PinchList: Leveraging Pinch Gestures for Hierarchical List Navigation on Smartphones

Teng Han*[1], Jie Liu*[2], Khalad Hasan[3], Mingming Fan[4], Junhyeok Kim[1], Jiannan Li[4], Xiangmin Fan[2], Feng Tian†[2], Edward Lank[5], Pourang Irani[1]

[1]University of Manitoba [2]Institute of Software, Chinese Academy of Science [3]University of British Columbia - Okanagan [4]University of Toronto [5]University of Waterloo

{hanteng, kimj3415, pourang.irani}@cs.manitoba.ca, {liujie2016, xiangmin, tianfeng}@iscas.ac.cn, khalad.hasan@ubc.ca, {mfan, jiannanli}@cs.toronto.edu, lank@uwaterloo.ca

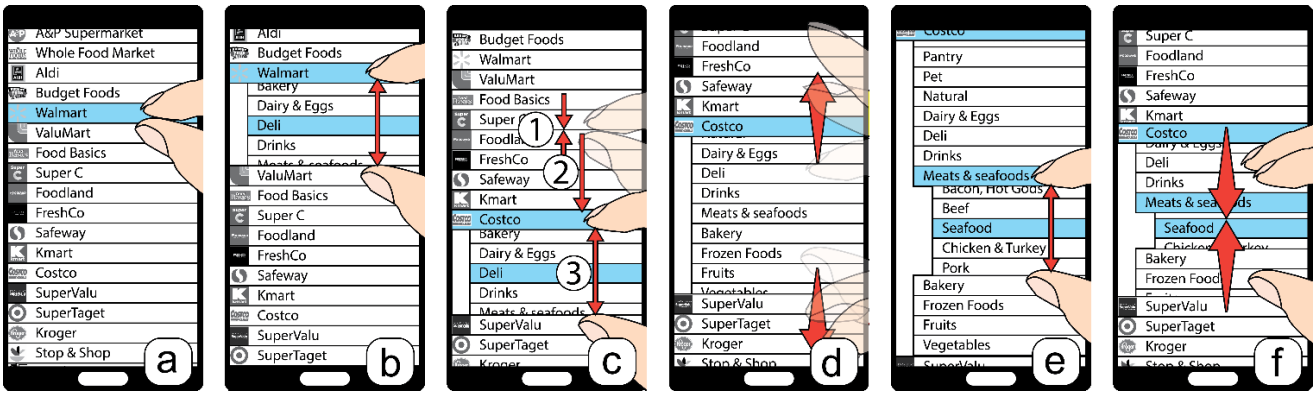

Figure 1. Steps of PinchList: (a) two pinched-in fingers starts browsing a list; (b) pinching-out the fingers reveals a sub-list; (c) move the fingers to explore another sub-list; (d) finger flicking to move the current view to the top and bottom edges of the screen; (e) start to explore the next two layers; (f) flicking with multiple fingers to navigate back to the previous layer.

ABSTRACT

Intensive exploration and navigation of hierarchical lists on smartphones can be tedious and time-consuming as it often requires users to frequently switch between multiple views. To overcome this limitation, we present PinchList, a novel interaction design that leverages pinch gestures to support seamless exploration of multi-level list items in hierarchical views. With PinchList, sub-lists are accessed with a pinch-out gesture whereas a pinch-in gesture navigates back to the previous level. Additionally, pinch and flick gestures are used to navigate lists consisting of more than two levels. We conduct a user study to refine the design parameters of PinchList such as a suitable item size, and quantitatively evaluate the target acquisition performance using pinch-in/out gestures in both scrolling and non-scrolling conditions. In a second study, we compare the performance of PinchList in a hierarchal navigation task with two commonly used touch interfaces for list browsing: pagination and expand-and-collapse interfaces. The results reveal that PinchList is significantly faster than other two interfaces in accessing items located in hierarchical list views. Finally, we demonstrate that PinchList enables a host of novel applications in list-based interaction.

CCS CONCEPTS

• Human-centered computing → **Human computer interaction;** *Gestural input; User interface design.*

KEYWORDS

Pinch gesture; Hierarchical list navigation; touchscreen;

ACM Reference format:

Teng Han, Jie Liu, Khalad Hasan, Mingming Fan, Junhyeok Kim, Jiannan Li, Xiangmin Fan, Feng Tian, Edward Lank, and Pourang Irani. 2019. PinchList: Leveraging Pinch Gestures for Hierarchical List Navigation on Smartphones. In *Proceedings of CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019).* ACM, New York, NY, USA. Paper 501, 13 pages. https://doi.org/10.1145/3290605.3300731

*Both authors contributed equally to the paper
†Contact author

CHI 2019, May 4-9, 2019, Glasgow, Scotland, UK.

ACM ISBN 978-1-4503-5970-2/19/05...$15.00.
DOI: https://doi.org/10.1145/3290605.3300731

1 INTRODUCTION

List view is a common and popular UI component for content presentation on smartphones. In a list view, items are aligned vertically in a list to support homogeneous data browsing and reading. It has been widely applied in mobile applications such as phonebook, setting, itinerary, email, notes and music playlist. In many cases, it follows hierarchical organizations to present multi-level structured information. Such list interfaces typically either use "expand-and-collapse" design to show/hide details of existing items in the same page, or use page transitions (i.e., pagination) to switch views of different layers [17].

Currently, users navigate lists on mobile devices via tap and swipe gestures. Although intuitive and easy to use, the gestures may not work efficiently on hierarchical list views [21]. Specifically, exploring items in the list views with tap and swipe gestures often demands intensive exploration while navigating ups and downs to access items in the hierarchy (theoretically analyzed cases are summarized in [4], e.g., comparing itinerary details when booking a flight, or searching for a setting option without knowing its exact position). In such cases, repeated tap and swipe operations could be tedious and time-consuming.

We propose PinchList, a novel interaction design that leverages pinch gestures to support quick and easy transitions in hierarchical lists on smartphones. PinchList allows users to examine an item's details or sub-list by splitting the list view with pinch-out gesture and return to a previous level with pinch-in gesture. PinchList supports two operation modes: (i) Pinch-and-Hold to navigate back and forth between two sequential layers in a list, and (ii) Pinch-and-Flick to switch among more than two levels. These two modes work coherently to enable efficient list exploration/browsing.

PinchList walkthrough: Figure 1 illustrates how PinchList works. A user starts exploring a list by touching the screen with two fingers pinched together (Figure 1a). An invisible line cursor is activated, in the middle of the fingers, upon the touch and consequently highlights an item located underneath it. Pinching-out the fingers splits the current view and reveals a window containing the item's child list (Figure 1b). The user explores the list by moving the fingers up and down. The user may also quickly switch to a different child list by pinching-in to close the current view, moving the fingers to another item, and pinching-out on the item to open the new child list (Figure 1c). We called this style of exploration as "Pinch-and-Hold" which works with a typical two-layer structure. If a list has more than two layers, "Pinch-and-Flick" operations are used to switch the views. Specifically, instead of holding the fingers, the user performs a flick gesture, which moves the top layer to edges and promotes the child layer to the front view (Figure 1d). The user may repeat steps a-c on the current layer (Figure 1e). To resume previous layers, the user makes a flick or pinch-in gesture with multi-fingers (Figure 1f), where the number of fingers indicates how many layers the operation applies to. For instance, two fingers resume one layer and three fingers resume two.

