The Hold-and-Move Gesture for Multi-touch Interfaces

Alexander Kulik **Jan Dittrich** **Bernd Froehlich**

Bauhaus-Universität Weimar
Bauhausstrasse 11, 99423 Weimar, Germany
<first name>.<last name>@uni-weimar.de

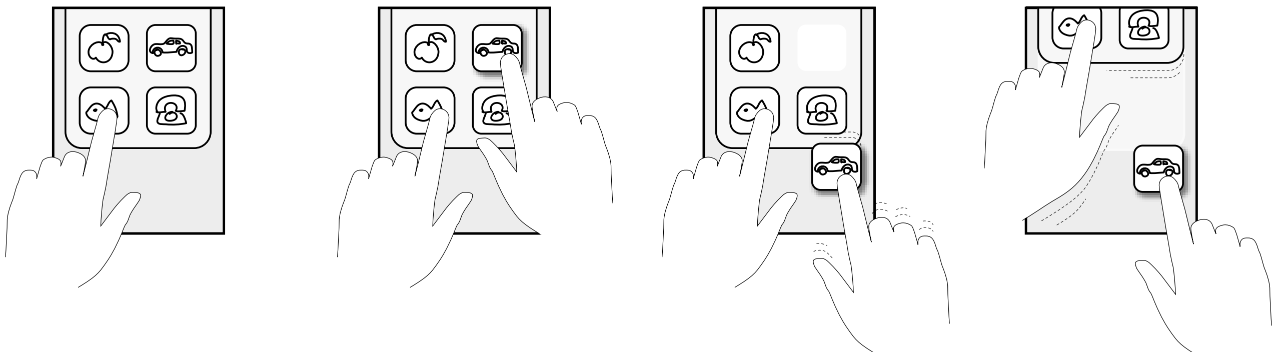


Figure 1. The *hold-and-move* gesture. (a) The first contact point (here a finger of the left hand) is always associated with the background. (b) The second contact point (here a finger of the right hand) selects an item. (c) The item can now be moved in relation to the background that is held with the left hand. (d) The left hand may perform panning of the background and even clutching while the right hand holds on to the selected item.

ABSTRACT
We present the two-finger gesture hold-and-move as an alternative to the disruptive long-tap which utilizes dwell times for switching from panning to object dragging mode in touch interfaces. We make use of a second finger for object selection and manipulation while workspace panning is operated with the first finger. Since both operations can be performed simultaneously, the cumbersome and hard-to-control autoscrolling function is no longer needed when dragging an object beyond the currently visible viewport. Single-finger panning and pinch zooming still work as expected. A user study revealed that hold-and-move enables faster object dragging than the conventional dwell-time approach and that it is preferred by most users.

Author Keywords
Multi-touch; dwell times; navigation; manipulation; hold-and-move.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: Input devices and strategies; Interaction styles.

INTRODUCTION
Mode switching based on dwell times is widely used with mobile touch devices. It generally toggles between reference frames for motion input, e.g. panning or scrolling the entire interaction space vs. dragging individual items. Motion input from a single finger is primarily used for moving the entire screen content. Dragging individual items requires that they are first selected by a long tap based on a predefined dwell time. This can be very annoying. Dwell times often take too long if the user wants to switch to the respective mode and they may also occur accidentally.

We developed the hold-and-move interaction technique (Figure 1), which uses an implicit input differentiation based on Guiard's "left-hand precedence in action" principle [8]. Hold-and-move associates the first contact point with the background and thus motion input from a single finger always controls panning. In order to manipulate individual items, the background must be held with the first finger. A second finger may then select an individual item and move it in relation to the background. In addition, the first finger may perform panning of the background and clutching while the second finger holds on to the selected item. The manipulative gesture consistently ties the fingers to the graphics displayed under the contact point, which enforces the physicality of the interface.

The development of the hold-and-move pattern was motivated by an exploratory study on the timing of common multi-touch input actions. We observed that dwell times between a touch event and the actual motion onset are an intrinsic characteristic of touch-based input and that their duration depends heavily on the activity. On average we recorded 130 ms dwell time during object dragging tasks if the functionality is directly available (without the requirement of an initial long tap). A regular long tap with a typical duration of 500 ms is therefore an interruption of the natural interaction flow. Simply reducing this threshold, however, is rarely feasible as it would increase the chance of false positives during navigation tasks.

The analysis of the pinch-zoom gesture confirmed the symmetric nature of this input action. Both fingers are placed down almost simultaneously with only 55 ms time difference on average. Hence, we designed the hold-and-move gesture as a complementary asymmetric two-finger gesture. It can be distinguished from symmetric input based on the time difference between both touch events. We found three main advantages of this approach:

- The disruptive long tap can be avoided.
- The asymmetric gesture is compatible with established one-finger gestures and can be clearly distinguished from symmetric two-finger input like pinch zooming.
- Manipulation and navigation can be operated simultaneously.

