely theoretical since the number of different chords one can physically articulate and the number of chord-to-command mappings one can remember is most probably much smaller — it has been found that people can comfortably execute up to 480 chord gestures [33]. Wagner *et al.*studied how to help users remember commands by improving mappings between command groups and finger chords [3]. For 9 chords, they found that a categorical mapping is better than a random one.

To exploit the potential of using this input space with direct manipulation interfaces, finger identification input must be applied using reasonable design considerations that recognize physical and cognitive limitations and are organized under a unifying paradigm.

3.2. A Three-step Interaction Technique

FingerCuts is a three-step interaction technique inspired by desktop keyboard shortcut usage. Using keyboard shortcuts for tool-like modes has the following three steps:

1. the non-dominant hand invokes the mode activation command using a key (*e.g.* `R`) or a key combination (*e.g.* `Ctrl+R`);
2. the dominant hand manipulates command parameters with the pointing device like the mouse;
3. the non-dominant hand is used again to modify how parameters are manipulated (*e.g.* by adding constraints) using modifier keys like `Ctrl`, `Alt`, and `Shift`.

The FingerCuts three-step interaction design pattern defines a different step 1 than that above: a command-to-finger mapping is selected, and we merge command invocation and parameter manipulation in step 2. Our step 3 is a similar modification step as

[4]$P(n,k)$ means *k-permutations*, the number of distinct permutations of size k in a size n set

that of keyboard shortcuts. Note that keyboard shortcuts introduce a spatial separation between command invocation and parameter manipulation across the first two steps which is undesirable. Formally, we define three sets of finger configurations (selectors, triggers, tuners) used in three steps:

1. a *selector* finger configuration[5] selects a group of command mappings;
2. a *trigger* finger configuration triggers one command from the available mappings and begins direct manipulation of the command parameters if applicable;
3. a *tuner* finger configuration is optionally used to tune (*i.e.* modify) the ongoing execution of the command.

We refer to the set of all selector finger configurations as *the set of selectors*, and likewise for the sets of triggers and tuners.

3.2.1. Illustrative Example

We return to Figure 1 to illustrate these three steps using formal terminology on a tabletop device context with a right handed person. Figure 1-a is step 1, where a left thumb and index finger chord is the *selector* configuration performed to select a command mapping. The command mapping is communicated to the user as a crib-sheet. In this case, the selected commands activate different drawing modes like lines, rectangles, ellipses, polygons, etc. Each of these commands are mapped to individual fingers on the right hand forming the complete set of available triggers. Figure 1-b is step 2, where the right middle finger is the trigger configuration performed to simultaneously trigger the "activate rectangle drawing mode" and begin direct manipulation of the rectangle width and height (the parameters for the draw rectangle command). Note that the selector combination can be freely released with no implications as early as the command has succesfully been selected: while continuing to draw the rectangle, the left hand is lifted in Figure 1-c. Figure 1-d is step 3, where the middle finger of the left hand is the tuner configuration performed to tune the ongoing execution of the draw rectangle command so that the rectangle is drawn from the centre rather than from the top left corner.

3.3. Design Justification, Variation, and Analysis

In the example above, the sets of selectors and tuners are most naturally assigned to the non-dominant hand, and the set of triggers to the dominant hand. For smaller devices and unimanual contexts this division is relaxed. We discuss different ways of defining and using selectors and triggers for different device form factors in later sections. In practice, we recommend the set of tuners be a subset of selectors and the sets of selectors and triggers be two disjoint sets.

To further guide the design and application of FingerCuts, we identified the following set of design considerations (listed in approximate order of importance):

[5]Recall that we refer to single finger contacts and multi-finger chords as the combined space of contact configurations.

Discoverability and Reinforcement: people need consistent ways to discover many finger-to-command mappings, instantaneous visual feedback of actions is important for reinforcement.

Hand Ergonomics: physical limitations of finger, wrist, and arm joints limit possible unimanual and bimanual chords, postures, and gestures.

Device Ergonomics: the number of hands, fingers, and usable chords are influenced by device size (*e.g.* smartphone, tablet, table) and device context (*e.g.* holding tablet, tablet on desk).

Backward Compatibility: standard finger mappings for existing interactions like Pinch-to-Zoom and Rotate-Scale-Translate (RST) should be preserved.

Consistency: new finger identification and chorded interaction techniques should be as consistent as possible across devices and applications.

Temporal Ambiguity: techniques should be robust to unsynchronized finger contact times when performing multi-finger chords.

Spatial Ambiguity: for single point continuous manipulation (such as drawing), techniques should disambiguate what geometric feature of a multi-finger chord is used as the interaction point.

Customization: people should be able to customize finger-to-command mappings to comply with personal ergonomic, cognitive, or device abilities.

Using these design considerations, the three steps may be further analyzed and justified:

Step 1, Command Set Selection: The user begins by selecting a command set mapping by performing a selector configuration. If the number of selectors is small due to hand or device ergonomics, then other strategies like movement direction can be used. What is important is that performing a selector configuration should always act as a quasimode. This way, the order and timing in which a chorded selector configuration is formed does not matter, making chords faster to recognize since no delayed time window is needed to resolve *temporal ambiguity*. Note that in most applications, the null selector configuration (no fingers) selects a default command mapping.

Step 2, Command Invocation and Parameter Manipulation: With a command set mapping selected, the user invokes the command using a trigger configuration, and the selector configuration is no longer required to be in contact with the surface. Separating command set selection and command invocation like this alleviates the fear of accidentally triggering an undesired operation, with the associated recovering cost. In addition, these first two steps offer the opportunity to integrate feed-forward and feedback to improve the *discoverability and learnability* of trigger configurations. The command map crib-sheet is shown near the fingers performing the selector configuration to visualize what commands are mapped onto trigger ones. Once a command is initiated, subsequent movement of the trigger fingers manipulates command parameters, providing the usual benefits of merging command selection and parameter control by direct manipulation [14]. In addition, the selector fingers may be lifted any time after the command is invoked.

Step 3, Parameter Manipulation Tuning: While manipulating command parameters, the user has the option to tune their control using a tuner configuration (as long as the selector configuration has been lifted since the command selection). Tuning can be a different way of interpreting manipulated command parameters, such as drawing rectangles from the center rather than from a corner in the example above, or adding constraints such as grid snapping, horizontal or vertical alignment, maintain aspect ratio, and so forth. The number of possible tunings is often small enough to map them to single fingers, which also has the benefit of applying several tunings at once by forming an ad hoc chord, for example to simultaneously maintain aspect ratio and use grid snapping.

3.3.1. Integrated feedback and feed-forward

For experts, this three-step progression of overlapping finger actions can form a tightly coupled phrase that can be chunked. For novices, integrated feedback and feed-forward provides scaffolding to discover and learn the interaction as they progress towards expert skill. The command map crib-sheet is shown when any selector configuration is performed without being quickly followed by a trigger configuration mapped to a command. The crib-sheet is used only as an information display, it does not contain buttons. Instead, it serves both as feedback for the selector configuration just performed and feed-forward for the trigger configurations that may follow, by illustrating the corresponding commands. Note that Vermeulen *et al*. [34] define feed-forward as something that "tells users what the result of their action will be." The crib-sheet is a simple form of feed-forward where the icons tell the user what commands will be associated with their dominant fingers while the current non-dominant chord is maintained. During parameter manipulation, a crib-sheet is also shown when a tuner configuration is performed to provide feedback on the selected tuning modes.

3.3.2. Number of available commands

The number of commands that can be triggered using this approach depends on N_{sel} and N_{tri}, and trigger configurations.

$$\underset{\substack{\text{default}\\\text{mapping}}}{N_{tri}} \quad + \quad \underset{\substack{\text{alternate}\\\text{mappings}}}{N_{sel} \times N_{tri}}$$

The lower part of Table 1 shows the numbers of k-combinations for 5 to 2 fingers. A total of 31 chords can theoretically be produced using the five fingers of a hand, but this number is reduced to 15, 7, and 3 when using a fixed selection of 4, 3, or 2 fingers. When using the non-dominant hand for the set of selectors and the dominant one for the set of triggers, the maximum number of commands theoretically accessible using our approach is $31 + 32 \times 31 = 1023$. The choice of the actually useful configurations of selectors and triggers is crucial to reduce and organize this interaction space. One may favour numerous selector configurations and a few trigger ones, or the opposite, or choose a middle ground. For example, using combinations of up to 3 of 4 fingers of the non-dominant hand and the single-touch combinations of the dominant one, one could select $((4+6+4)+1)\times 5 = 75$ commands, which should be enough for many applications[6].

Although the above examples assume all selectors are performed with the non-dominant hand and all triggers with the dominant hand, this is not imposed by our technique. For the most part, the sets of selectors and triggers might well be specifically tuned for a particular application or device context. While smartphone and tablet applications are likely to favour a symmetrical mapping between the fingers of the two hands to support ambidextrous use, tabletop applications may foster asymmetric bimanual interaction (*device ergonomics*).

Commands with parameter such as drawing a shape or adjusting its size are more suited for single-finger triggers (*e.g.* using the index or middle finger), while parameterless commands like "save document" are more suited for chords (*e.g.* thumb-index-middle). Single finger triggers are inherently suited for commands involving up to two parameters, *e.g.* adjusting an object's corner position (*spatial ambiguity*). The index and middle fingers should also be favoured when precise positioning is important (*hand ergonomics*).

3.3.3. Support for the novice-expert transition

The crib-sheet visualization enables novices to review possible trigger configurations and corresponding commands without risk of triggering a command by accident (Figure 1-a). Similar to marking menus [36], as novices begin to recall what selector and trigger configurations to trigger a command, they can perform both configurations quickly using exactly the same physical movements. To further encourage novice-to-expert transition, a short delay can also be inserted before revealing the crib-sheet.

Since FingerCuts is inspired by shortcut key usage, we anticipate people can understand the principles of this technique by transferring their knowledge from the three steps used with keyboard shortcuts. The separation between selectors and triggers allows the creation of groups of commands similar to those usually found in menus or toolbars. This way of organizing commands by semantic similarity has also been encouraged by Wagner *et al.* [3] to help users remember chord-to-command mappings.