Benefits of PinchList: On mobile interactions, pinch gestures are commonly used for zoom in and zoom out operations (aka pinch-to-zoom). PinchList informs a new viable way to use pinch gestures for navigating lists on mobile UIs. It offers the following benefits: i) it minimizes the need for tap and swipe operations for list view switching, which often poses efficiency challenges with roundabout navigations [4]. ii) PinchList supports seamless multi-level list navigation via pinch and flick gestures, making PinchList applications scalable. iii) PinchList enables novel user experiences on list-based interaction. To show this, we designed several list applications to demonstrate that with PinchList, users can intuitively re-order files across folders, efficiently apply repetitive operations on multiple list items, and make list command and value selection in one step. These applications are not well supported with traditional list UI on smartphones.

We conducted two user studies to explore the design and performance of PinchList. In the first study, we quantitatively evaluated the performance of item acquisition with pinch-in and pinch-out gestures. Results revealed that item selection time with pinch gestures can be predicted with Fitts' law, in both scrolling and non-scrolling conditions. It was also suggested using the pinch gestures can accurately and efficiently select items at the height of 6mm, comparatively smaller than the industrial guideline (e.g., 8-10mm) for smartphone [17]. This allows more items to be shown on one screen, reducing the need for scrolling. The second study confirmed that PinchList outperforms traditional view-switch and expand-collapse lists in task completion time when users need to navigate back and forth to access information in a hierarchical list.

We make the following contributions: i) PinchList, a novel smartphone interface design that leverages pinch gestures for efficient navigation with hierarchical list view; ii) design of parameter and evaluation of PinchList's

performance on smartphone touchscreen; iii) application design that leverages PinchList for novel list interactions.

2 RELATED WORK

PinchList enables seamless and efficient list view manipulations via adopting pinch gestures. We first review related work on touch input paradigms that improve UI performance, which is also the design goal of PinchList, then we look at scrolling performance modeling and use of pinch gesture in other interactions.

2.1 Efficient Touch Input Paradigms

Touch input becomes prominent on modern computing devices but suffers from the fat finger and occlusion problems. Besides providing theoretical perspectives to help us understand human touch input [7, 15], a large body of work have been done to propose alternative UI designs and presentations to improve the touch input efficiency [24, 42]. The explorations include the design of multi-touch gestures that are considered promising for intuitively and efficiently manipulating UI elements. Researchers were interested in using multi-touch to emulate mouse functions for precise item selection [6, 36]. Two fingers are used to control a cursor that appears in the middle. A benefit of this metaphor is adding a tracking state layer between non-selection and selection on touchscreens [9], which enables functions like hover-to-preview. PinchList also takes benefits from such input metaphor on mobile touchscreens.

Other than this, a rich set of projects demonstrated benefits of novel input paradigms deploying the dexterities of fingers. For instances, Kin et al. [27] designed a multi-stroke two-handed marking menu for simultaneous menu and sub-menu selections tasks. Lepinski et al. [33] designed a marking menu based on simultaneous finger touches. Pin-and-Cross [35] requires users to use one finger to pin an object and another finger to select a target from a pre-activated menu. FastTap [19], built on users' spatial memory, is another touch-based interface for rapid access to menu items. In contrast, our work is not proposing a new gesture set, but looks into the use of pinch gestures in the context of list view manipulations.

Mobile and wearable devices have smaller touchscreens that are normally under 6 inches [16]. Under this constraint, using stroke gestures showed benefits in text input [11], command search [34] and item access [20]. Besides, to expand the vocabulary of touch input on miniature touchscreens, derivatives of simple gestures are used, e.g., touch/tap, double touch/tap, long touch/press, and sequences of taps [30, 39] and swipes [29]. Pinch gesture is also popular on smartphones, but to our knowledge, it has not been applied for menu or list selection.

2.2 List Scroll and Navigation Performance

Scrolling is a common task in content browsing UI, e.g., on list view. On desktop, typical scrolling techniques include using a mouse wheel, joystick, scroll bar, and multi-finger gesture on trackpad. On mobile devices, scrolling is often carried out with scroll gestures (e.g., finger swiping or flicking). Scrolling is considered more suitable than pagination on continuous and lengthy content [3].

To quantify the scrolling performance on desktop, Hinckley et al. [22] tested four scrolling techniques and found that the movement time (MT) had a good fit to the index of difficulty (ID), and thus the scrolling performance can be modeled with Fitts' law. The experiment was based on the case that target's position was known. Anderson [1] later reported that the scrolling movement time (T) is linear to the target distance (D) if users did not have prior knowledge of the target's position. Cockburn and Gutwin [12] clarified that the scrolling model is either logarithmic or linear, depending on "whether users can employ an open-loop ballistic phase of motion toward the target". For example, linear regression models fit to random ordered lists. In contrast, alphabetically or numerically ordered lists fit better with logarithmic models. The paper also stated that scrolling and non-scrolling tasks resulted in substantial differences among parameters, considering the fact that users could not visually locate the target quickly if scrolling action was required.

Navigating hierarchical lists involves a series of scrolling and selection, each of which depends on the layout manner of the corresponding layer [12]. In this paper, we rely on the previous findings to gain the prior understanding of PinchList's scrolling performance. However, using pinch gestures and a dual finger mid-line metaphor to select in a list on mobile touchscreens has not been formally studied.

2.3 Pinch Gestures in Interactions

The use of pinch gestures for interactions could be traced back to the 1980s [10]. Krueger [28] demonstrated a vision- based tracking system that allowed users to use index and thumbs of both hands to scale graphical objects. Wellner's [43] projected tabletop demonstrated pinch

gestures for scaling and panning operations. This revealed the basic attributes of pinch gesture: pulling two fingers apart to scale and dragging both fingers to translate. Pinch gestures have since been widely explored [23] and applied to support interactive tasks, such as pinch-to-zoom [25], context + focus data visualization [38], and parameter control [13]. Negulescu et al. [37] explored bi-manual pinch-to-zoom gestures. FingerGlass [26] used pinch gestures to define an area of interest and use the other hand to select in a magnified view. WritLarge [44] frames a selection portion on canvas and adjusts its size and orientation with pinch gestures. Such selection mechanism integrates pinch-to-zoom and better supports actions with the other hands. Pinch-to-Zoom-Plus [2] empirically examined designs that reduce clutching and panning required with current pinch-to-zoom technique. On a Macbook touchpad, Multi-finger pinch is used to display desktop or Launchpad [25].

Pinch gestures have been widely used on mobile phones since the announcement of iPhone in 2007 [40]. An iconic function of the gesture is zooming in and out while browsing a photo, webpage and map etc. Tran et al. [41] systematically examined how users pinch and spread on mobile phones. Designers have been exploring alternative use of pinch gestures on mobile phones, such as performing pinch gestures to close and open photo browser, or as a shortcut to "return" command [14]. In this paper, we incorporate pinch gestures to support seamless list view manipulations on mobile devices.

3 STUDY 1: SELECTION PERFORMANCE

PinchList uses a dual finger mid-line metaphor to select in a list. Prior similar work [6, 36] were neither designed for list navigation nor formally evaluated. We considered key parameters such as item size and item visibility while designing PinchList. In particular, we hypothesized that item size could be smaller than conventional requirements as items are selected by a line cursor instead of a finger, and users' performance would be affected by the visibility of the item (e.g., on-screen *vs.* off-screen). Consequently, Study 1 was designed to quantitatively evaluate the performance and to identify suitable design parameters for PinchList.