Mode switching based on dwell times provokes interferences between the two input modalities, e.g., object dragging and panning. Hold-and-move avoids this. The asymmetric two-finger gesture may instead interfere with other two-finger input, e.g., pinch zooming. Our studies indicate that this rarely happens as the pinch-zoom gesture is generally initiated symmetrically. A comparative user study revealed that hold-and-move can be more efficient than the dwell-time approach and that it is preferred by most users. We further analyzed the timing structure of the hold-and-move pattern and derive guidelines for its implementation.

RELATED WORK

Guiard's analysis [8] on asymmetric bi-manual activity inspired the design of many highly effective human computer interfaces (e.g. [5, 3, 11, 6]). Unlike our approach, asymmetric bi-manual interaction techniques generally require the explicit identification of input from the user's dominant and non-dominant hand. Towards this end, several researchers recently proposed to operate tasks of the non-dominant hand with direct touch input while the dominant hand performs more accurate operations with a pen device [21, 4, 10].

Other approaches to increase the expressiveness of direct motion input build on explicit mode switching [13, 15], distinct hand shapes [7] or various motion gestures [20, 16, 18, 17]. Li et al. [13] compared different mode switching techniques for pen-operated interfaces: dwell times, pressing the pen's barrel button, pressure and mode switching with additional input from the non-dominant hand. Dwell times showed the worst performance whereas the best results were achieved with bi-manual input. This observation clearly supports our approach to circumvent dwell times by exploiting the additionally available information of multi-touch devices.

Lank et al. [12] also analyzed mode switching with the non-dominant hand for pen input. They were focusing their research on the initiation pattern and found that the mode switching action triggered by the non-dominant hand generally precedes the dominant hand's action. Following the principles of naïve physics, Siio et al. suggested to use the palm of the pen-operating hand to fixate a virtual background for inking as if it were paper to write on [19].

Lü et al. [14] introduced a drawing-based selection technique for small targets on mobile touch devices. They note that drawing requires a mode change in most applications and suggest to hold the background with one finger before drawing with another one. While this idea is similar to our hold-and-move technique, it is also more constrained to serve only this particular application. The implementation details and the usability aspects of such a bi-manual mode switching technique are not addressed in their publication. In that sense, our work is complementary to theirs. We contribute an analysis of the timing parameters to distinguish hold-and-move from other two-point motion input, describe its general applicability and verify its usability benefits in a formal study.

THE TIMING OF TOUCH GESTURES

We were interested in the timing of common input actions on touch-screen devices. In particular, we analyzed unconscious dwell times in common panning and object-dragging tasks and also the time difference between both touch events initiating a pinch-zoom gesture. We expected that dwell times between a touch event and the respective motion onset would occur naturally, both in panning and dragging tasks. In case of occurrence, we were also measuring its duration to derive an optimal dwell time for mode switching. For the symmetric pinch-zoom actions requiring two instead of only a single contact point, we also expected intrinsic timing characteristics that would allow a clear distinction from other two point touch gestures.

Experimental Setup

We asked ten students from our campus to perform a series of dragging, panning and zooming tasks using a mobile touch device (Apple iPod® touch 2). All participants but one stated to have only marginal previous experience with touchscreen interfaces. With detailed instructions and preparatory training we ensured that our test users were fluently operating the device during the data gathering. The purpose of the study was not explained to the participants.

Mobile devices are operated in different situations. On account of this we involved two *support* conditions in our experiments. All participants performed the tests while seated, half of the tests with the device held in their non-dominant hand (Figure 2 a, c, d) and the other half with only the dominant hand involved and the device fixed to the table surface (Figure 2 b).

For every touch input we recorded the time of the contact event and the corresponding motion onset. The latter was defined as a deviation from the contact position larger than 1.7 mm (11 px). In informal tests this position threshold was found to eliminate involuntary motion input while maintaining the responsiveness of the interface. Dwell times were measured as the time difference between these two events.

For zooming gestures we computed the time difference between the contact events of both fingers. In the following we explain the experimental tasks and the involved variables in detail.

Dragging

Dragging an on-screen object always involves selecting it at its current position and moving it to the desired target position. We expected a short dwell time between these subsequent actions as both are aimed movements that require a certain amount of planning.

The *dragging* experiment consisted of a docking task that was designed in correspondence to the main menus of many touch-based mobile devices. A grid of gray squares was displayed with white outlines on a black background. One of these items was highlighted with a green fill color designating it as the object to be dragged. Another one or two squares with white fill color marked the docking targets (Figure 2a).

We randomized the number of target options (one or two) and also varied the grid size (6x6 or 4x4 items) as such variations may affect the duration of dwell times. Each participant performed 120 docking tasks in four consecutive blocks, with short breaks of about one minute in between. If an item had been released outside the target area, it jumped back to its original position. The task had to be repeated in this case, which resulted in additional dragging actions for our records.

Panning

Depending on the users' intentions, panning a large map or scrolling a page may expose different timing structures. If a user is searching for a salient target or if the target distance is known, the motion behavior corresponds to aimed movement in pointing tasks [9]. On mobile devices this type of scrolling task is generally accomplished with a series of rapid strokes (flicks).

If users are scanning the displayed graphics or reading a text instead, panning operations are rather slow and continuous [1]. We expected that this difference between panning operations would also affect dwell times. Thus, we designed two panning tasks, both of which consisted of vertically scrolling a text on the display.