To discover and explore commands for which no feed-forward is provided (such as the default command mapping), a help mode can be explicitly invoked by touching the input surface with all selector and trigger fingers. This special two handed help configuration is easily remembered and metaphorically relates to querying the system to display all finger mappings. Once in the help mode, the application provides an interactive overview of all of the selector, trigger and tuner configurations and their relations.

[6]Malacria *et al.* [35] found 15 to 112 hotkeys in the 30 mainstream Mac applications they examined ($\mu = 55.31, \sigma = 20.53$)

3.3.4. Compatibility with previous techniques

Finger identification is *backward compatible* with most existing multi-touch techniques such as tap, double tap, gestures (*e.g.* flick), dwell, and pressure. This is because finger identification is orthogonal to these techniques: it can be used in addition to or in combination with techniques like these. For example, a single finger trigger configuration can be overloaded so that tapping triggers one command and flicking another command. One could imagine overloading a single finger configuration further with directional flicks for even more commands, hard versus soft taps, *etc*. In practice, this type of overloading should be used for semantically related commands. For example, flicking a thumb in one direction for “undo” and flicking the thumb in the opposite direction for “redo”.

Being able to perform the same command using simple finger counting (mapping a specific number of any fingers to a command) can be convenient and it matches all multi-finger, multi-touch interactions. For example, rotating and scaling an object can be done with any two fingers in most current multi-touch interfaces. Mapping the same command to different trigger configurations is possible with finger identification of course, but mapping to all configurations of *n* fingers to one command clearly defeats any advantage. In practice, reasonable *backward compatibility* with finger counting techniques can be achieved by identifying which fingers are involved most of the time with a given finger counting technique and map these configurations to the same command. For example, people often use their thumb and index fingers to rotate and scale small objects [37], and especially on tabletops, they may use two index fingers to rotate and scale large objects (*device ergonomics*).

3.3.5. Configuration and personalization

There are many possible finger configurations and many command mappings to choose from. The way these are assigned not only depends on the type of application: device size and context of use are also important factors to be taken into consideration (*device ergonomics*). For example, a photo editing application on a tablet typically has a lower number of commands compared to its desktop counterpart and applications will often be interacted with one hand while the other hand holds the tablet [38]. In such a scenario, selectors, triggers, and tuners could be chosen so as to assign all selectors and commands to one single hand, and mirror this mapping to the other hand to allow for the same usage regardless of which hand is used. In a more complex application like a photo editor on a tabletop, where the number of commands is high and the two hands are available, the ideal asymmetrical mapping of commands with selectors and tuners on the non-dominant hand and triggers on the dominant hand. In the latter case, the interface should provide a way to configure handedness (*customization*).

Providing means for personalizing the interface is another way to assist skill development. Users could personalize selectors, triggers, and tuners, the configurations used, and the corresponding mappings to application commands and parameters. Users might want to remap the commands they most frequently use to certain fingers (*customization*). We anticipate that personalization might be especially important when using a small number of fingers in the three sets to make optimum use of the more limited input space.

3.4. Summary

The FingerCuts interaction technique is inspired by keyboard shortcuts by defining three sets of fingers (selectors, triggers, and tuners) to be used in three steps. These finger sets and three-step approach use integrated feedback and feed-forward to improve discoverability and novice-to-expert transition. The designer's task is to define these three sets and the respective command mappings according to the device form factor, context of use, and type of application, using guidelines provided above. In the following sections, we demonstrate how FingerCuts can be applied to applications for a tabletop, tablet, and smart phone.

4. Tabletop Demonstration Application

Our primary demonstration for FingerCuts is a vector-based drawing application on a tabletop form factor. When creating a vector drawing, people often have to switch back and forth between many different drawing modes, such as creating a shape, transforming it, and editing its control points. This serves as a canonical example where bimanual input is possible and natural and the full capability of FingerCuts can be explored. In Section 5, we report the results of two studies evaluating this form factor and in Section 6 we describe two additional demonstrations on tablet and smartphone form factors. An accompanying video demonstrates FingerCuts on the three device form factors[7].

4.1. Finger Identification Technique

Our focus is not on the technical problem of finger identification, but to realize the interaction technique we need to reliably identify which fingers are touching. Marquardt *et al*.'s fiducial marker glove [13] is not compatible with capacitive touch sensors, but our high level approach is similar: we add something to each finger to make their tracking easier.

We use five Gametraks — originally designed to track golf club swings — to track the 3D position of 10 fingers at 125Hz using the *libgametrak*[8] library. The Gametraks are mounted above the tablet and the strings are attached near each finger tip using a ring-like loop of fabric (Figure 2). The strings and rings add some visual clutter that remain minimally intrusive.

The 3D tracked finger positions are transformed to the 2D reference frame of a 32" 3M^{TM} C3266PW multi-touch monitor laid flat on a desk. This is achieved using a homography calculated from a short calibration procedure where tracked finger positions are sampled at known positions on the display. This finger identification software sends custom TUIO events annotated with hand and finger IDs to the touch device application.

4.2. Finger Configuration Notation Syntax

In this and following sections, we use a concise notation to describe finger configurations. This syntax can be explained with three representative examples: "index" means a

[7]Video demonstration at: https://www.dropbox.com/s/u22lc0l71c13ry7/FingerCuts-ijhcs.mp4?dl=0
[8]https://github.com/casiez/libgametrak

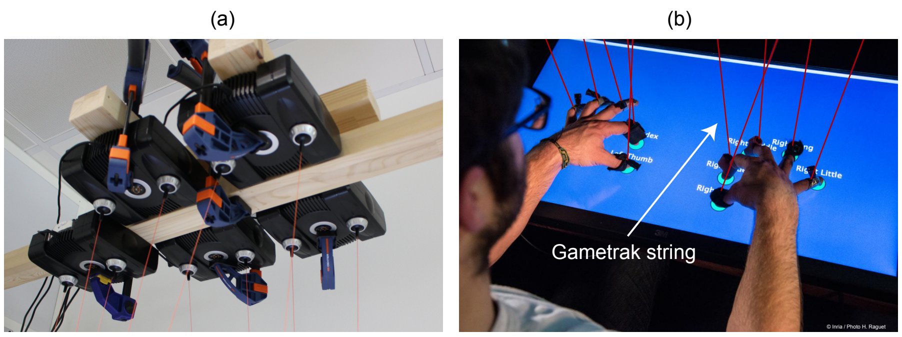

Figure 2: Finger identification technique for the tabletop: (a) five Gametrak devices are mounted above the tabletop; (b) strings from the Gametracks are attached to each finger.

single index finger is used on the *dominant hand*; “index+middle” means an index finger and middle finger *chord* performed with the dominant hand; “nd-index+nd-middle” means an index finger and middle finger chord performed with the *non-dominant* hand.

4.3. FingerCuts for a Vector Drawing Application

Given the large table form factor, we mapped finger selectors to the non-dominant hand and finger triggers to the dominant hand. There are 7 different sets of command mappings associated to the non-dominant hand (Figure 3-1a-h and first column of Table 2). These sets are activated by configurations composed of combinations of three non-dominant hand fingers: nd-thumb, nd-index, nd-middle, nd-thumb+nd-index, nd-thumb+nd-middle, nd-index+nd-middle, and nd-thumb+nd-index+nd-middle. A *default command set* is active for the null selector configuration — we chose the default set to be the same as the nd-thumb set.

Each command set is comprised of at most 5 commands. This way each command can be triggered with individual trigger fingers on the dominant hand. As discussed above, in theory up to $7 \times 5 = 35$ commands can be supported ($8 \times 5 = 40$ including the null configuration with no selectors), but for backward compatibly and ergonomics, the system provides access to 29 commands (Table 2). For example, the nd-index selector selects a shape creation command set with 4 commands (Figure 3-1b), index triggers the curve creation command (Figure 3-2a), middle triggers the rectangle creation command (Figure 3-2b), ring triggers the ellipse creation command (Figure 3-2c), and little triggers the text-box creation command (Figure 3-2d). The thumb does not trigger any command in this command set. Using similar mappings for the other 6 selector configurations, 25 different trigger commands are accessible in step 2.

Figure 3-2e shows a detailed view of a crib-sheet: feed-forward for trigger commands is displayed in the crib-sheet as 5 icons representing the mapped commands; blank icons appear for fingers not yet assigned with any command, such as the thumb here; icons are slightly shifted up and down to suggest the mappings between commands and the natural heights of fingers. The crib-sheet appears after 0.33s, a value similar to that used in marking menus [39]. Once a tool is selected with a finger trigger,