The list could be static or scrolling (i.e., dynamic) when fingers move, depending on the number of list items and its length compared to the screen height. We use a scrolling design that is akin to scrollbar on desktop. Specifically, the vertical position of the cursor on the screen is mapped to the portion of the list view that is displayed. This scrolling method is straightforward and simplifies training users on how to use it. The evaluation of the performance of such list item selection technique under both the static and scrolling conditions is required to optimize design parameters such as item size and list length.

We made the following analysis before designing the study. First, the selection performance is expected to follow the results of Cockburn and Gutwin's [12], that the selection time will be linear or logarithmic with distance, depending on whether users are able to employ a feedback-free ballistic phase of motion towards the target. If they can, the performance will be logarithmic. This requires users to have a mental model of the list contents, e.g., the list is alphabetically or numerically ordered. Second, different from tapping to select, the item selection in PinchList is not constrained by the contact area of the fingertip, thus the items could be smaller in size than the regular requirement (i.e., 8-10mm) from design guidelines [17]. This would allow more items to be displayed on the screen at a time, reducing the need for scrolling. Third, scrolling with pinch gestures is not expected to be efficient on a long list, where the CD gain (= length of the list / length of the screen) would be too large for practical operation. With these considerations, we followed Hinckley et al.'s [22] approach, designed and conducted this study using the Fitts' task paradigm.

3.1 Experiment Conditions

This study used 1D Fitts' reciprocal tasks to investigate how the target size and target distance affects the target acquisition time using PinchList, under both scrolling and non-scrolling conditions. The standard tap technique was included as a baseline.

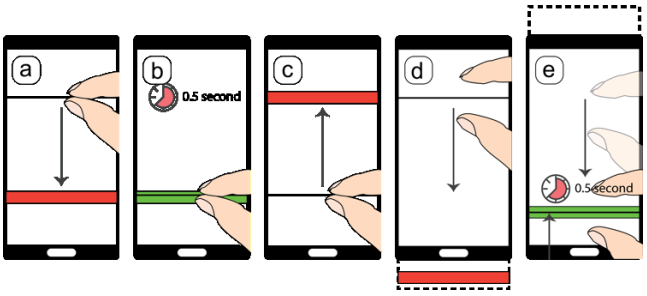

Figure 2: Examples of Study 1 interfaces. (a-c) static tasks with PinchIn; (d-e) dynamic tasks with PinchOut.

3.1.1 Techniques. Three techniques were evaluated: *PinchIn*, *PinchOut* and *Tap*. With *PinchIn*, a user acquires a target with thumb and index fingers pinched-in (Figure 2a-c). *PinchOut* is similar to *PinchIn*, whereas the two

fingers are kept apart at distances of users' choice (Figure 2d-e). We did not add any restrictions, but simply instructed participants to naturally and comfortably pinch-in/out their fingers. *Tap* is akin to the standard tap gesture.

3.1.2 Target Size. The target size was defined by the height of an item in the list view. We included target sizes of 2, 4, 6, 8 and 10mm. These target sizes were chosen to reflect the sizes of common UI elements on smartphones (e.g., 2.5mm for a hyperlink of a website; 4mm for the size of a key on a soft keyboard, or 10mm for a button).

3.1.3 Target Distance. We explored users' performance on selecting both on-screen and off-screen items in static and dynamic conditions, respectively. For *Static* condition, targets were placed 30, 60 and 90mm apart, making them always visible on the screen (Figure 2a-c) and participants did not need to scroll the view. For *Dynamic* condition, targets were placed 150, 180, and 210mm apart, making the targets invisible to participants initially (Figure 2d-e). Participants had to scroll the view to see the targets. To avoid potential impact of different CD gains resulting from different list length, we set a fixed CD gain of 2 for the dynamic selections. In case of *Tap*, participants swipe the view and tap on the target when it becomes visible.

3.2 Tasks and Procedure

For each condition, trials start with rendering two targets symmetrically about the screen center, colored in red. A participant moves their fingers up and down to select the targets until they complete all the repetitions of the condition. A line cursor is visualized to indicate current position. For *PinchIn* and *PinchOut*, once the cursor is moved within the target, the target turns to green, and the participant holds it (i.e., dwell) for 500 milliseconds to confirm the selection. For off-screen targets, the interface draws a seekbar to indicate the targets' position. The participants were instructed to select the targets as quickly and accurately as possible. The trial time was calculated from the time when a target was displayed on/off the screen to the time when they successfully selected the target.

We used a 3 (techniques) × 5 (target sizes) × 6 (target distances) within-subject design for this study. Each condition was repeated 11 times, and the first trial was excluded as users were asked to start from the screen center on the first trial. These yield 900 trials per participant. We counterbalanced the techniques across the participants and randomized the order of target sizes and distances.

A Huawei P9 Plus smartphone was used in the study. The phone has a screen size of 5.5 inches with a resolution of 1080×1920 pixels. The software was implemented with HTML5 + Javascript that ran on Chrome in full-screen mode. We used the same device for the other studies.

12 participants (4 females) with ages between 22 and 37 (M = 24.6, SD=3.9) participated in the study. All of them were right-handed and used smartphones frequently in their daily activities. Participants were seated while holding the device in their left hand, and performing the tasks using their right hand. They were first introduced the gestures and tasks, and had a few practice trials. The participants could take a short break after each condition. The study took about 40 mins for each participant.

3.3 Results

In Study 1, the participants had to successfully select each target before proceeding to the next trial. As such, there was no error data. The same protocol was used in prior evaluations [12, 18]. We removed 155 outliers (1.44%) from the data where the trial times were 3 standard deviation away from the mean. For PinchIn and PinchOut, average trial time increased drastically (10mm → 8mm: 5%, 8mm → 6mm: 6%, 6mm → 4 mm: 11%, 4mm → 2mm: 21%) when the target size was smaller than 6mm (Figure 3b).

We then performed repeated measures ANOVA (RM-ANOVA) and Bonferroni adjusted post-hoc pairwise comparisons to rest of the trials. As suggested by [12], users can rapidly visually locate the target and move to the data in the static condition, while in the dynamic condition, users have to scroll the list as the precursor to visual inspection. These make substantial differences, and as a result, the two conditions were analyzed separately.

RM-ANOVA showed a significant effect for the technique in static condition ($F_{2,22} = 19.49$, $p < 0.001$). Post-hoc pairwise comparisons showed that Tap (M = 674ms) was the fastest, followed by PinchIn (M = 1311ms) and PinchOut (M = 1356ms) (all $p < 0.05$). We also observed a significant effect in the dynamic condition ($F_{2,22} = 26.39$, $p < 0.001$). Post-hoc pairwise comparisons showed that Tap (M = 1682ms) was significantly faster than other two techniques, PinchIn (M = 2210ms) and PinchOut (M = 2160ms) (all $p < 0.05$). This is understandable as tapping can be done with no delay, while PinchIn and PinchOut require a need to slow down finger movement in order to prepare to stop in place.