In the *Search* condition we instructed the users to scroll down until they recognized a text passage highlighted with a blue rectangle on a light gray background. The marker was set to 3 cm in width and 1.5 cm in height, thus easily recognizable even if the screen content was moving rapidly (Figure 2b).

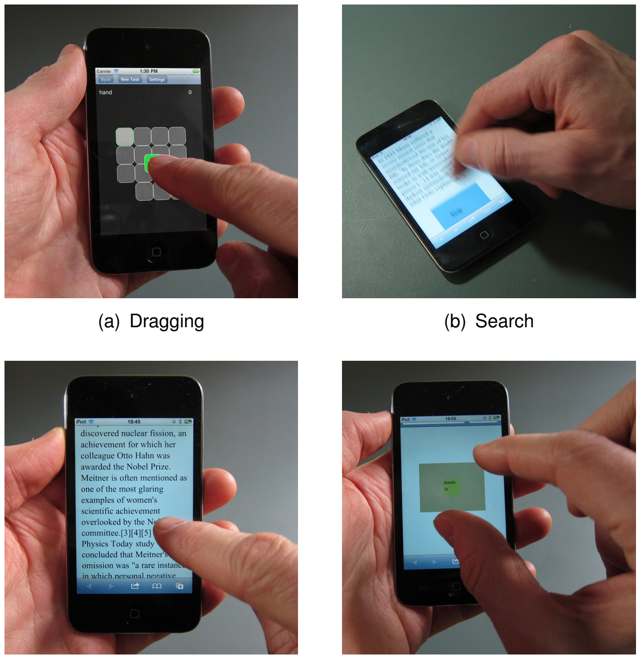
The *Read* condition on the other hand consisted of reading a short text of about 200 words (Figure 2c). Thereafter some related questions were asked in order to motivate the users to read the article thoroughly instead of only skimming it.

Panning inertia as provided by the operating system was enabled during all tests. The line height was always set to 7 mm on the screen, such that 13 lines were be visible at once. The tasks were not repeated but they required a series of panning actions for completion. In the *search* condition, the target was 120 lines away from the initially visible workspace area, in the *read* condition the text stretched over 51 lines.

Zooming

Currently, zooming is the most relevant gesture on mobile devices that involves more than one contact point. Many other meaningful gestures can be performed with two fingers. However, the gestural vocabulary for mobile touch devices can only be extended with input patterns that can be clearly differentiated from the established ones. Toward a better understanding of the pinch-zoom gesture, we designed another experiment to capture and identify its timing characteristics.

The screen showed two rectangles, a smaller one containing a short text nested inside a larger one (Figure 2d). The users were instructed to zoom in until the small rectangle fully filled the screen without cutting away any part of the contained letters. Then they were asked to zoom out until the minimal zoom level of 50% of the initial value was obtained and finally zoom back in to the initial state. This procedure had to be sequentially repeated for at least three times. Some participants did the task more often which provided us with a larger database for the evaluation.



(a) Dragging (b) Search

(c) Reading (d) Zoom

Figure 2. Experimental tasks for measuring the timing of common touch gestures. The *Search* task is shown in the *support* condition with the device fixed to the table surface. All others with the device held in the non-dominant hand.

Recorded Time Intervals

We recorded 1545 dragging actions, 315 panning actions in the *read* condition and 474 panning actions in the *search* condition. From this data we extracted dwell times between the contact event of the finger touching the screen surface and the corresponding motion onset. From the 618 recorded pinch-zoom gestures we derived the average time difference between the contact events of both involved fingers.