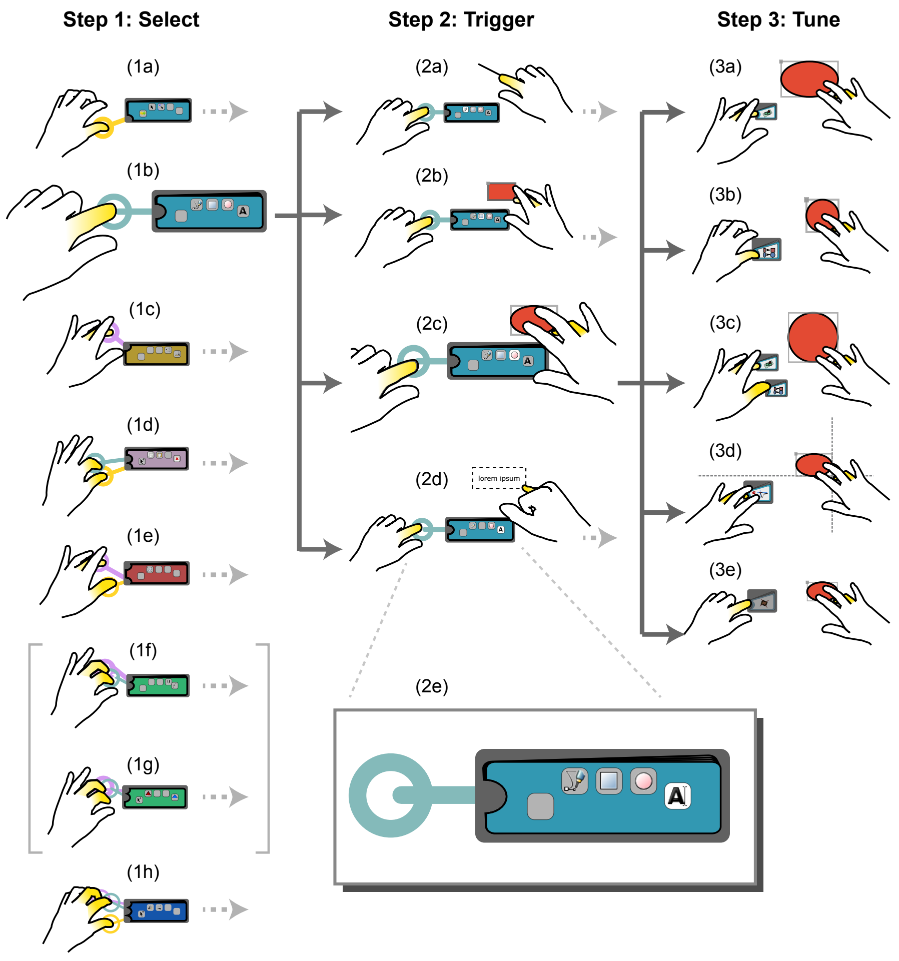

Figure 3: FingerCuts for the tabletop vector drawing demonstration. In step 1, the user performs a selector configuration to select a command set mapping: (1a) nd-thumb selector configuration (*Undo/Redo, Selection, Point edition* set); (1b) nd-index selector configuration (*Curve, Rectangle, Ellipse, Text box* set), (1c-h) other nd selector configurations (see Table 2). In step 2, the user invokes the command and manipulates its parameters using a trigger configuration while maintaining the selector configuration: for the nd-index selector, the user draws (2a) a curve with her index; (2b) a rectangle with her middle finger; (2c) an ellipse with her ring finger; or (2d) a text box with her little finger. When a command is triggered, the corresponding icon is highlighted in the crib-sheet (2e). In step 3, the user can optionally constrain the command parameter using a tuner configuration: when drawing an ellipse, the user can (3a) draw the ellipse from the center; (3b) draw a circle instead of an ellipse; (3c) or even combine the two previous constraints to draw a circle from the center; (3d) draw the ellipse with snapping enabled. When a constraint is not applicable the corresponding icon in the crib-sheet is greyed out: disabling the rescaling is available when performing RST gesture but not when drawing (3e).

Chord	thumb	index	middle	ring	little
○○○○○ ○○○○●	Undo/Redo	Selection	Point editing		
○○○●○		Curve	Rectangle	Ellipse	Text box
○○●○○	S+	Raise to top	Raise	Lower	Lower to bottom
○○○●●	S+	Copy	Cut	Paste	Delete
○○●○●	S+	Group	Ungroup		
○○●●○	S+	⟵ contextual commands ⟶			
○○●●●	S+	Vsym	Hsym		

Table 2: Command sets. Each row of the table is a command set. The non-dominant chord is represented as five circles for finger states (left is little, right is thumb) where ○ and ● represent which fingers are up and down respectively. The commands mapped to the dominant hand fingers are listed in the following columns where S+ represents the multi-selection mode.

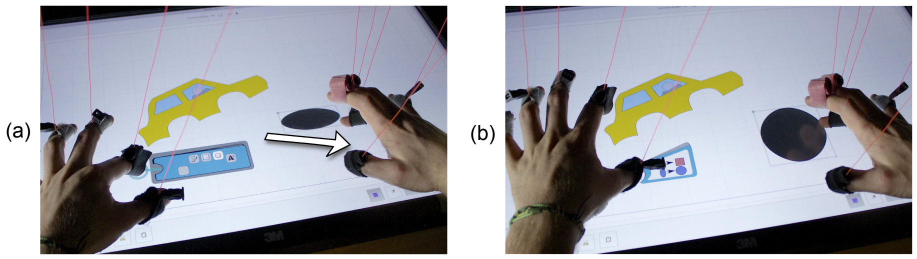

Figure 4: Example tabletop FingerCuts interaction: (a) nd-thumb and nd-index activate a command set in step 1, as ring triggers drawing ellipse in step 2; (b) nd-thumb tuner used to constrain aspect ratio of ellipse in step 3.

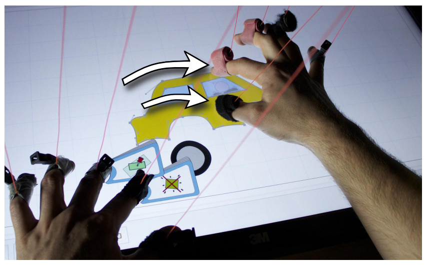

Figure 5: RST using thumb and index but constrained by the nd-index (no-scaling) and nd-middle (fixed-center) to only rotate the object.

visual feedback is provided on the crib-sheet (*e.g.* text box tool with the little finger in Figure 3-2e).

While manipulating command parameters, the user can remove their selector fingers and then use tuner fingers to modify how the command is interpreted. For example, while drawing ellipses (Figure 3-2c) the available tuning options are nd-middle to draw from the centre (Figure 3-3a), and nd-thumb to constrain the aspect ratio (Figure 3-3b and Figure 4-b). These can be combined by chording the corresponding fingers (*e.g.* Figure 3-3c) in an ad hoc tuner configuration. If a tuning option is not compatible with the current command, it is greyed out in the crib-sheet (Figure 3-3e).

In the default command set (first row in Table 2), certain dominant hand configurations are reserved for existing multitouch techniques to remain backward compatible. For example, the thumb+index and index+nd-index are mapped to RST gestures and the index is used to move shapes. Users already use these fingers for these standard manipulations in many applications. The FingerCuts technique further augments these standard gestures. Tuner fingers can optionally constrain RST actions, for example, to only rotate the object with or without a fixed centre (Figure 5). Other frequently used commands such as moving curve control points and undo/redo are mapped to the middle finger and directional thumb flicks respectively. Note that these latter commands are discoverable from the nd-thumb selector, which triggers a crib-sheet showing this default command set (first row of Table 2).

We also explored an interactive crib-sheet in the form of a colour chooser widget (see Figure 6). It follows the same 3-step pattern but the crib-sheet itself has an interactive element: a standard Hue, Lightness, Saturation (HLS) colour wheel selector. Once the colour chooser is opened with a nd-thumb + nd-little chord, the dominant hand can interact directly with the colour wheel portion of the crib-sheet. Dominant hand d-index and d-middle taps outside the crib-sheet apply the colour to object fill or stroke, and d-ring and d-little copy the object's fill or stroke colour to the colour wheel.

To further promote discoverability and learnability, a globally accessible help mode is available by tapping all ten fingers on the screen at the same time (Figure 7). When active, an interactive visualization of all commands is displayed where all finger combinations may be explored without affecting the working area.

5. Evaluation

We conducted two user studies and a Fingerstroke-Level Model (FLM) analysis to evaluate FingerCuts in the tabletop vector drawing application described above. The goal of the first study was primarily qualitative, with a focus on discoverability and novice-to-expert transition. The goal of the second study was quantitative, to compare task performance using FingerCuts against a standard touch graphical user interface (GUI). The GUI interface was based on existing tablet vector drawing applications where all commands could be accessed using always-visible palettes of buttons. Button icons were taken from the popular Inkscape desktop application. The FLM analysis further validates and investigates the results of the quantitative experiment. It also justifies FLM as a reasonable method for analytical performance analysis of FingerCuts with other tasks and device form factors.

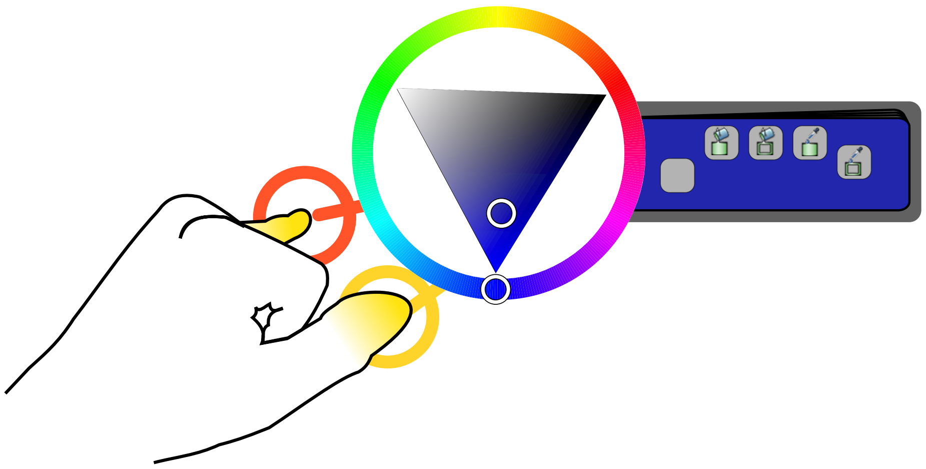

Figure 6: Example of an interactive crib-sheet. The nd-thumb + nd-little chord opens an interactive colour chooser widget with associated crib-sheet. The colour is adjusted with the dominant hand by dragging directly on the colour wheel. The colour is applied by tapping on objects using different dominant fingers: d-index sets object fill colour and d-middle sets object stroke colour. In addition, the d-ring and d-little act as "eyedroppers" to copy the fill or stroke colour of tapped objects to the colour wheel.

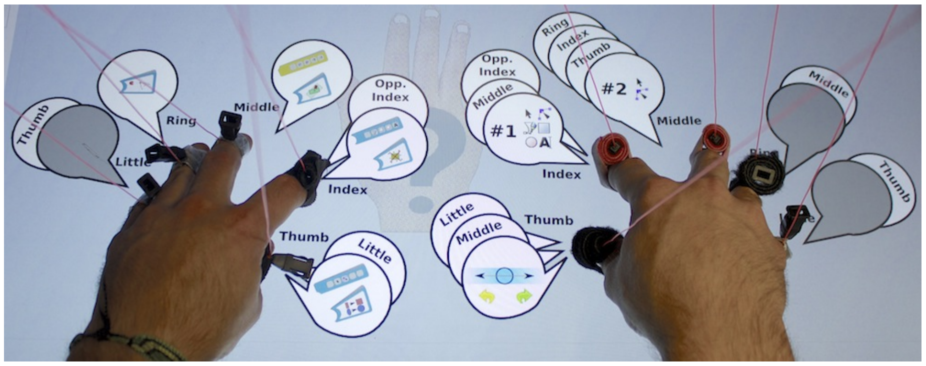

Figure 7: Global help mode accessed by tapping all ten fingers at the same time.