For both conditions, significant differences were observed on target size (static: $F_{4,44} = 342.80$, $p < 0.001$,

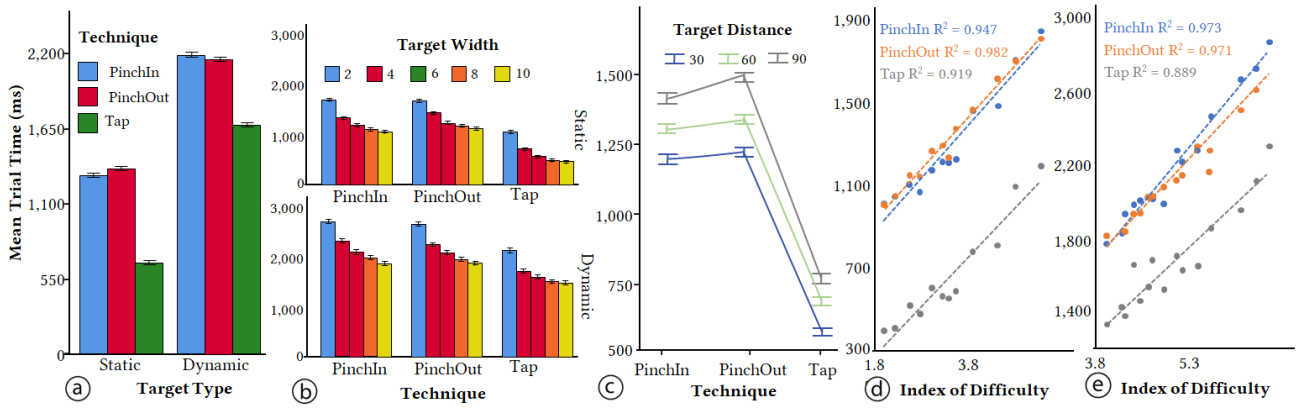

Figure 3: Study 1 results.

dynamic: $F_{4,44} = 193.12$, $p < 0.001$) and target distance (static: $F_{2,22} = 130.45$, $p < 0.001$, dynamic: $F_{2,22} = 44.68$, $p < 0.001$). Post-hoc pairwise comparisons of the target sizes revealed statistically significant difference between all the pairs (all $p < 0.05$). The average trial time increased with target size: the average time for 10mm, 8mm, 6mm, 4mm and 2mm were 906ms, 955ms, 1020ms, 1184ms, 1504ms in static condition, and 1740ms, 1813ms, 1922ms, 2097ms and 2514ms in dynamic condition, respectively. Additionally, post-hoc pairwise comparisons of target distance showed significant difference between 30mm (999ms), 60mm (1112ms) and 90mm (1230ms) in static condition, and between 150mm (1920ms), 180mm (1992ms) and 210mm (2139ms) in dynamic condition (all $p < 0.05$). As expected, targets at a closer distance was always faster than the targets that are located at a further distance.

For the static conditions, our results showed an interaction effect between technique and target distance ($p < 0.05$) (Figure 3c). It appears that the poor performance of *PinchOut* at a longer target distance is mainly caused by the fact that users have difficulty controlling the *PinchOut* when the targets are located at the screen edge.

Trial times across the techniques were further analyzed to see whether it can be modelled with the Fitts' law. Pairing the target size and target distance yielded 15 conditions with the index difficulty (ID) ranging from 2 to 5.52 bits for the static condition and between 4 and 6.73 for the dynamic condition. Linear regression analysis on the data revealed a strong correlation between the trial time and ID, with all R^2 values above 0.89 (Figure 3d and 3e). These findings confirm that target selection with *PinchIn* and *PinchOut* can be modelled with Fitts' law. Interestingly, we observed lower R^2 values for *Tap* compare to both *PinchIn* and *PinchOut*. It could possibly be due to the finger switching between touchscreen and mid-air for target selections with *Tap*, which was not required for the other two techniques.

3.4 Discussion

The results suggested that using *PinchIn* and *PinchOut* can select list items efficiently at a size of 6mm, comparatively smaller than regular guideline (i.e., 8-10mm) [17]. This potentially allows more items to be displayed at one screen, reducing the necessity of scrolling. It was found that selection performance with pinch gestures can be modelled with Fitts' law, providing further guidance on determining item size and list length of PinchList.

The results showed that *Tap* is faster than the two pinch gestures for both static and dynamic conditions. We believe that the results are primarily due to the extra dwell time added to the two pinch gestures. When scrolling is needed, manipulating the scroll as a precursor for visual inspection becomes a key limiting factor [12], and this weakens the impact of the dwell action. More importantly, the result delivers the message that using pinch gestures for a single layer list item selection may not be as efficient as tapping. This motivates us to investigate how the pinch gestures could benefit the hierarchical list navigations, which is the essential goal of designing PinchList.

The dwell time was included in the trial time to avoid biased analysis. However, dwell is not an ideal choice for item selection as it might interfere with task performance [22]. Selection can be made with other methods, e.g., force press, finger lift-off, depending on the tasks and applications. Meanwhile, selection actions are only needed to invoke an item and there are cases where invocation is not needed, such as when browsing items in a list.

One limitation of the study design is that we did not investigate the pinch gestures' performance on lists of various length. We suspect that there exists a tradeoff between the performance and the list length / number of items. When scrolling is required, using the pinch gestures takes benefits from being able to quickly locate the target with a small movement of fingers. However, if the list is too long, the scrolling speed will be too fast for users to visually search the target, and accurately position the line cursor onto the target. It is not recommended to use the pinch gestures in such cases.

4 STUDY 2: HIERACHICAL NAVIGATION PERFOMANCE

PinchList supports users to perform Pinch-and-Hold to navigate back and forth between two layers, and Pinch-and-Flick to switch views among more than two layers in list hierarchy. To validate this design, we compared the performance of PinchList with two traditional list view interfaces: Expand-and-Collapse and View-Switch, in a mockup navigating task that requires users to explore items under a hierarchical list.

4.1 Pinch-and-Hold, and Pinch-and-Flick

We incorporate pinch gestures with the use of multi-fingers to design the following two techniques.

4.1.1 Pinch-and-Hold. This is to enable seamless transitions between two layers. With Pinch-and-Hold, users start by touching the screen with two pinched-in fingers (normally index and thumb) and move them up and down to select items on a layer, e.g., layer 1 (Figure 4a). Pinching out the fingers splits the view, and shows a sub-list under a previously selected item. Users may keep the fingers apart and move them up and down to examine the sub-list, e.g., layer 2 (Figure 4b). If users do not find targets of interest, they could pinch-in the fingers to close the sub-list view, and continue exploring other items on layer 1 (Figure 4c-e).

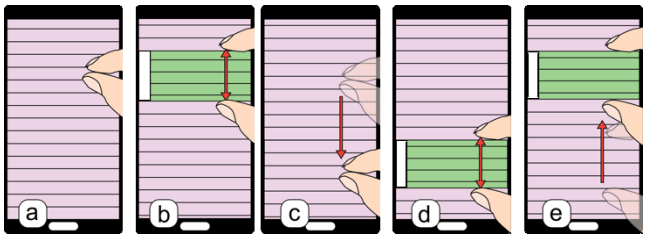

Figure 4: Use Pinch-and-Hold gestures to navigate between two layers.

4.1.2 Pinch-and-Flick. Users may need to explore lists with many layers. However, due to the binary states of pinch-in and pinch-out operations, Pinch-and-Hold imposes challenges while accessing more than two layers. To solve this issue, we design Pinch-and-Flick that combines pinch gestures with finger flicking. Instead of performing pinch-in gesture to resume a parent layer, a user can flick the fingers outwards (Figure 5a). This action moves and anchors the current layer to the top and bottom edges on the screen, and sends the child layer into the foreground (Figure 5b). The user is now able to perform the Pinch-and-Hold gestures on the next two layers in the hierarchy.