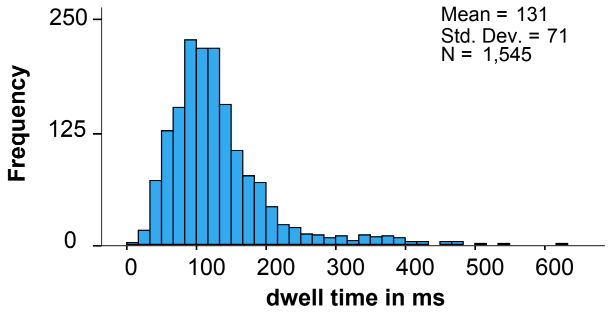


Figure 3. Histogram of dwell times during the *dragging* experiment

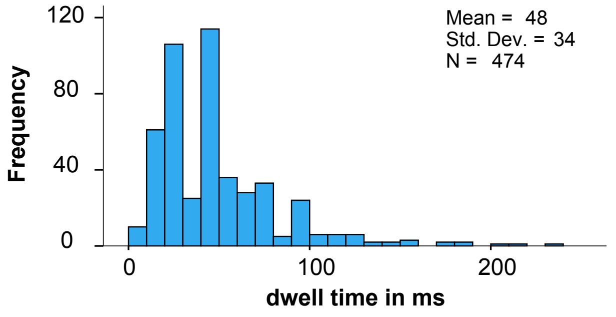


Figure 4. Histogram of dwell times during the *search* experiment

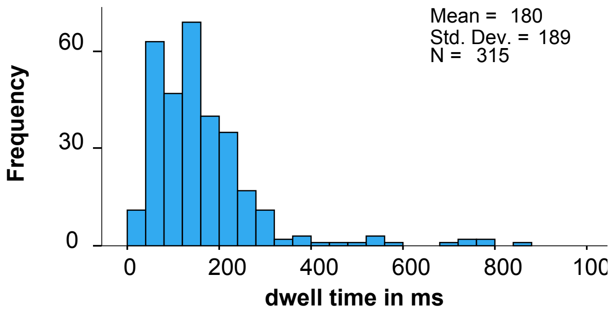


Figure 5. Histogram of dwell times during the *read* experiment

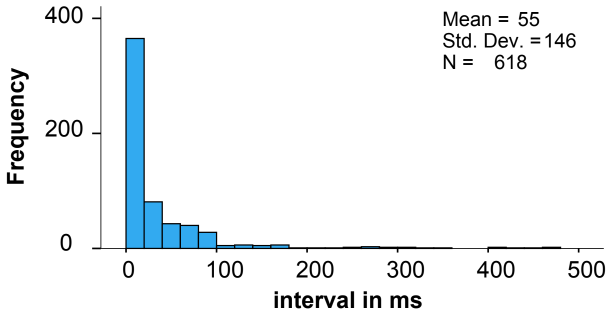


Figure 6. Histogram of time differences between the contact of both fingers with the screen surface when initiating a pinch-zoom gesture

The results from the dragging and panning tests revealed that dwell times are indeed an implicit characteristic of touch input (Table 1). We also observed that their duration is clearly different among the tested tasks. The average dwell time during dragging tasks was 131 ms (*sd*=71 ms) (Figure 3). For panning tasks we measured average dwell times of 48 ms (*sd*=34 ms) if participants were scanning for salient targets (Figure 4), but if they were asked to read a displayed text with diligence, average dwell times increased to 180 ms with a standard deviation of 189 ms (Figure 5).

Regarding the pinch-zoom gesture, we observed that both fingers generally touch the surface almost simultaneously. We excluded 19 outliers from our records with a time difference of more than 1000 ms. In the other 599 cases the time differences were under 500 ms with a mean of 55 ms and a standard deviation of 145 ms (Figure 6).

Design Considerations

It seems quite obvious to exploit dwell times for implicit mode switching since they are an implicit characteristic of dragging tasks. However, we measured much shorter time intervals than the commonly used threshold of 500 ms. Dwell times above 500 ms occurred naturally in only 3 of all recorded dragging actions. In the other 1542 cases the dwell time would have been disruptive. Thus, reducing the threshold bears the potential to increase the fluency of interaction. During more than half of all recorded dragging actions a dwell time longer than 100 ms occurred. Hence, this value could be a better suited dwell-time threshold - if interferences with panning actions can be avoided.

Our measurements indicate that this may be possible in tasks that require only rapid flicks to navigate through the displayed content. Panning actions in the *search* condition exposed with 48 ms on average the shortest dwell times in our experiments. 95% of all rapid panning gestures we recorded in this condition involved a dwell time shorter than 110 ms (see Table 1).

percentiles	**min**	**5%**	**median**	**95%**	**max**
	dwell times				
dragging	11	46	111	270	624
search	4	12	42	110	237
read	10	44	141	452	2032
	interval between two contact events				
zooming	0	0	18	143	478

Table 1. Percentiles of recorded time intervals during different input actions (all data in ms).

On the other hand, the data recorded in the *read* condition shows that dwell times during panning actions significantly increase if the task is cognitively more demanding. While reading a text our test users induced dwell times of 180 ms on average. This is longer than the dwell times recorded during dragging actions. While reading, a threshold of 110 ms would have been exceeded in 63% of all recorded panning actions. In order to avoid involuntary mode switching for at least 95% of all the panning actions in the *read* condition, the threshold would have to be set to a value above 452 ms. This corresponds to the Apple® design guidelines for long press gestures [2]. The default duration here is 500 ms. In our recorded data, this interval was exceeded in only 1% of all panning actions. We conclude that the common dwell time threshold of 500 ms is well adjusted to reduce the chance of involuntary activation. We also note, however, that it frequently interferes with the interaction flow during object manipulation.

In our study the pinch-zoom gesture exhibits a highly symmetric initiation pattern for most cases. The interval between the contact events of both fingers was less than 100 ms in 93% of the cases. The 19 excluded outliers show that pinch zoom is not necessarily initiated symmetrically, but in more than 98% of our records it was. We conclude that any input from two fingers that touch the screen with a time difference of more than 100 ms is most probably not meant to initiate a pinch-zoom gesture. It could thus be interpreted differently.

THE HOLD-AND-MOVE INPUT PATTERN

Li et al. [13] found that asymmetric bi-manual interaction can be an advantageous alternative to the disruptive long tap. Following the principles of Guiard [8], high-level motion input like navigation ought to be assigned to the non-dominant hand while the dominant hand performs object manipulation within the provided reference frame. Guiard also observed precedence of action of the non-dominant hand as a general principle of asymmetric bi-manual activities. We suggest that this principle can be directly exploited for the interpretation of multi-touch input.