5.1. Qualitative Study

We recruited 8 volunteers (aged 20–35, 2 females, 2 non computer scientists). All participants used multi-touch devices daily, all used a desktop vector drawing application at least monthly (either Adobe Illustrator or Inkscape), and all were right-handed. We did not recruit for specific multi-touch experience, vector drawing experience, or handedness. After a 5-minute introduction to vector drawing applications and a basic explanation of finger identification, the experiment proceeded in two sequential parts.

In part one, participants were asked to freely explore the features of the application for 20 minutes. They were encouraged to draw simple shapes as well as more complex ones. While using the system, participants were asked to use the think-aloud protocol to externalize their thought process and reactions. Afterwards, the experimenter showed features that participants did not discover to prepare them for a second part of the experiment.

In part two, participants were asked to perform four simple manipulation tasks and one complete drawing task. Participants were free to use GUI widgets and/or FingerCuts to complete the tasks. All actions, including applying constraints, can be performed using either the standard GUI or FingerCuts. Each of the four manipulation tasks required a particular type of manipulation: (Task A) creating shapes with approximate dimensions and positions; (Task B) duplicating polygon shapes to create a sky with stars; (Task C) changing different shape colours; and (Task D) manipulating different shapes. For each task, the participant was presented with a starting screen (Figure 8-top) and prompted with a target image they had to reproduce (Figure 8-bottom). No instruction on accuracy nor completion time were mentioned. The tasks were presented in the same order to each participant. The last task was the most complex in that it required the creation of a simple scene from scratch. Participants were instructed to draw a house, with mandatory components as follows: at least one door with handle, two windows, and a triangular roof — with no further indication. No time limit was imposed to complete the part two but it took 30 minutes on average.

After each part of the study, participants were invited to take a break. At the end, they completed a short survey rating seven statements on a 5-point Likert-type scale (1=*Strongly disagree*; 5=*Strongly agree*): "The software was *close* to vectorial drawing applications I know"; "I was able to *accomplish* what I wanted to do"; "The shortcuts were easy to *learn*"; "The shortcuts were easy to *discover*"; "The shortcuts were easy to *use*"; "I *preferred* to use the classical Graphical User Interface over the provided shortcuts"; and "The *flow* of interaction was more continuous compared the classical GUI".

5.1.1. Results

Participants were able to complete both parts with minimal difficulty. For part two, 1 out of 8 participants used only the standard GUI, 3 participants mixed GUI and FingerCuts controls (more commands were performed with FingerCuts), and 4 participants used FingerCuts exclusively. For participants who mixed GUI and FingerCuts: one used the GUI for a command after a tracking system error and for all content delete actions; one used the GUI for most delete and undo actions, as well as some mode switching; the other used the GUI occasionally without a clear pattern. All participants discovered

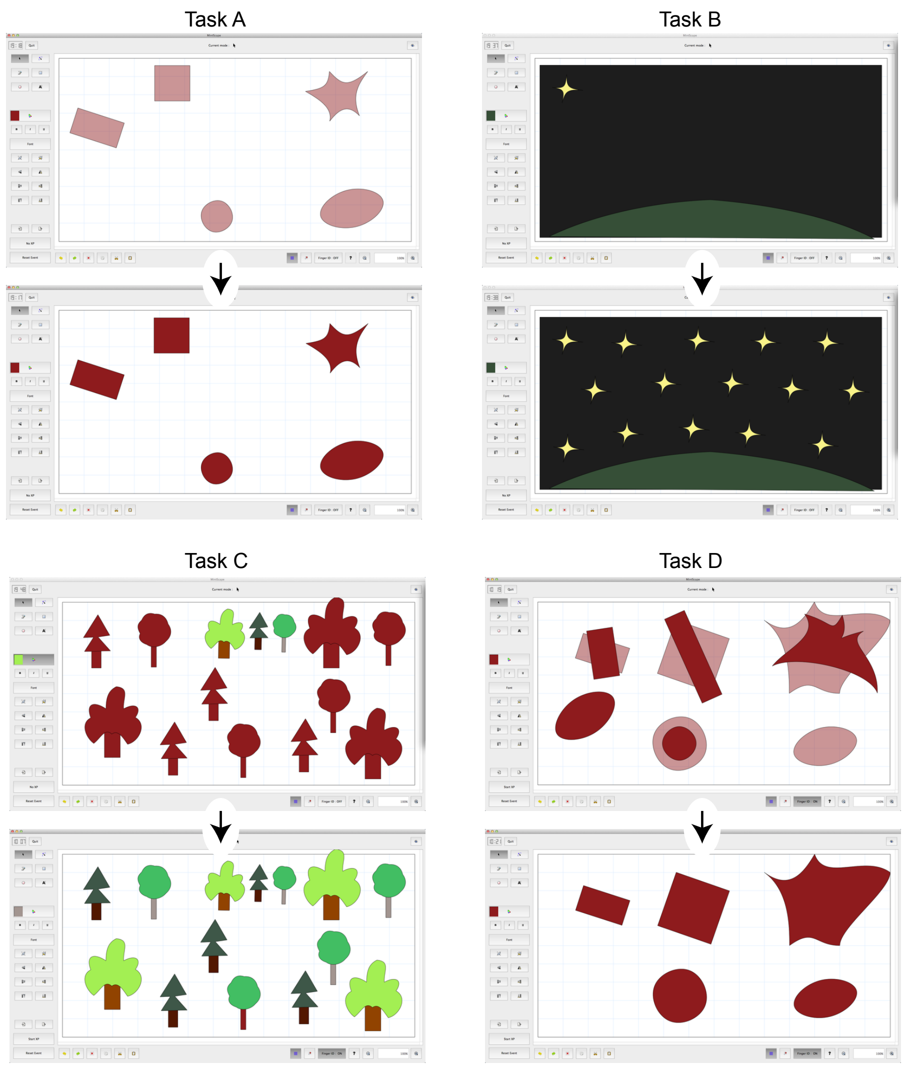

Figure 8: Manipulation tasks for part two of the qualitative study: Task A, creating shapes ; Task B, duplicating polygon shapes; Task C, changing shape colours; and Task D, manipulating shapes. Top screen of each task is the starting screen and bottom screen shows the target image given to participants.

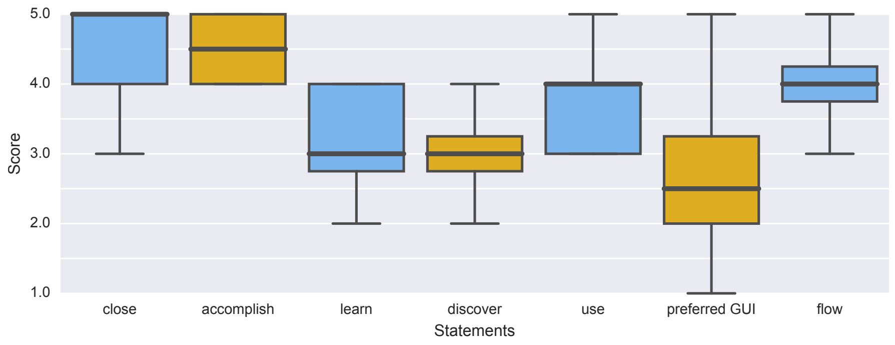

Figure 9: Overall ratings of seven statements in post-study survey. Each box shows the minimum value, first quartile, median, third quartile and maximum value.

the crib-sheet by themselves. However, we observed some participants initially pressed crib-sheet icons as if they were buttons. The affordances of the current feed-forward visualization could be improved by removing the icons' border or by blending them with the background. All participants also used the tuners to constrain the aspect ratio to create squares and circles. Most participants did not discover the contextual crib-sheets, likely because they require objects to be selected first. Few participants created text-boxes during the exploration part.

From the questionnaire, participants confirmed the *similarity* between our prototype application and vectorial applications they knew (mode 5, median 5) and they felt they successfully performed operations they wanted with FingerCuts (mode 4, median 4.5). This suggests that FingerCuts is expressive enough to provide a set of commands for moderately complex applications. The overall *ease-of-use* of FingerCuts was found to be positive (mode 4, median 4). Participants commented they needed time to *discover* (mode 3, median 3) and *learn* the different shortcuts (mode 3, median 3) but all participants acknowledged they would improve their performance with more practice. They also reported that the *flow* of interaction was more continuous compared to the classical GUI (mode 4, median 4). The latter result is further supported by the fact that participants *did not prefer the GUI* (mode 2, median 2.5). Although we were careful to be neutral when running the study, the results for flow and GUI preference could be affected by compliance or novelty bias since its clear FingerCuts is a new technique. However, even if some bias exist, positive scores suggest FingerCuts worked well relative to a standard GUI.

Like keyboard shortcuts, FingerCuts requires time and practice to become familiar and proficient. We were not surprised to observe that participants did not fully master FingerCuts after only one hour of usage, but participants indicated they were learning: P7 said "With practice, I would be quicker." We were pleased to see that participants were able to adopt the FingerCuts pattern and re-discover commands, P3 said "There are a lot of shortcuts and I did not remember them all but they are easy to retrieve." We also saw signs of emerging novice-to-expert transition, P1 commented "The shortcuts I used made me go quicker."

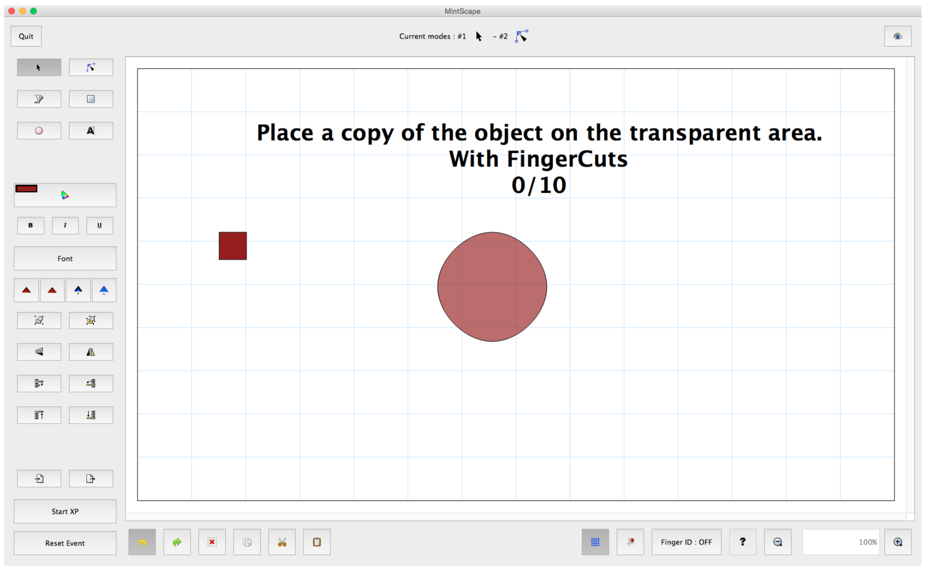
Place a copy of the object on the transparent area.
With FingerCuts
0/10

Figure 10: The GUI interface showing the Copy task.