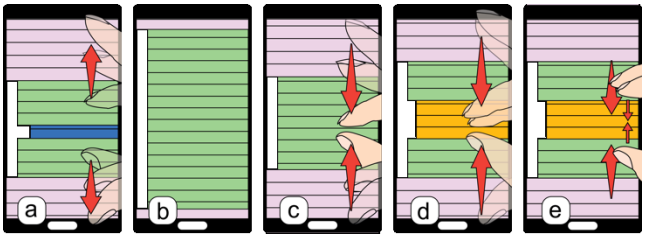

Figure 5: Pinch-and-Flick gestures.

To navigate back to previous layers, the user performs flick-inwards gesture (Figure 5c). This is analogous to the pinch-in expect that (i) the user shall move fingers faster, and (ii) the fingers may get off the screen when the gesture is completed. The user may also use multiple fingers to indicate the number of previous layers to resume, e.g., 2-finger flicking resumes one previous layer, and 3-finger flicking resumes two previous layers (Figure 5d). This setting also applies to the pinch-in gestures, such that users can navigate back to a certain previous layer directly without going through intermediate layers (Figure 5e).

4.2 Experiment Tasks and Conditions

Study 2 aims to evaluate the performance of the described gestures in accessing items in a hierarchical list. To mimic real usage scenarios, an analytic decision-making task was designed that asked users to explore and compare fruit prices at different markets, on different dates and find the minimum price. We used a three-layer hierarchical list in the study, with 10 items on each layer (i.e., no scrolling needed, 1st layer – dates, 2nd layer – markets, 3rd layer – fruits with prices). To minimize the visual search effort and examine the transition efficiency among layers, the items on each layer were either numerically or alphabetically ordered, and the target items were highlighted. In each trial, participants followed given target path(s) to find the information needed for comparing the prices.

4.2.1 Navigation Techniques. Participants are able to use Pinch-and-Hold and Pinch-and-Flick on the PinchList interface (Figure 6-top). Besides, we implemented an Expand-and-Collapse and a View-Switch list interface as baselines. These are two commonly used interface

paradigms to support list navigations that are operated with tap and swipe gestures. Expand-and-Collapse shows and hides details of existing list items by expanding and collapsing list content vertically upon a finger tap (Figure 6-middle). View-Switch paginates list layers into separate views (Figure 6-bottom). Navigation among the views is done by tapping an item to open a view containing the sub-lists or tap on a back button to return to a previous view.

To eliminate the potential effects of item size on navigating efficiency of the techniques, we set the item size to 8mm according to [17] in the study for all conditions.

4.2.2 Task Types. Two types of tasks are set. One asks participants to "Type 1 - find the lowest price of a fruit, in a market, on *x* dates", and the other asks users to "Type 2 - find the lowest price of a fruit, in *x* markets, on a date". This is to examine efficiencies when users need to navigate among 3 layers (Type 1) and among 2 layers (Type 2).

4.2.3 Task Complexities. This condition defines the value of *x* in Task Types. Here in this study, we set $x = 1, 3, 5$, which means in a trial, participants need to compare 1, 3, 5 prices before they can find the lowest one. In case of 1, they just need to report the observed price, while in cases of 3 and 5, they need to navigate to the prices sequentially and report the lowest price. This is to examine the multi-layer navigation efficiencies using different techniques.

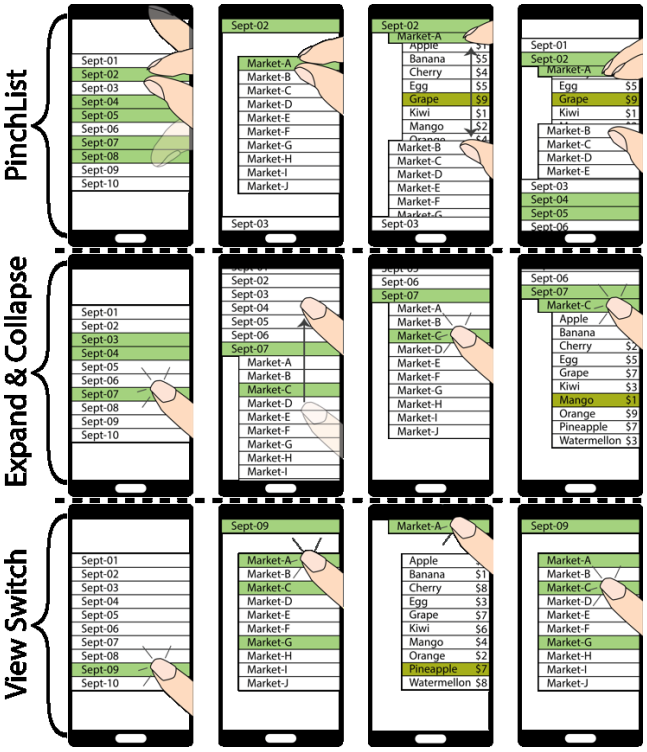

Figure 6: The three techniques used in Study 2.

4.3 Procedure

We used a 3 (techniques) × 2 (task types) × 3 (task complexities) within-subject design, with each condition repeated 10 times, resulting in a total of 180 trials per participant. The techniques were counter-balanced across the participants and the presentation orders of the task types and task complexities were randomized. The tasks of each Type-Complexity group were pre-defined with randomly generated sequences. For example, Type 1 and Complexity 3 always returned the same set of 10 tasks, no matter which technique was being tested. This was to make sure the same tasks were performed with each of the techniques. Participants had to complete the 10 tasks before moving to the next Type-Complexity group. To avoid leveraging memories of previous task sequences and results, the order of the 10 tasks was randomized, and the prices to check were randomly generated in real time.

At the beginning of each trial, a prompt of the trial task was shown on the top of the screen. Participants spent a short time reading the task and pressed a "start" button on the screen to start the trial. The prompt was always shown during the test. After browsing the items, participants pressed the "back" button on the device to end the trial and typed the answer. If the answer was not correct, a trial with the same task was shown at the end of the 10 original trials.

We instructed the participants to perform the tasks as quickly and accurately as possible. The trial time was measured by the time span between the presses on the "start" and "back" buttons. The errors count was incremented by 1 if an incorrect answer was entered.

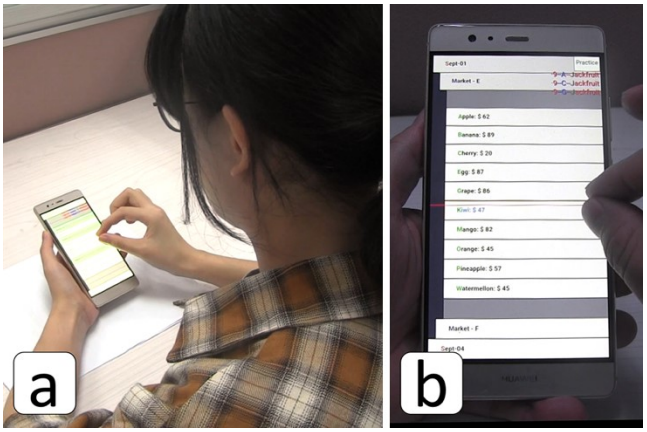

Figure 7: (a) A participant in Study 2; (b) Study 2 interface.