We implemented the hold-and-move gesture as follows: The first finger is always assigned to *hold* the background (Figure 1a). A second touch input is used to select and *move* individual items relative to the background if the touch occurs at least 100ms after the first touch event (Figure 1b & c). In the following, we refer to this time parameter as the hold time – as opposed to the dwell time to discern a long tap. If both touch events occur within less than 100ms their input is interpreted as a pinch-zoom gesture instead. An interaction sequence of hold and move only ends when both fingers release the touch sensitive surface. The assignment of fingers to background or foreground thus remains in effect if only one finger is released, which naturally enables clutching. Long-distance panning operations can be controlled with successive input of one finger while the other one keeps holding a dragged item. (Figure 1d).

The mode switching technique clearly consists of two steps: holding the background followed by the selection of a foreground item. Hence, it should be easily distinguishable from the pinch-zoom gesture. Recall that we measured intervals shorter than 100 ms between both touch events in 93% of the 618 recorded pinch-zoom gestures. During this initial data gathering, only a single input mode had to be operated. We expected that the awareness of a second two-finger input mode would further increase the accuracy of the symmetric pinch-zoom initiation.

Target-based zooming is still available alongside with hold-and-move. Only the initiation of the input gesture matters for the differentiation between symmetric input like zooming (almost simultaneous contact of both fingers) and asymmetric input like hold-and-move (successive touch of both fingers). Thereafter, motion input may be induced symmetrically or asymmetrically.

We identify the following main advantages of using hold-and-move for mode switching:

- It eliminates involuntary object selection during panning as well as workflow interruption during intended object manipulation.
- The two-handed technique omits the need for automatic scrolling to drag an item beyond the visible screen area.
- It facilitates successive object manipulations. Once the background is attached to one finger, several objects can be sequentially dragged with another finger.

USER STUDY

We evaluated the performance and user acceptance of both the long-tap and the hold-and-move mode switching techniques in two test applications using an Apple iPod® touch 4 device. Both test applications were focusing on object dragging. Only little workspace panning and no zooming was required for task completion. Nevertheless, both types of workspace navigation were continuously available during all tests in order to observe potential issues of interference: between hold-and-move and zooming as well as between object selection with the long-tap and panning.

Hold-and-move was implemented as described above with a 100 ms hold time threshold to discern the zooming functionality. The *long-tap* implementation used the common dwell time of 500 ms and provided autoscroll functionality that was actuated if the fingertip was less than 4 mm away from the screen border.

Experimental Tasks

Our two test scenarios (*List* and *Grid*), both consisted of several object manipulation tasks that also required some navigation input. The basic activity of moving icons or pictures across the screen was implemented as a color association task. The users were asked to drag colored items (red, green, blue, gray) to corresponding text fields naming the color.

If a dragged object was released on the incorrect target position, it was automatically moved back to its origin. After having moved a matching item to each visible target field, one set of tasks was accomplished and another version was loaded, proffering a different sorting of targets and pieces to be dragged.

Note that the association from color to text added a constant cognitive load to the task. This was meant to reproduce a realistic situation in that users focus on accomplishing a certain task rather than the correct operation of the interface.

Grid

The color and text items were presented in a four by four grid structure (Figure 7 left). The first and third row each contained the four differently colored items. For each item there was a corresponding target field showing the name of the respective color in the line below. Besides this fixed vertical structure all items were placed randomly on the horizontal axis such that the users had to sort them while performing the task. Eight color items had to be dragged to the corresponding text fields on the line below. The subtask sequence was accomplished when all 8 items had been placed correctly. Then the next sequence was loaded.

List

The color items were presented in a list structure. The first three entries on top of the list always consisted of the color items red, green and blue in randomized order (Figure 7 right). Below, further entries contained gray items. All color items could be selected and moved across the screen, but only one red, green or blue item had to be dragged to the target position indicated by the word naming the respective color. We included seven different target distances (1–7 steps down in the list with Fitts' IDs ranging from 0.3–1), each appearing twice in random order. For the larger distances 5–7 the target position was sometimes too far away from the source to see both simultaneously on the screen. The users had to navigate to the target position in order to identify which item to drag there. Dragging the item to the target position correspondingly involved navigation. This second input mode was provided simultaneously in the *hold-and-move* condition or with autoscroll in the *long-tap* condition.

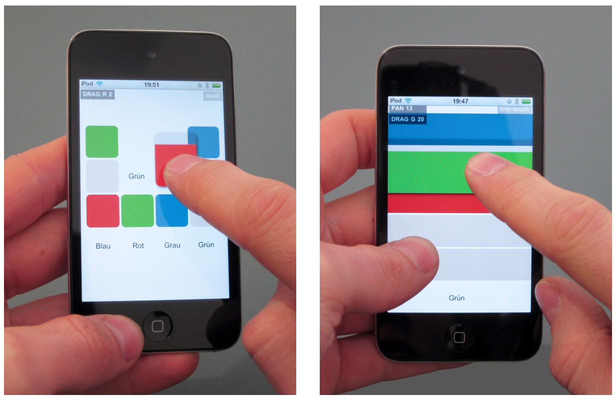


Figure 7. The test scenarios *Grid* (with *long-tap*) on the left and *List* (with *hold-and-move*) on the right.