5.2. *Quantitative Study*

We recruited 12 participants for the second study (aged 23–35, 1 female, 3 non computer scientists, all right-handed). A within subject design was used with independent variables for Technique with 2 levels (FingerCuts and a standard touch GUI as displayed on Figure 10), and Task with three levels: Copy — create a copy of an object and position the copy at a target location; Create — create a rectangle with opposite corners positioned at specific locations; and Edit — edit a triangle by moving one vertex to a specific location (Figure 11). Technique and Task were counter balanced across participants using a Latin square. Each Technique $\times$ Task was repeated 50 times over 5 blocks. Participants could take a break between blocks and they had to successfully perform a trial before moving to the next one. The task parameters (initial target position, size, ...) remained constant for each task. Participants were made aware of the optimal method in which to use both techniques. This was achieved by the experimenter systematically instructing participants on the fastest way to perform each task with each technique and encouraged them to perform each task as fast as possible. For example, for the GUI technique participants were encouraged to use their non dominant hand to press buttons on the left-side toolbar while they interacted with their right hand.

In summary, the experimental design is: 12 participants $\times$ 2 Techniques $\times$ 3 Tasks $\times$ 5 blocks $\times$ 10 trials = 3000 data points.

5.2.1. *Results*

The primary dependent variable is task completion time. We define this as the time between the first touch down event and the last touch up event during a single trial. As

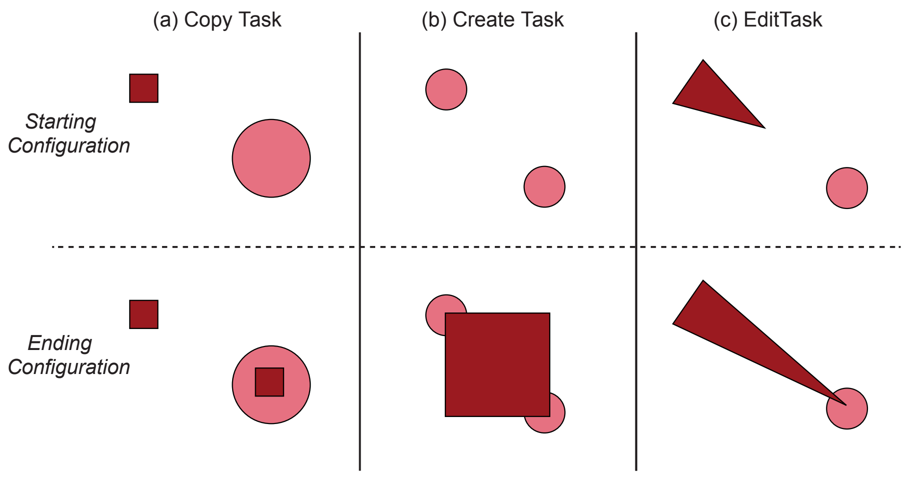

Figure 11: TASKS of the quantitative experiment. In the COPY task, users needed to duplicate the square and place the copy inside the circular area. In the CREATE task, users needed to draw a rectangle starting from one of the circular area and ending in the other one. In the EDIT task, users needed to move the bottom right vertex inside the circular area.

completion times exhibited a non-normal distribution, the experimental data was pre-processed using an Aligned Rank Transform [40] before running a repeated-measures ANOVA.

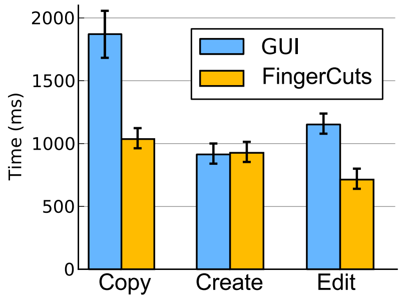

Figure 12: Mean task time for FingerCuts and standard GUI (error bars are 95% CI).

The analysis showed a significant main effect of TECHNIQUE ($F_{1,11} = 38.7$, $p < 0.001$) and a significant TECHNIQUE × TASK interaction ($F_{2,22} = 48.8$, $p < 0.001$) on completion time (Figure 12). Post-hoc analysis revealed significant differences between the two techniques for the COPY task ($p < 0.001$, GUI: 1.87s, FingerCuts: 1.03s) and the EDIT task ($p < 0.001$, GUI: 1.14s, FingerCuts: 0.71s) while the CREATE task did not show significant differences between techniques (GUI: 0.91s, FingerCuts: 0.92s). The significant differences observed for tasks COPY and EDIT can be explained by the additional actions to perform in the GUI condition compared to FingerCuts— pressing the copy and paste buttons as well as changing mode. While COPY and EDIT required multiple command mode changes, the CREATE task had only one mode change regardless of the technique used.

5.3. Fingerstroke-Level Model Analysis

We also model the tasks in the quantitative experiment using Lee *et al*.'s Fingerstroke-Level Model (FLM) [41] (which is an extension of the well-known Keystroke-Level Model (KLM) [42] to touch input). This reveals what actions are likely contributing to the differences in the observed overall times. Times predicted by FLM were found empirically by Lee *et al*. using mobile devices which have a different form factor than the tabletop used in our study. Therefore, we focus on the ratio between times of FingerCuts and times of the GUI. If the ratios predicted by FLM agree with those measured in the experiment, it justifies using FLM to predict relative performance for other FingerCuts tasks. We use the following set of FLM operators (with operator times as provided by Lee *et al*.):

M for *Mental thinking time* — the time to prepare the sequence of actions;

T for *Tapping* (0.31s) — the time to select a virtual target by tapping on it at a designated location;

P for *Pointing* (0.52s) — the time to point and select a virtual target by tapping on it;

D for *Dragging* (0.28s) —the time to translate a finger or chord from one location to another while remaining in contact with a touchscreen;

We extend FLM with one additional operator needed for our analysis:

C for *Chord* (0.31s) — the time to touch the screen with a chord *at a non-specific location* (new). Motivated by our experiment results, we use the approximation that C = T when chords are performed by experts[9].

5.3.1. Analysis

Table 3 summarizes the FLM analysis comparing FingerCuts to GUI for the three experiment tasks.

COPY *task.* Using the GUI, the sequence of actions is: mental preparation; tap to select the object to copy; tap the copy button; point the paste button; tap to select the copied object; drag the copied object at the target location (fingerstroke sequence: MTTPTD). Using FingerCuts, the sequence of actions is: mental preparation; perform the chord nd-thumb+nd-index; tap to copy the object with the d-index; point to paste the copied object at the specific location with the d-middle. (fingerstroke sequence: MCTP).

[9]Further empirical investigation is necessary to better asses the (theoretical) cost of each operation, but a possible explanation for the lack of difference we observed between Tap and Chord can be that the cost of the complexity of combining several fingers to perform a chord anywhere on the display is compensated by the distance traveled to aim at the target.

Task	GUI	FingerCuts	Difference (FingerCuts - GUI)	Time Ratio Experiment	Time Ratio FLM
COPY	(M)TTPTD	(M)CTP	+ C - TTD	0.55	0.66
CREATE	(M)TTD	(M)CTD	+ C - T	1.01	1.00
EDIT	(M)TTD	(M)TD	- T	0.62	0.66

Table 3: Comparative Fingerstroke-Level Model analysis between GUI and FingerCuts. The time ratio columns are FingerCuts over GUI for measured experiment times and FLM predicted times.

CREATE *task.* Using the GUI, the sequence of actions is: mental preparation; tap to select the rectangle tool; tap the starting area; drag to the ending area (fingerstroke sequence: MTTD). Using FingerCuts, the sequence of actions is: mental preparation; perform the chord nd-index; tap the starting area with the d-middle; drag to the ending area (fingerstroke sequence: MCTD).

EDIT *task.* Using the GUI, the sequence of actions is: mental preparation; tap to select the point edition tool; tap the bottom right vertex; drag to the ending area (fingerstroke sequence: MTTD). Using FingerCuts the sequence of actions is: mental preparation; tap the bottom right vertex with the d-middle; drag to the ending area (fingerstroke sequence: MTD).

5.3.2. Discussion

Since participants were trained, we assume mental preparation time is negligible compared to motor action time. Predicted times and empirical ones differed, but we observed a correlation between the ratios of FingerCuts and GUI times when both empirically measured and FLM predicted (two rightmost columns of Table 3). We hypothesis that the absolute difference between predicted and empirically found times is due to the difference of form factors (*e.g.* mobile devices [41] vs. tabletop).

Although this is not a comprehensive test, we argue it does justify using FLM as a theoretical model to compare FingerCuts and GUI task performance.

6. Other Applications and Device Form Factors

To demonstrate adaptability of FingerCuts to other device form factors and applications, we created a PDF document annotation application on a tablet and a SMS text messaging application on a smartphone (Section 6).

Considering the small device form factors and type of applications, we assigned all selectors, triggers, and tuners to the same fingers on both hands. Wagner *et al.*[38] showed that the non-dominant thumb can be used to interact even as the non-dominant hand supports the device. We exploit this by using the thumb of either hand as a single selector configuration (with directional thumb movement used to select different command sets, explained in detail below). This means that the non-dominant thumb may be used to select a command set while the same hand holds the device. Since all mappings are mirrored on both hands, finger names are given without hand for simplicity.