We recruited a different group of 12 participants (4 females) with ages between 21 and 33 (M = 26.6, SD = 3.7). All of them were right-handed and used smartphones

frequently in their daily activities. Like Study 1, the participants were seated while holding the device in their left hand, and performed the tasks using their right hand (Figure 7). They were given practice trials to get familiar with the three techniques and tasks. The study started when the participants felt comfortable using the navigation techniques. The study lasted ~1 hour for each participant.

4.4 Results

In total, data from 2375 trials were collected, among which 215 trials were marked as errors because the participants typed wrong answers. We observed that the errors were distributed equally across the techniques. In specific, for PinchList, Expand-and-Collapse, and View-Switch, 9.9%, 7.5% and 9.8% error rates were observed, respectively.

We excluded these trials for the rest of our analysis. The trial time was analyzed using RM-ANOVA and Bonferroni adjusted post-hoc pairwise comparisons. The results revealed that PinchList with a mean trial time of 5.99s was significantly faster ($F_{2,22} = 383.21$, $p < 0.001$) than View Switch (M = 7.13s) and Expand-and-Collapse (M = 9.53s) (Figure 8a). Both Task Type and Task Complexity were found to have interaction effects with Technique. First, an interaction effect was found on Technique and Task Type ($F_{2,22} = 27.64$, $p < 0.001$) (Figure 8b). As expected, the participants used different strategies for different task types. With PinchList, they mainly performed pinch-in and pinch-out gestures for the Type 2 tasks, and more flick gestures are required when performing Type 1 tasks. The frequent switch between the pinch and flick gestures made the task completion time longer. Meanwhile, an interaction effect was found on Technique and Task Complexity ($F_{4,44} = 78.16$, $p < 0.001$) (Figure 8c). When users need to browse more items to make a decision, using PinchList was faster compared to the other techniques.

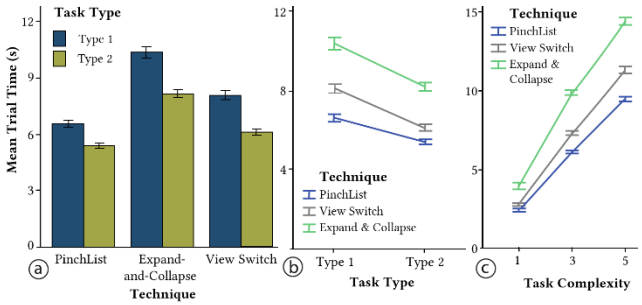

Figure 8: Study 2 results.

Besides, in total, the finger lift-up (from the screen) happened 3528, 8704, and 8614 times for the techniques. While using PinchList, 1262 out of 1409 times (89.6%) the participants used the three finger flicking gesture to navigate back to the 1st layer from the 3rd layer.

4.5 Discussion

Compared to the Expand-and-Collapse and View-Switch list interfaces, PinchList showed advantages in browsing information located in different levels of a hierarchy. It was shown to be faster in both types of the tasks, and for different levels of complexity. This makes it a promising list navigation technique when users need to frequently navigate up and down in multiple layers. We anticipate that the more layers users need to browse in the hierarchy, the more efficiency benefit PinchList could bring. This is due to the following two reasons. First, the layer transitions with pinch-in and pinch-out gestures do not require users to leave the fingers off the screen, nor moving the fingers to reach a "back" button. Second, PinchList provides an efficient way to navigate back to previous layers, i.e., using multiple fingers to indicate the number of layers to resume, and users can start right away to explore the next item as fingers remained on the list.

User feedback on PinchList was generally positive. Most participants appreciated the simple design of PinchList. However, some found it challenging when frequent switch between pinch and flick gestures was needed. They felt the pinch gestures were intuitive to operate a two-layer hierarchy, but the mixed use with flick gestures was not that straightforward. Unlike the pinch gestures, flick gestures do not require users to keep fingers on the screen when the action is completed. Besides, flick gestures are normally performed in a faster way, without the need for examining or locating targets. Users may get confused while using the two gestures together. We expect this confusion will diminish when users get more training and become more familiar with the interface.

In the study, we did not consider long lists, where scrolling on each layer is required and more visual inspection is needed. Involving more factors will make the study conditions hard to control and the study procedure redundant. Specifically, for View-Switch and Expand-and-Collapse, more swiping gestures are expected. For PinchList, users need to move fingers up and down more often to bring the targets into view. It is expected that with a long list, the height of the sub-window view created with the pinch-out gesture shall be considered as a

factor, which affects how fast users can visually locate the targets.

5 APPLICATIONS

Besides improving the navigating efficiency, PinchList enables novel interaction opportunities on mobile list UI.

5.1 List Reorder

Users may want to re-order a list, such as when managing their agenda, or photo albums. The current interface supports users reordering items by dragging them on non-hierarchical lists (i.e., single layer) [17], which is simple and intuitive. This action, however, is not applicable to multi-layer lists, where users often have to select the item, invoke "transfer to" command, and select a target folder.

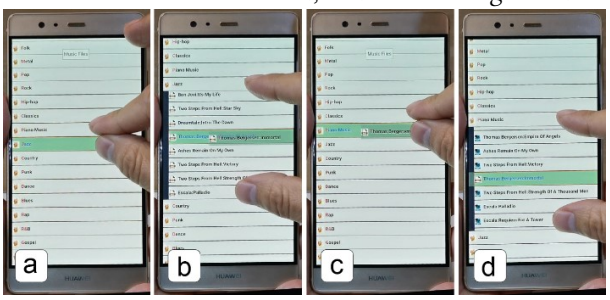

Figure 9: Using pinch gestures to re-order a music list.

PinchList supports the intuitive "drag" action to move an item across different folders. For example, a user wants to move a song in folder "Jazz" to "Piano". She first moves to the song with a pinch-out gesture and makes a harder force press to select the item (Figure 9a-b). When she pinches-in and closes the folder, the song item is visually docked to the side of the fingers while not affecting the fingers' operations (Figure 9c). She then opens the "Piano" folder with the pinch gestures and moves the fingers up and down to put the item to a desired position in the list (Figure 9d). Releasing the fingers completes the task.

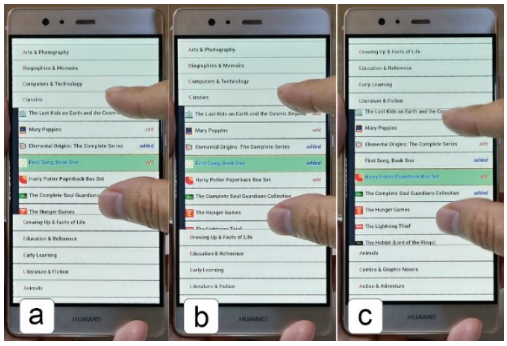

Figure 10: Using pinch gestures to add multiple shopping items under different categories.

5.2 Subview Menu

PinchList can be used to support quick multi-selections with sub-view menus. When a user needs to apply repetitive operations to multiple list items, e.g., adding items to cart when browsing a shopping app, she normally has to navigate back and forth from the list views. With PinchList, such efforts could be alleviated. For instance, the user wants to purchase several books which are located under different categories. She could use pinch-in and pinch-out gestures to quickly navigate to a book and add it to the shopping cart with a force press (Figure 10a-c). She can then continue the browsing and add another book following the same procedure.