Participants

20 people participated in the study, four of which were female, 16 male. The age ranged from 20 to 33 with a mean of 24.75. Six of the 20 participants claimed to use touch-screen devices on a daily basis, while 3 claimed to have no experiences with them. On a 5 point scale ranging from no experiences (1) to daily use (5) the mean was 3.77.

Design and Procedure

The participants were advised to hold the device in portrait orientation and in their non-dominant-hand. With each technique every participant performed six sets of eight dragging tasks in the *Grid* condition and thereafter 14 dragging tasks in the *List* condition. The order of techniques was counterbalanced. Each session started with a learning phase involving 6 dragging actions in a more holistic chess task that required also a lot of panning and zooming to manipulate the small chess pieces. No data was logged during the chess task, but it served the participants to subjectively explore the usability of the respective techniques. After all tests had been completed, we asked the participants to rate the suitability of both techniques for each scenario.

Hypotheses

We assumed that users would complete the dragging tasks faster in the *hold-and-move* condition – not because the dwell time in the *long-tap* condition is a disadvantage by itself but rather due to the users' difficulties to cope with the interruption of their workflow. In the *Grid* condition the dragging tasks were presented in sets of eight. This allowed users to take advantage of activity planning. In particular they could minimize the effort in the *hold-and-move* condition. Holding the background once enabled the object manipulation mode for all consecutive dragging actions. We thus expected a bigger advantage for *hold-and-move* in this scenario.

Results

Data was collapsed and entered into a 2 (*scenarios*)×2 (*techniques*) analysis of variance. Normality was verified using the Kolmogorov-Smyrnov test. We found significant main effects for task ($F_{(1,19)} = 56.0, p < 0.01$) and technique ($F_{(1,19)} = 62.7, p < 0.01$) as well as a significant interaction between these two independent variables ($F_{(1,19)} = 8.6, p < 0.01$). Post-hoc comparisons using Tuckey's HSD test revealed significant differences between the tested *techniques* in both *scenarios* ($p < 0.01$ for the *Grid* condition and $p < 0.05$ for the *List* condition).

As can be seen in table 2 the performance difference between both *techniques* is more pronounced in the *Grid* condition. A more detailed review of the data revealed that these additional benefits are indeed related to the improved integration of subsequent manipulation tasks. We observed that users were setting a new reference with an initiating touch event only twice within one set of the eight dragging tasks – just before starting to manipulate another group of four items in one line. Hence, the competitive edge for our novel gesture is less pronounced if we compare both techniques only in the first and the fifth dragging subtask: 2119 ms (*sd*=1267 ms) for *hold-and-move* vs. 2940 ms (*sd*=1565 ms) in the *long-tap* condition.

The impact of the target distance in the *List* condition was visible in the results, but no difference between *techniques* could be found in that regard. Overall we found considerable performance advantages of the *hold-and-move* technique as compared to the *long-tap* interface in both test scenarios (Figure 8).

	Grid	**List**
long-tap	2866 (1459)	3459 (2151)
hold-and-move	1366 (991)	2651 (2395)

Table 2. Mean task completion times in ms for both tested *techniques* (lines) in the two usage *scenarios* (columns). Standard deviation in parenthesis

Subjective user ratings reflect the quantitative results. The average usability rating of the two techniques on a 5-point Likert scale was similar with a mean of 4.0 for *hold-and-move* and a mean of 3.7 for the dwell times, indicating that both

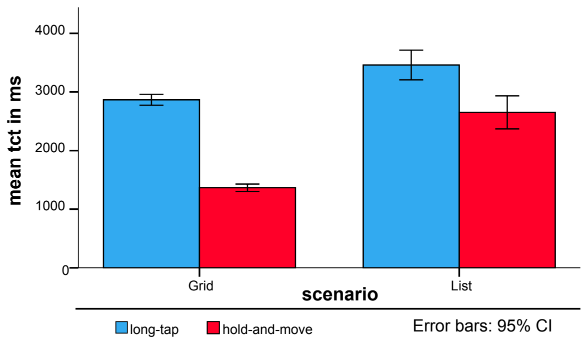


Figure 8. Mean task completion times in ms.

techniques were well suited for the given tasks. Most users had a clear preference for one of both techniques. Ten users preferred *hold-and-move* to operate the chess game while six voted for the *long-tap*. For the *List* sorting task eleven users preferred *hold-and-move* while seven users preferred the *long-tap*. Regarding the *Grid* task, many users appreciated the benefits of *hold-and-move* for successive object manipulation. Sixteen of 20 users preferred the novel technique in the *Grid* condition while three voted for the *long-tap*. Only few users were undecided in some scenarios.

Error Analysis

Different types of erroneous input may occur with the two mode-switching techniques at test. Dwell-time-based object manipulation may interfere with other motion input from one finger, e.g. panning. Hold-and-move, on the other hand, can interfere with zooming that is also operated with motion input from two fingers. In both cases, the correct interpretation of the user's input critically depends on the timing.