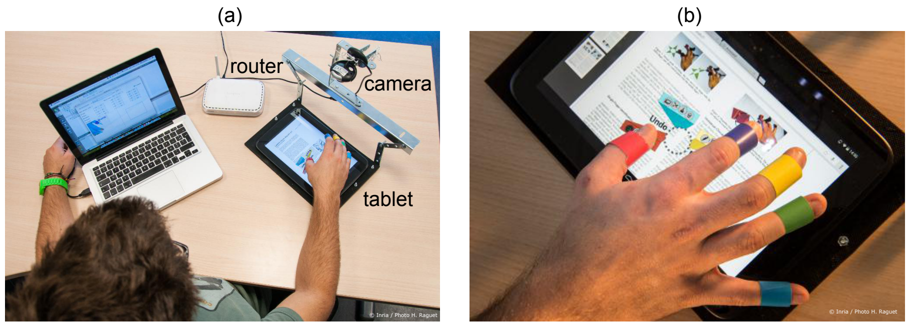

Figure 13: Finger identification technique used for the tablet and smartphone: (a) a camera is mounted on a frame fixed to the tablet or smartphone; (b) coloured rings are attached near each finger tip.

6.1. Finger Identification Technique

The GameTrak-based finger identification technique used for the tabletop assumes a fixed device, which is not the case for the tablet and smartphone. Instead, we attach different coloured rings to each finger and track the 2D positions using a 640×480px RGB camera located above the surface (Figure 13) similar to Wang and Canny's method [27]. This technique can reliably track up to 6 fingers. We track all 5 fingers of the dominant hand and the non-dominant thumb.

Like the tabletop tracking system, the 3D tracked finger positions are transformed to the 2D reference frame of our 10" tablet or 4" smartphone using a homography calculated from a short calibration procedure. This finger identification software sends the same custom TUIO events annotated with hand and finger IDs to the touch device application running on the tablet or smartphone. Our tracking software uses the OpenCV library and runs at approximately 20Hz.

6.2. Tablet: PDF Annotation Application

PDF viewing and annotation applications like Adobe Acrobat require commands to navigate the document and commands to annotate with different styles of ink, stamps, and text notes. Our PDF application supports 18 commands, 15 commands are organized into four command sets (including a default one) and the remaining 3 are through flicks. Either the dominant or non-dominant thumb selects which commands are mapped to the index, middle, ring and little trigger fingers. With no thumb touch, a default command set is active where the index finger scrolls (Figure 15-a), the middle finger is a pen highlighter (Figure 15-b), and the ring finger creates text annotation notes. In addition, a simultaneous thumb and index chord zooms using that standard pinch gesture. The other three command sets are selected by dragging a thumb in one of three directions (Figure 14-a). When one of these command sets are selected, a crib-sheet is displayed around the thumb showing the four commands mapped to the index, middle, ring and little trigger fingers (Figure 14-b). The three command sets are colour coded: (blue) for copy, paste, cut and delete; (yellow) for customizable pens; and (red) for customizable

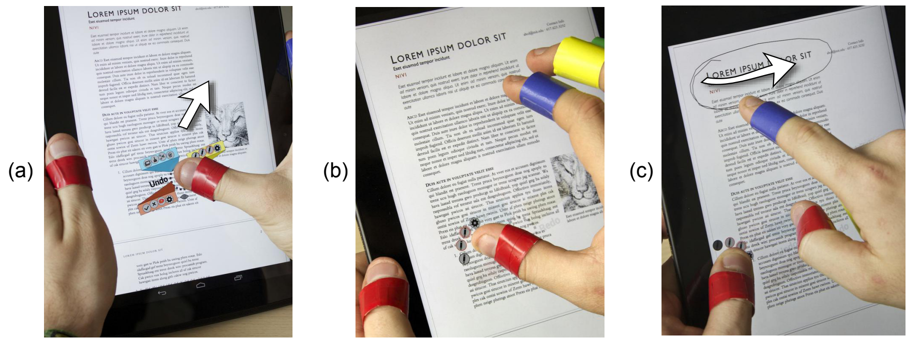

Figure 14: Thumb selector with tablet: (a) thumb selector showing the crib-sheet with three command sets; (b) using the top-right set after sliding the thumb to the top-right to select a command set, commands associated to trigger fingers appear next to the thumb; (c) the thumb selector can also be used with the non-dominant hand to relax use of the dominant finger.

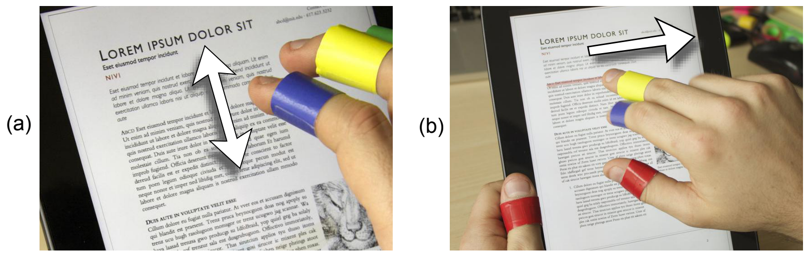

Figure 15: Global trigger finger examples: (a) index scrolls the document; (a) middle finger used as a pen highlighter.

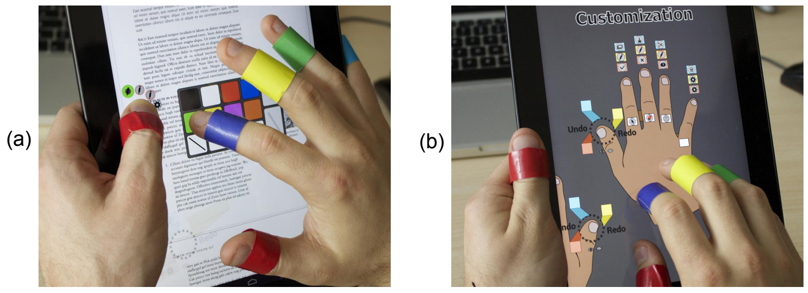

Figure 16: Other features: (a) customization of the pen associated to the index; (b) five finger chord entering the customization mode (commands can then be remapped with drag and drop of the associated icons).

stamps. When any pen command is triggered, the thumb functions a single tuner to constrain the ink to follow text lines (Figure 14-c).

The pen and stamp command sets include a customization mode are triggered with the little finger. In these modes, the colour, width, size, opacity, and shape of the three pens or stamps are customized by tapping the corresponding pen or stamp finger on a predefined colour, line width, and line opacity (Figure 16-a).

There are also globally accessible commands. Left and right horizontal thumb flicks trigger undo and redo, and downward thumb flick toggles a page thumbnail side bar. These thumb movement directions do not conflict with selecting among the three command sets. Standard global commands also enable scrolling and zooming. Finally, a five finger tap invokes a help screen, visually depicting all the command to finger associated to fingers (Figure 16-b). For personalization, commands can be re-mapped by dragging a command icon from one crib-sheet to another to swap them.

6.2.1. Fingerstroke-Level Model Analysis

We use the FLM operators defined in the previous section to compare FingerCuts to a traditional user interface with the addition of one additional operator defined by Lee *et al*. [41]:

F for *Flicking* (0.11s) — the time to perform a quick and short drag;

Consider a simple but common task: scroll a document to page 2, highlight a sentence, and add a text note. In Adobe Acrobat 15.2.2 on Nexus 10, the sequence of actions is: mental preparation; tap the first page; drag to the second page; point to select the highlighter mode; point at the location at the beginning of the sentence; drag to highlight the sentence; point to select the text box mode; point at the location of the note and drag to define the dimensions of the box. The corresponding fingerstroke sequence is MTDPPDPPD. Using FingerCuts, the sequence of actions is: mental preparation; tap the first page with the index finger; drag to the second page with index finger; point at the location of the beginning of the sentence with the middle finger; drag to highlight the sentence; point at the location of the note with ring finger and drag it to adjust the dimension. The corresponding fingerstroke sequence is MTDPDPD. In this example, FingerCuts eliminates one P per tool switch, two P in total.

Consider another simple task: highlighting a paragraph with three different colours. In Adobe Acrobat 15.2.2 on Nexus 10, the sequence of actions is: mental preparation; point to select the highlighter mode; point at the location at the beginning of the sentence; drag to highlight the sentence; optionally tap the stroke; tap the colour item on the contextual menu; tap the wanted colour. Only one colour at a time can be selected. The corresponding fingerstroke sequence for highlighting twice in yellow, once in red, and once in green is M-PPD-PPD-PPDTTT-PPDTTT. Using FingerCuts, the sequence of actions is: mental preparation; flick with the nd-thumb; point at the location of the beginning of the sentence with the index/middle/ring finger; drag to highlight the sentence. The corresponding fingerstroke sequence is M-F-PD-PD-PD-PD. In this example FingerCuts eliminates one P per highlight and the need to reselect colours (three T per colour selection). The is a reduction of four P and six T in total.

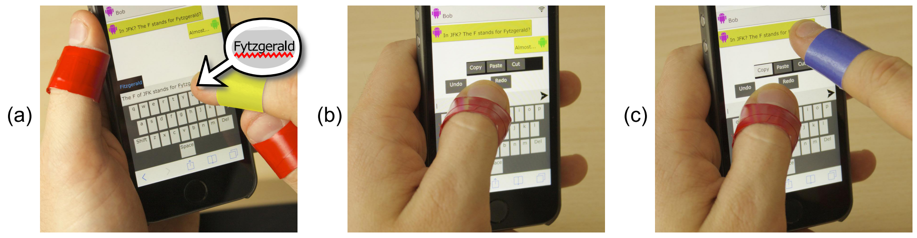

Figure 17: Smartphone selectors: (a) middle correcting a word; (b) thumb used as a selector and showing the crib sheet; (c) index triggering *copy*

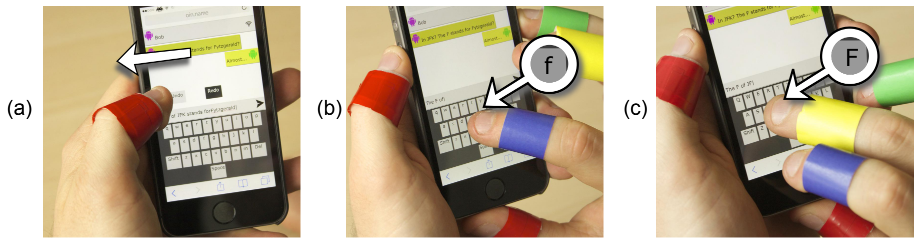

Figure 18: Smartphone triggers: (a) right swipe with thumb triggering undo; (b) index triggering lowercase key; (c) middle finger triggering uppercase key.