5.3 Simultaneous Command and Value Selection

A benefit of using PinchList interface is that it supports command invocation and value selection in a single action (i.e., no need to leave fingers off the screen before an operation is completed). Upon making a selection of a command item with pinch gestures, a user can choose a value from the sub-view with fingers moving up and down. Releasing the fingers confirms the selection. The user can also cancel the value selection by pinching-in and leaving the fingers off the screen. This command invocation mechanism not only applies to selection of discrete values, but also applies to the selection of continuous values. For instance, the user can adjust music volume by first invocating the command, and moving the fingers up and down to adjust the volume (Figure 11).

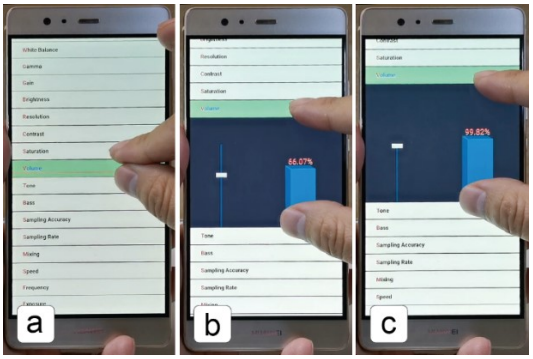

Figure 11: Using pinch gestures to select volume command and adjust the value.

6 DISCUSSION AND FUTURE WORK

Study 1 results showed that users can select list item efficiently at size 6mm, allowing more items to be displayed at one screen and reducing the necessity of scrolling. Using pinching gestures to select list item in both scrolling and non-scrolling conditions can be modeled with Fitts' Law. The pinch gestures were slower

than Tap for both scrolling and non-scrolling conditions, suggesting that using pinch gestures may not be best suited for single layer list selections. Study 2 investigated PinchList's performance in navigating hierarchical layers. The results clearly indicated that PinchList was faster in browsing information located in different levels of a hierarchy. PinchList can be efficient for hierarchical list exploration when users need to navigate up and down frequently. In both studies, we used a visible line to provide visual cues on the fingers' position and selected item. In real applications, other visual cues can be used (e.g., item highlight, fisheye effect [5, 32]). It is also possible to provide no visual cues, but further studies are required to evaluate the performance.

We did not observe fat finger problems from the studies and nor were they mentioned in the participants' feedback. This might be due to the indirect nature of selection actions with the line cursor. In real applications, contents can be placed to avoid potential finger occlusions (e.g., weighted towards the left to allow the fingers to operate on the right side). Nonetheless, we have found several limitations of PinchList:

Screen Edges: It becomes difficult for users to move fingers when an item is located close to top or bottom screen edges, especially when using pinch-out gestures. One solution could be extending the length of the list, such that the list is always in scrolling mode when users move the fingers up and down, and they do not need to reach the screen edges to acquire the items. Designers could also decrease the item height (e.g., 6mm based on Study 1), and make more buffer space on the edges.

View Displacement: With PinchList, users could conveniently examine a sub-list via moving the pinched-out fingers up and down. This may sometimes scroll the parent list's position when users pinch in and return to the previous layer. Designers should be careful on the possible view displacement which may cause confusion to users.

Learnability and Discoverability: Although users are familiar with pinch gestures, applying them to operate list view is still new. We found that PinchList has learnability challenges. The study participants spent more time to get used to the PinchList gestures. Meanwhile, some participants expressed that they felt more mental workload when using PinchList, as they have to memorize the new gestures and recall how the list would behave with the gestures. Some other participants felt uncomfortable when they had to keep the fingers on the screen for a longer period with pinch-in/out gestures. In addition, new users may not know such unconventional gestures are available. These problems can be alleviated by providing sufficient tutorial and cues, e.g. video instructions and reminding the users of the availability of PinchList gestures when the tasks require frequent up and down navigations.

Usage: Like other multi-touch interactions, PinchList is limited as it requires a typical two-handed interaction scenario, with one hand holding the device and the other hand performing the tasks. Nonetheless, one-handed version of PinchList could be explored via deploying more expressive dimensions of single touch input [8], or incorporating finger aware interactions on the device [31].

Future works include carrying out studies on real lists, to quantify PinchList's pros and cons on exploring lists with greater lengths and deeper hierarchies, as well as on different data types. We will further investigate how PinchList can be used in combination with tap and swipe gestures. This will make it more adaptive to current interfaces that users are already familiar with. Furthermore, it is also worth exploring new use scenarios with PinchList that could make list view more interactive and more expressive in functions.

7 CONCLUSION

PinchList informs a new viable way to use pinch gestures for navigating lists on mobile UIs, enabling a host of new applications in list-based interaction. We first conducted a user study to evaluate the performance of item selection with pinch-in and pinch-out gestures. The results confirmed that the performance can be modeled with Fitts' Law under both scrolling and non-scrolling conditions. The second study compared PinchList with two standard list interfaces: View Switch and Expand-and-Collapse, in tasks that involve browsing multiple items before reaching a decision in a list hierarchy. The results revealed that with PinchList, users can access hierarchical items faster than the other two interfaces that we commonly use on smartphones. Finally, the paper demonstrates that PinchList enables new list UI interaction opportunities.

ACKNOWLEDGMENTS

We acknowledge the support from the National Key R&D Program of China (Grant No. 2016YFB1001405), the National Natural Science Foundation of China (Grant No. 61872349), the Key Research Program of Frontier Sciences, CAS (Grant No. QYZDY-SSW-JSC041), the CAS Pioneer

Hundred Talents Program, the NSERC CRC program as well as the NSERC Discovery grant.

REFERENCES

[1] Tue Haste Andersen. 2005. A simple movement time model for scrolling. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05). ACM, New York, NY, USA, 1180-1183. http://dx.doi.org/10.1145/1056808.1056871

[2] Jeff Avery, Mark Choi, Daniel Vogel, and Edward Lank. 2014. Pinch-to-zoom-plus: an enhanced pinch-to-zoom that reduces clutching and panning. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14). ACM, New York, NY, USA, 595-604. https://doi.org/10.1145/2642918.2647352

[3] Nick Babich. UX: Infinite Scrolling vs. Pagination. Last retrieved on Sep 8th, 2018, from https://uxplanet.org/ux-infinite-scrolling-vs-pagination-1030d29376f1

[4] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2016. Visual Menu Techniques. ACM Comput. Surv. 49, 4, Article 60 (December 2016), 41 pages. https://doi.org/10.1145/3002171

[5] Benjamin B. Bederson. 2000. Fisheye menus. In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (UIST '00). ACM, New York, NY, USA, 217-225. http://dx.doi.org/10.1145/354401.354782

[6] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. 2006. Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06). ACM, New York, NY, USA, 1263-1272. http://dx.doi.org/10.1145/1124772.1124963

[7] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: modeling finger touch with fitts' law. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 1363-1372. https://doi.org/10.1145/2470654.2466180

[8] Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012. The fat thumb: using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services* (MobileHCI '12). ACM, New York, NY, USA, 39-48. https://doi.org/10.1145/2371574.2371582

[9] William BUXTON. 1990. A three-state model of graphical input. In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction* (1990), North-Holland Publishing Co., 725582, 449-456.

[10] William Buxton. Multi-Touch Systems that I Have Known and Loved. Bill Buxton. Microsoft Research. Original: Jan. 12, 2007. Version: Feb 1, 2009.