Quantifying such mode switching errors is not trivial. Without a predefined input sequence, we cannot easily know which mode changes were intended and which were not. Additionally, we must consider false positives (the mode changed although not intended) and false negatives (the intended mode change was not achieved).

False Positives

From our initial study on the timing of touch gestures we derived that a dwell time of 500 ms is a well suited threshold to minimize false positives during panning. Only 14 of the 789 recorded panning actions involved a dwell time longer than 500 ms.

We also estimated the rate of false positives for the hold-and-move gesture from the records of pinch-zoom gestures we gathered during this initial study. In 7% of the 618 recorded pinch zoom gestures, both fingers touched the screen surface with a time difference longer than our threshold of 100 ms. They would have had invoked the object manipulation mode although their input was meant to control the zoom level. We expected, though, that users would initiate the pinch-zoom gesture more accurately with both fingers simultaneously if they are aware of a second two-finger gesture that is initiated asymmetrically instead.

During our second study comparing hold-and-move to dwell-time based mode switching we recorded 483 hold-and-move gestures, 446 of them were completed with the successful docking of one or more dragging items. In the other 37 cases the gesture may have been initiated involuntarily. This is plausible if both touch events occurred within the range of time intervals we measured for pinch-zoom gestures. We define this range based on the average interval we measured for pinch-zoom gestures plus three times standard deviation (55 ms $+ 3 * 145$ ms $= 493$ ms). In 18 cases of our 37 error candidates both fingers touched the screen within a shorter interval, which would correspond to a rate of 3.7% false positives for a hold-time threshold of 100 ms.

False Negatives

Towards an estimate of false negatives, we can compare the frequency of "unnecessary" input actions. Most of our experimental tasks did not require panning or zooming as the source and the target position were displayed on visible screen area. Any panning or zooming would thus be inefficient and can be interpreted as an error.

In the *Grid* task users could benefit from the integration of up to eight subsequent dragging operations in one hold-and-move sequence while the long-tap involves a mode selection for each subsequent dragging operation. Thus, we only consider the first dragging action from each of these sets for the error analysis. Some of the *List* tasks involved target positions off screen and therefore required panning or zooming. This should have the same impact on the performance with both techniques, but to account for this difference we distinguish short and long target distances in the analysis. Distances of up to 4 rows were always defined to be short since the targets were always visible without navigation.

Zooming occurred rarely in all tasks and with both techniques (see Table 3). The slightly higher number of zooming actions in the *hold-and-move* condition, indicates that involuntary input may have occurred, but none of the small differences proved to be statistically significant. We conclude that hold-and-move did not had a relevant impact on the operability of pinch zoom during our experiments. Although we applied a rather low threshold of 100ms, only one of 20 users noticed the possibility of interfering with pinch-zoom input.

	Grid	**short List**	**large List**
	zooming input		
long-tap	0.13 (0.47)	0.12 (0.49)	0.26 (0.64)
hold-and-move	0.13 (0.59)	0.07 (0.28)	0.19 (0.61)
	panning input		
long-tap	1.62 (1.01)	1.66 (0.95)	2.98 (1.405)
hold-and-move	0.58 (1.59)	0.42 (0.97)	2.08 (3.028)

Table 3. Average numbers of panning and zooming actions per dragging task. Standard deviation in parentheses.

For mode selection based on dwell times false negatives are indicated by an increased amount of panning actions. We observed that the average number of panning actions per docking task is more than two times higher in the *long-tap* condition as compared to *hold-and-move* (see table 3). A MANOVA reveals that the rate of panning actions differs significantly between techniques ($F_{(1,19)} = 25.68, p < 0.01$) and also between the three groups of tasks ($F_{(2,38)} = 30.06, p < 0.01$), while the interaction between both factors is not significant. We conclude that indications of input errors related to dwell times can be found in all task conditions. User feedback supports this observation. Five participants explicitly complained about being interrupted by dwell times.

Hold-Time Analysis

With a minimal hold-time threshold of 100 ms, false negatives do not seem to occur at all. Instead, we observe a chance to invoke the object manipulation mode involuntarily. This risk can be reduced by maximizing the applied threshold. Our records from 446 successful hold-and-move gestures, indicate that this is a feasible approach. The average hold time was 658 ms (*sd*=485 ms). Interestingly, this is longer than commonly applied dwell times of about 500 ms, which confirms our assumption that the reason for the inferior performance of the *long-tap* is not the dwell time itself. User feedback indicates that the drawback rather results from the difficulties to cope with the workflow interruption.

Only 5% of the hold-and-move gestures were executed with a hold time of less than 190ms (Table 4). We reasoned that such extremely short hold times only occurred because our study was focused on consecutive dragging tasks. Consequently, we suggest a hold-time threshold above 150ms (corresponding to the 95% percentile value of pinch-zoom input) and below 190ms (corresponding to the 5 percentile value of hold-and-move gestures). Depending on the target user group and different applications, other values may offer a better suited balance (see figure 9).