6.3. SmartPhone: SMS Application

Writing a SMS text message requires commands to enter the text, correct spelling, copy, cut, paste, and navigate message history. Our smartphone SMS application uses either thumb as a single selector, similar to the tablet. When the thumb is contacting, a crib-sheet is displayed with the command mapping for the four fingers shown as adjacent rectangles trigger (Figure 17-b). Tapping the index, middle, and ring fingers on single words triggers copy, paste, and cut commands respectively (Figure 17-c). Although precise text selection is not implemented, dragging the index finger without the thumb touching could select alternate text ranges to cut, copy, or paste over. When the thumb is not contacting, a default command set is used. The index finger scrolls the message history and tapping a word with the middle finger opens a menu of correct spellings (Figure 17-a). Undo and redo are supported with the same left and right thumb flicks (Figure 18-a). When typing, lower case letters are entered with the index finger (Figure 18-b) and upper case letters with the middle finger (Figure 18-c). The ring finger could also be used to access punctuation when typing. Supporting thumb typing could be achieved by implementing a kind of quasi-mode to differentiate thumb swipes on the message and thumb taps on the keyboard: we discuss this current limitation in the Discussion section.

6.3.1. Fingerstroke-Level Model Analysis

We use FLM to compare the performance of FingerCuts to a standard interface in a simple task. The *Messaging* app on Android 5.1.1 is our reference for a standard interface. Like many smartphone apps, commands like cut, copy, and paste are accessed from a contextual menu revealed after a long press. This necessitates a new FLM operator:

L for *Long press* (between 0.3s and 0.5s) — the dwell time needed to activate a long press (new). Time range is derived from 0.333s used in Marking Menus [36] and 0.5s as the default long press time in Android;

Consider a simple, but common task: while typing a new message, the user selects a word in a previous message from the same conversation thread (*e.g.* the name of a restaurant), then they copy and paste that word into the message being composed. The sequence of operations in our standard reference interface is: mental preparation; tap the word to copy; wait for a long press; tap the copy button; tap at the location where the word needs to be copied; wait for a long press; tap the paste button. The corresponding fingerstroke sequence is M-TLT-TLT. Using FingerCuts the sequence of operations is: mental preparation; chord with the thumb to select the command set; tap the word to copy with the index finger; tap at the location where the word needs to be copied with the middle finger. The corresponding fingerstroke sequence is M-CT-T. FingerCuts reduces the number of operations by 2 $\times$ LT actions for the cost of one C operations.

For another task example, consider writing a short sentence with an acronym and punctuation, like the message: *"The FLM model :)"*. The sequence of operations in our standard reference interface is: mental preparation; tap the *"T"* key; point at the *"h"*, *"e"*, *"space"*, *"Shift"*, *"F"* keys and so on; point at the *"symbol"*, *":"* and *")"* keys. The corresponding fingerstroke sequence is M-TPP-P-PPPPPP-P-PPPPP-P-PPP. Using FingerCuts, the sequence of actions is: mental preparation; tap the *"T"* key with the middle finger; point at the *"h"*, *"e"* and *"space"* with the index; point at the *"F"*, *"L"* and *"M"* with the middle finger and so on; point at the *":"* and *")"* keys with the ring finger. The corresponding fingerstroke sequence is M-TPP-P-PPP-P-PPPPP-P-PP. In this example FingerCuts eliminates all the P corresponding to keyboard modifiers keys (three *"Shift"* keys and one *"symbol"* key). Four P operations are eliminated in total.

7. Discussion

Motivated by our three FingerCuts demonstration applications, we summarize key points, provide design guidelines and recommendations, and acknowledge current or inherent limitations.

7.1. Device ergonomics

Device ergonomics play a key role in the number of available fingers to input chords. Smartphones and tablets are typically held with the non-dominant hand, restricting the interaction with screen content to the dominant hand. However the non-dominant thumb usually remains available to interact with the screen [38]. Our SMS and PDF annotation applications use either thumb as a selector to support this capability and provide flexibility for how the selector and trigger are performed. A single selector configuration was enough for the SMS application and in the PDF application we illustrated a strategy using finger movements to enable more than one selector action to be mapped to a single configuration. In contrast, tabletops are larger and typically support applications with more commands. In that context, our vector drawing demonstration mapped finger selector to the non-dominant hand and the finger trigger to the dominant hand.

7.2. Mapping consistency

In designing the demonstration applications, we balanced backward compatibility with existing finger counting techniques and consistency across devices and applications to optimize skill transfer. For example, undo and redo are consistently accessed using left and right thumb flicks. In the PDF and vector drawing demonstrations, we reserved the thumb and index chord for zooming and object transformation. We also consistently use the index finger is to drag objects or scroll the document. However, these decision are assuming that the thumb and index is the most common chord for zooming and transformation and index fingers are used to scroll. Further studies are needed to clarify which fingers are actually used to perform standard finger counting interactions.

When we could not be completely consistent across devices, we use the same pattern. For example, although copy, paste, and cut are accessed with different selector chords on different devices, the index always copies, the middle always pastes, and ring always cuts.

7.3. Choosing command-to-finger mappings

To combine command selection and parameter manipulation, one first needs to know what parameters to control and how they can be controlled. Our demonstrations focus on the continuous control of a single point position, which is best achieved using a single finger. However other parameters may require additional degrees of freedom, involving additional finger triggers (*e.g.* RST is performed using two-finger pinch and rotate gestures). Yet, using only one single finger trigger — with arbitrary number of finger selectors — solves the problem of temporal ambiguity since no time window is needed to recognize a chord. As a result, commands are executed with no delay. In addition there is no spatial ambiguity since the user can clearly expect the action will occur beneath her finger trigger.

During the development of our demonstrations, we informally observed the index and middle fingers provide better accuracy and less occlusion than the ring and little fingers. This appears to be due to how the the other fingers can be tucked with index and middle, and the awkwardness of pressing with ring and little fingers. Frequent and precise commands should be assigned to higher performing fingers. For example in the PDF application, the index is used for frequent scrolling and copying as well as precise pen control. There is currently little empirical evidence for performance differences between fingers. Recent work by Colley *et al*. [43] provides results for dominant hand pointing, but there is little or no work examining performance differences for the non-dominant hand or for completing more complex tasks like dragging, scaling, and rotating with different fingers or chords.

Commands requiring no parameter, like “save”, can combine different finger trigger as they do not require any specific interest point. In addition chords that are more difficult to execute can be associated with more risky commands. For instance, in the drawing application, the delete command is associated to the nd-thumb+nd-index+little chord.

7.4. Efficiency of merging command selection and parameter manipulation

In the comparison of three techniques merging command selection and direct manipulation, Guimbretière *et al.*state that the determinant factor affecting the performance of all the techniques lies in the “fluidly” of mixing command selection and direct manipulation [14]. This is consistent with the results of our user study showing that FingerCuts can be faster than traditional user interfaces for canonical drawing tasks. Although participants could use both hands to interact with the traditional GUI, the merging of command selection and parameter manipulation offered by FingerCuts yielded less actions resulting in shorter completion times. Our experiment targeted expert use through many repetitions of the same tasks. We expect that the three-step interaction sequence can help novices quickly gain expertise and become proficient users.

The FLM analysis support this reduction of time consuming actions, especially when mode switching is required. FLM analysis provides a first validation step for generalizing the outcome of the first tabletop studies, but further evaluations need to be conducted to empirically quantify the actual time of each action in various contexts.

7.5. Limitations

An initial limitation with FingerCuts is that new users need to discover which fingers or chords are selectors. Once this is known, they can discover all commands and triggers by trying different selectors and examining the crib-sheet (no command is issued until a trigger is used). Although there is an initial barrier, our initial user study showed people can discover and use selectors with minimal instructions. It also highlighted that the feed-forward needs to be carefully designed to avoid interpreting the crib-sheet as a tool pallet of buttons.

Since selectors do not require a location, they enable users to focus on the trigger. A drawback is that the selector configurations can not also be used to interact with the widgets. For example in the current SMS application, users can no longer type with two thumbs. One way to alleviate this problem is to interpret selectors configurations as direct interaction based on contact time and simultaneous triggers. For example, a short thumb press without a simultaneous trigger finger means type a key, but a longer press displays the crib-sheet.

All chords that include a finger selector benefit from feed-forward. However, by definition, the crib-sheet cannot be used for chords using only finger triggers. For example, one can consider using index+middle for relative scrolling and middle+ring for absolute scrolling in a document. Yet, if the index, middle and ring are all finger triggers, we have no way of informing the user that such action is possible. The same problem applies for single fingers but one can argue that the number of such commands is limited so the memorizing cost remains low. A help mode could be designed to discover such commands.

Our work assumes robust finger identification sensing on multi-touch interfaces. If early commercial prototypes do not provide perfect identification accuracy, feed-forward and feedback provide some tolerance for finger misidentification and explicit correction. For example, if the correct crib-sheet does not appear, the user may re-adjust their posture slightly to force re-identification.

Finally, FingerCuts assumes an explicit handedness configuration when the two hands play different roles (as in the tabletop drawing application). A button or gesture could toggle left and right handedness.

8. Conclusion

We introduced FingerCuts, an interaction technique leveraging finger identification for integrated command selection and parameter manipulation. Considering the large input space offered by finger identification and the associated discoverability problems, FingerCuts defines a three-step pattern using finger trigger to trigger commands, finger selector to change the mapping of commands to finger trigger, and finger tuners to alter the ongoing execution of a command. This three-step interaction helps novices by introducing feed-forward and feedback in a crib-sheet, while enabling experts to perform actions simultaneously and rapidly. We illustrated FingerCuts with applications in three different contexts demonstrating how to maintain backward compatibility with existing multi-touch techniques and guide designers when instantiating the interaction technique on different devices form factors. The results of a preliminary user study and Fingerstroke-Level Model analysis show that people are able to use finger identification to integrate command selection with direct manipulation, and they are able to outperform traditional interfaces when tasks require frequent mode switching.

9. References

[1] E. L. Hutchins, J. D. Hollan, D. A. Norman, Direct manipulation interfaces, Human-Computer Interaction 1 (1985) 311–338. URL: http://dx.doi.org/10.1207/s15327051hci0104_2.

[2] M. Beaudouin-Lafon, Instrumental interaction: an interaction model for designing post-WIMP user interfaces, in: Proc. CHI, ACM, 2000, pp. 446–453. URL: http://doi.acm.org/10.1145/332040.332473.