[11] Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 2014. Swipeboard: a text entry technique for ultra-small interfaces that supports novice to expert transitions. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14). ACM, New York, NY, USA, 615-620. https://doi-org/10.1145/2642918.2647354

[12] Andy Cockburn, and Carl Gutwin. 2009. A predictive model of human performance with scrolling and hierarchical lists. Human-Computer Interaction 24, 3, 273-314. DOI= http://dx.doi.org/10.1080/07370020902990402.

[13] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. 2009. A review of overview+detail, zooming, and focus+context interfaces. ACM Comput. Surv. 41, 1, Article 2 (January 2009), 31 pages. http://dx.doi.org/10.1145/1456650.1456652

[14] Sam Costello. 2018. Clear To Do List iPhone App Review. Last retrieved on Jan 7th, 2019, from https://www.lifewire.com/clear-to-do-list-app-review-1999157

[15] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47(6), 381-391. http://dx.doi.org/10.1037/h0055392

[16] Media Genesis. Popular Screen Resolutions: Designing for All. Last retrieved on Sep 8th, 2018, from https://mediag.com/news/popular-screen-resolutions-designing-for-all/

[17] Google. Material Design - Lists. Last retrieved on Sep 8th, 2018, from https://material.io/design/components/lists.html

[18] Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 281-290. https://doi.org/10.1145/1054972.1055012

[19] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. 2014. Faster command selection on tablets with FastTap. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 2617-2626. http://dx.doi.org/10.1145/2556288.2557136

[20] Teng Han, Jiannan Li, Khalad Hasan, Keisuke Nakamura, Randy Gomez, Ravin Balakrishnan, and Pourang Irani. 2018. PageFlip: Leveraging Page-Flipping Gestures for Efficient Command and Value Selection on Smartwatches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA, Paper 529, 12 pages. https://doi.org/10.1145/3173574.3174103

[21] Khalad Hasan, David Ahlström, Junhyeok Kim, and Pourang Irani. 2017. AirPanes: Two-Handed Around-Device Interaction for Pane Switching on Smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 679-691. https://doi.org/10.1145/3025453.3026029

[22] Ken Hinckley, Edward Cutrell, Steve Bathiche, and Tim Muss. 2002. Quantitative analysis of scrolling techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02). ACM, New York, NY, USA, 65-72. http://dx.doi.org/10.1145/503376.503389

[23] Eve Hoggan, Miguel Nacenta, Per Ola Kristensson, John Williamson, Antti Oulasvirta, and Anu Lehtiö. 2013. Multi-touch pinch gestures: performance and ergonomics. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces* (ITS '13). ACM, New York, NY, USA, 219-222. https://doi.org/10.1145/2512349.2512817

[24] Christian Holz and Patrick Baudisch. 2010. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 581-590. https://doi.org/10.1145/1753326.1753413

[25] Apple Inc. Use Multi-Touch gestures on your Mac. Last retrieved on Sep 8th, 2018, from https://support.apple.com/en-us/HT204895

[26] Dominik P. Käser, Maneesh Agrawala, and Mark Pauly. 2011. FingerGlass: efficient multiscale interaction on multitouch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 1601-1610. https://doi.org/10.1145/1978942.1979175

[27] Kenrich Kin, Björn Hartmann, and Maneesh Agrawala. 2011. Two-handed marking menus for multitouch devices. ACM Trans. Comput.-Hum. Interact. 18, 3, Article 16 (August 2011), 23 pages. https://doi-org/10.1145/1993060.1993066

[28] Myron W. Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. 1985. VIDEOPLACE—an artificial reality. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '85). ACM, New York, NY, USA, 35-40. http://dx.doi.org/10.1145/317456.317463

[29] Yuki Kubo, Buntarou Shizuki, and Jiro Tanaka. 2016. B2B-Swipe: Swipe Gesture for Rectangular Smartwatches from a Bezel to a Bezel. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 3852-3856. https://doi-org/10.1145/2858036.2858216

[30] Benjamin Lafreniere, Carl Gutwin, Andy Cockburn, and Tovi Grossman. 2016. Faster Command Selection on Touchscreen Watches. In *Proceedings of the 2016 CHI Conference on Human*

Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 4663-4674. https://doi-org/10.1145/2858036.2858166

[31] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (UIST '18). ACM, New York, NY, USA, 779-792. https://doi.org/10.1145/3242587.3242605

[32] Eric Lecolinet and Duc Nguyen. 2006. Représentation focus+contexte de listes hiérarchiques zoomables. In *Proceedings of the 18th Conference on l'Interaction Homme-Machine* (IHM '06). ACM, New York, NY, USA, 195-198. http://dx.doi.org/10.1145/1132736.1132767

[33] G. Julian Lepinski, Tovi Grossman, and George Fitzmaurice. 2010. The design and evaluation of multitouch marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 2233-2242. https://doi.org/10.1145/1753326.1753663

[34] Yang Li. 2010. Gesture search: a tool for fast mobile data access. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 87-96. https://doi.org/10.1145/1866029.1866044

[35] Yuexing Luo and Daniel Vogel. 2015. Pin-and-Cross: A Unimanual Multitouch Technique Combining Static Touches with Crossing Selection. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15). ACM, New York, NY, USA, 323-332. https://doi-org/10.1145/2807442.2807444

[36] Justin Matejka, Tovi Grossman, Jessica Lo, and George Fitzmaurice. 2009. The design and evaluation of multi-finger mouse emulation techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 1073-1082. https://doi.org/10.1145/1518701.1518865

[37] Matei Negulescu, Jaime Ruiz, and Edward Lank. 2011. ZoomPointing revisited: supporting mixed-resolution gesturing on interactive surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (ITS '11). ACM, New York, NY, USA, 150-153. https://doi.org/10.1145/2076354.2076382

[38] Dmitry Nekrasovski, Adam Bodnar, Joanna McGrenere, François Guimbretière, and Tamara Munzner. 2006. An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 11-20. http://dx.doi.org/10.1145/1124772.1124775

[39] Ian Oakley, DoYoung Lee, MD. Rasel Islam, and Augusto Esteves. 2015. Beats: Tapping Gestures for Smart Watches. *In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 1237-1246. https://doi-org/10.1145/2702123.2702226

[40] Nilay Patel. 2012. The myth of pinch-to-zoom: how a confused media gave Apple something it doesn't own. Last retrieved on Sep 8th, 2018, from https://www.theverge.com/2012/8/30/3279628/apple-pinch-to-zoom-patent-myth

[41] Jessica J. Tran, Shari Trewin, Calvin Swart, Bonnie E. John, and John C. Thomas. 2013. Exploring pinch and spread gestures on mobile devices. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services* (MobileHCI '13). ACM, New York, NY, USA, 151-160. https://doi.org/10.1145/2493190.2493221

[42] Daniel Vogel and Patrick Baudisch. 2007. Shift: a technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 657-666. https://doi.org/10.1145/1240624.1240727

[43] Pierre Wellner. 1991. The DigitalDesk calculator: tangible manipulation on a desk top display. In *Proceedings of the 4th annual ACM symposium on User interface software and technology* (UIST '91). ACM, New York, NY, USA, 27-33. http://dx.doi.org/10.1145/120782.120785

[44] Haijun Xia, Ken Hinckley, Michel Pahud, Xiao Tu, and Bill Buxton. 2017. WritLarge: Ink Unleashed by Unified Scope, Action, & Zoom. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 3227-3240. https://doi.org/10.1145/3025453.3025664