[3] J. Wagner, E. Lecolinet, T. Selker, Multi-finger chords for hand-held tablets: Recognizable and memorable, in: Proc. CHI, ACM, 2014, pp. 2883–2892. URL: http://doi.acm.org/10.1145/2556288.2556958.

[4] T. Grossman, P. Dragicevic, R. Balakrishnan, Strategies for accelerating on-line learning of hotkeys, in: Proc. CHI, ACM, 2007, pp. 1591–1600. URL: http://doi.acm.org/10.1145/1240624.1240865.

[5] K. Kin, B. Hartmann, M. Agrawala, Two-handed marking menus for multitouch devices, ACM ToCHI 18 (2011) 16:1–16:23. URL: http://doi.acm.org/10.1145/1993060.1993066.

[6] M. Serrano, E. Lecolinet, Y. Guiard, Bezel-tap gestures: quick activation of commands from sleep mode on tablets, in: Proc. CHI, 2013, pp. 3027–3036. URL: http://doi.acm.org/10.1145/2470654.2481421.

[7] E. Ghomi, G. Faure, S. Huot, O. Chapuis, M. Beaudouin-Lafon, Using rhythmic patterns as an input method, in: Proc. CHI, ACM, 2012, pp. 1253–1262. URL: http://doi.acm.org/10.1145/2207676.2208579.

[8] F. Wang, X. Ren, Empirical evaluation for finger input properties in multi-touch interaction, in: Proc. CHI, ACM, 2009, pp. 1063–1072. URL: http://doi.acm.org/10.1145/1518701.1518864.

[9] C. Harrison, J. Schwarz, S. E. Hudson, TapSense: enhancing finger interaction on touch surfaces, in: Proc. UIST, ACM, 2011, pp. 627–636. URL: http://doi.acm.org/10.1145/2047196.2047279.

[10] Q. Roy, Y. Guiard, G. Bailly, E. Lecolinet, O. Rioul, Glass+skin: An empirical evaluation of the added value of finger identification to basic single-touch interaction on touch screens, in: Proc INTERACT, Springer, 2015, p. to be published.

[11] C. Holz, P. Baudisch, Fiberio: a touchscreen that senses fingerprints, in: Proc. UIST, ACM, 2013, pp. 41–50. URL: http://dl.acm.org/citation.cfm?id=2501988.2502021.

[12] H. Benko, T. S. Saponas, D. Morris, D. Tan, Enhancing input on and above the interactive surface with muscle sensing, in: Proc. ITS, ACM, 2009, pp. 93–100. URL: http://doi.acm.org/10.1145/1731903.1731924.

[13] N. Marquardt, J. Kiemer, D. Ledo, S. Boring, S. Greenberg, Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit, in: Proc. ITS, ACM, 2011, pp. 21–30. URL: http://doi.acm.org/10.1145/2076354.2076358.

[14] F. Guimbretière, A. Martin, T. Winograd, Benefits of merging command selection and direct manipulation, ACM ToCHI 12 (2005) 460–476. URL: http://doi.acm.org/10.1145/1096737.1096742.

[15] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, T. D. DeRose, Toolglass and magic lenses: the see-through interface, in: Proc. SIGGRAPH, ACM, 1993, pp. 73–80. URL: http://doi.acm.org/10.1145/166117.166126.

[16] S. A. G. Wensveen, J. P. Djajadiningrat, C. J. Overbeeke, Interaction frogger: a design framework to couple action and function through feedback and feedforward, in: Proc. DIS, ACM, 2004, pp. 177–184. URL: http://doi.acm.org/10.1145/1013115.1013140.

[17] W. Buxton, Chunking and phrasing and the design of human-computer dialogues, in: Proc. IFIP World Computer Congress, 1986, pp. 475–480. URL: http://www.dgp.toronto.edu/OTP/papers/bill.buxton/chunking.html.

[18] J. Raskin, 3-2-3 Modes and Quasimodes, in: The humane interface: new directions for designing interactive systems, Addison-Wesley, 2000, pp. 55–59. URL: http://en.wikipedia.org/wiki/The_Humane_Interface.

[19] X. A. Chen, T. Grossman, D. J. Wigdor, G. Fitzmaurice, Duet: Exploring joint interactions on a smart phone and a smart watch, in: Proc. CHI, ACM, 2014, pp. 159–168. URL: http://doi.acm.org/10.1145/2556288.2556955.

[20] S. Houben, N. Marquardt, Watchconnect: A toolkit for prototyping smartwatch-centric cross-device applications, in: Proc. CHI, ACM, 2015, pp. 1247–1256. URL: http://doi.acm.org/10.1145/2702123.2702215.

[21] O. K.-C. Au, C.-L. Tai, Multitouch finger registration and its applications, in: Proc. OZCHI, ACM, 2010, pp. 41–48. URL: http://doi.acm.org/10.1145/1952222.1952233.

[22] P. Ewerling, A. Kulik, B. Froehlich, Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions, in: Proc. ITS, ACM, 2012, pp. 173–182. URL: http://doi.acm.org/10.1145/2396636.2396663.

[23] S. Murugappan, Vinayak, N. Elmqvist, K. Ramani, Extended multitouch: recovering touch posture and differentiating users using a depth camera, in: Proc. UIST, ACM, 2012, pp. 487–496. URL: http://doi.acm.org/10.1145/2380116.2380177.

[24] W. Westerman, Hand tracking, finger identification, and chordic manipulation on a multi-touch surface, Ph.D. thesis, University of Delaware, 1999. URL: http://www.eecis.udel.edu/~westerma/main.pdf.

[25] G. J. Lepinski, T. Grossman, G. Fitzmaurice, The design and evaluation of multitouch marking menus, in: Proc. CHI, ACM, 2010, pp. 2233–2242. URL: http://doi.acm.org/10.1145/1753326.1753663.

[26] S. Malik, A. Ranjan, R. Balakrishnan, Interacting with large displays from a distance with vision-tracked multi-finger gestural input, in: Proc. UIST, ACM, 2005, pp. 43–52. URL: http://doi.acm.org/10.1145/1095034.1095042.

[27] J. Wang, J. Canny, FingerSense: augmenting expressiveness to physical pushing button by fingertip identification, in: Proc. CHI EA, ACM, 2004, pp. 1267–1270. URL: http://doi.acm.org/10.1145/985921.986040.

[28] A. Sugiura, Y. Koseki, A user interface using fingerprint recognition: holding commands and data objects on fingers, in: Proc. UIST, ACM, 1998, pp. 71–79. URL: http://doi.acm.org/10.1145/288392.288575.

[29] K. Vega, H. Fuks, Beauty tech nails: Interactive technology at your fingertips, in: Proc. TEI, ACM, 2013, pp. 61–64. URL: http://doi.acm.org/10.1145/2540930.2540961.

[30] G. Bailly, E. Lecolinet, Y. Guiard, Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces, in: Proc. CHI, ACM, 2010, pp. 591–594. URL: http://doi.acm.org/10.1145/1753326.1753414.

[31] C. Gutwin, A. Cockburn, J. Scarr, S. Malacria, S. Olson, Faster command selection on tablets with FastTap, in: Proc. CHI, ACM, 2014, p. to appear. URL: http://www.malacria.fr/pdf/gutwin14-fasttap.pdf.

[32] M. S. Uddin, C. Gutwin, B. Lafreniere, Handmark menus: Rapid command selection and large command sets on multi-touch displays, in: Proc. CHI, ACM, 2016, pp. 5836–5848. URL: http://doi.acm.org/10.1145/2858036.2858211.

[33] E. Ghomi, S. Huot, O. Bau, M. Beaudouin-Lafon, W. E. Mackay, Arpège: Learning multitouch chord gestures vocabularies, in: Proc. ITS, ACM, 2013, pp. 209–218. URL: http://doi.acm.org/10.1145/2512349.2512795.

[34] J. Vermeulen, K. Luyten, E. van den Hoven, K. Coninx, Crossing the Bridge over Norman's Gulf of Execution: Revealing Feedforward's True Identity, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13, ACM, New York, NY, USA, 2013, pp. 1931–1940. URL: http://doi.acm.org/10.1145/2470654.2466255. doi:doi:10.1145/2470654.2466255.

[35] S. Malacria, G. Bailly, J. Harrison, A. Cockburn, C. Gutwin, Promoting hotkey use through rehearsal with ExposeHK, in: Proc. CHI, ACM, 2013, pp. 573–582. URL: http://doi.acm.org/10.1145/2470654.2470735.

[36] G. Kurtenbach, W. Buxton, Issues in combining marking and direct manipulation techniques, in: Proc. CHI, ACM, 1991, pp. 137–144. URL: http://doi.acm.org/10.1145/120782.120797.

[37] D. Vogel, G. Casiez, Hand occlusion on a multi-touch tabletop, in: Proc. CHI, ACM, 2012, pp. 2307–2316. URL: http://doi.acm.org/10.1145/2207676.2208390.

[38] J. Wagner, S. Huot, W. Mackay, Bitouch and bipad: designing bimanual interaction for hand-held tablets, in: Proc. CHI, ACM, 2012, pp. 2317–2326. URL: http://doi.acm.org/10.1145/2207676.2208391.

[39] G. Kurtenbach, W. Buxton, User learning and performance with marking menus, in: Proc. CHI, ACM, 1994, pp. 258–264. URL: http://doi.acm.org/10.1145/191666.191759.

[40] J. O. Wobbrock, L. Findlater, D. Gergle, J. J. Higgins, The aligned rank transform for nonparametric factorial analyses using only anova procedures, in: Proc. CHI, ACM, 2011, pp. 143–146. URL: http://doi.acm.org/10.1145/1978942.1978963.

[41] A. Lee, K. Song, H. B. Ryu, J. Kim, G. Kwon, Fingerstroke time estimates for touchscreen-based mobile gaming interaction, Human Movement Science 44 (2015) 211 – 224. URL: http://www.sciencedirect.com/science/article/pii/S0167945715300373.

[42] S. K. Card, T. P. Moran, A. Newell, The keystroke-level model for user performance time with interactive systems, Communications of the ACM 23 (1980) 396–410.

[43] A. Colley, J. Häkkilä, Exploring finger specific touch screen interaction for mobile phone user interfaces, in: Proc. OZCHI, ACM, 2014, pp. 539–548. URL: http://doi.acm.org/10.1145/2686612.2686699.