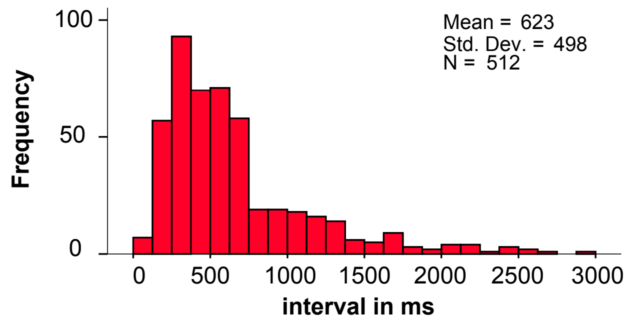


Figure 9. Histogram of recorded time differences between the contact of both fingers with the screen when initiating a hold-and-move input gesture.

percentile	**min.**	**5%**	**median**	**95%**	**max.**
hold time	103	189	514	1681	2927

Table 4. Hold time extrema (all data in ms).

Discussion

The results of our user study demonstrate that the hold-and-move gesture is an efficient alternative to the disruptive long-tap. Note that we tested with a very basic implementation of the technique. In practice, we expect that applications will take further information into account. Most obviously, context information should be included as known from using the long tap for object selection, which is only executed if the finger rests on top of a selectable item. In applications with a sparse distribution of selectable items, this simple adaptation would directly enable to operate zooming even after asymmetric initiation.

Compared to the long-tap, hold-and-move has the limitation in that it cannot be operated using only the thumb when holding the device in one hand. This will be less of a restriction for larger devices that are generally not operated single-handedly, like tablet devices. In addition, there is no technical reason that one-finger dragging based on dwell time cannot be made available in the interface in addition to the two-finger hold-and-move gesture.

It is important to consider that the normal mode of interaction for both techniques is navigation via panning and scrolling. We tested them both in manipulation-focused tasks. We expect that hold-and-move is beneficial also for navigation tasks that do not require a secondary input modality, since it prevents accidental mode switching (false-positives). Hold-and-move demands a second touching finger and cannot be triggered involuntarily in single-touch navigation (e.g. if leaving the finger on the screen while reading).

CONCLUSIONS AND FUTURE WORK

We designed the two finger object dragging pattern hold-and-move to avoid the disruptive long-tap in touch interface implementations. Our approach involves a second finger to enable the smooth execution of object selection followed by object dragging. With the novel technique regular workspace panning remains available during dragging. As a side effect, the cumbersome and hard-to-control workspace autoscrolling function is no longer needed when dragging an object beyond the currently visible viewport. Involuntary object selection during panning is also eliminated. Our user study revealed that the hold-and-move pattern allows for faster object dragging than the conventional dwell-time approach and is preferred by most users. We also showed that the novel gesture can be applied without affecting the recognition quality of the established single-finger panning and pinch-zoom gestures.

Pinch zooming mimics squeezing and stretching the screen content. Consequently, the finger motion is induced along the vector between both initial touch positions. Instead, for hold-and-move motion input is induced in the direction of a target position that is independent of the orientation of the initial touch positions. This observation could be used to further reduce involuntary mode changes. The implementation of the hold-and-move pattern could also benefit from the possibility to adapt the thresholds to the user and from context sensitivity. For example, if there are no objects near the second finger that can be dragged, the user is likely to start a pinch zoom even though the threshold for enabling this gesture has been exceeded.

While object dragging may be only rarely required in common mobile phone applications, the mode-switching technique enables many other functionalities that are similarly operated. Selecting snippets of text and images from websites and pasting them into a new document is currently a cumbersome task. Hold-and-move bears the potential to operate such tasks with comparable productivity as with the mouse in desktop environments. Furthermore, it provides a reliable basis for drawing-based interaction techniques like Gesture Avatar [14].

Our investigations into the timing structure of common multi-touch gestures and of our hold-and-move pattern provide a solid basis for designing and implementing further multi-touch gestures that do not require any additional hardware. Such gestures can be easily integrated into multi-touch enabled operating systems for mobile devices, which would extend the slowly evolving multi-touch language.

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF) under grant 03IP704 (project Intelligentes Lernen). We thank the participants of our studies and the members and students of the Virtual Reality Systems group at Bauhaus-Universität Weimar (http://www.uni-weimar.de/medien/vr) for their support.

Many thanks go to Felicitas Höbelt and Daniel Fischer who provided the software for the test applications. We also thank the participants of our studies and the reviewers of this paper for their helpful comments.

REFERENCES

1. Andersen, T. H. A simple movement time model for scrolling. In *Ext. Abstracts CHI 2005*, ACM Press (2005), 1180–1183.
2. UILongPressGestureRecognizer class reference. http://developer.apple.com/library/ios/#documentation/uikit/reference/UILongPressGestureRecognizer_Class/Reference/Reference.html.
3. Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. Toolglass and magic lenses: the see-through interface. In *Proc. Siggraph 1993*, ACM Press (1993), 73–80.
4. Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proc. AVI 2008*, ACM Press (2008), 154–161.
5. Buxton, W., and Myers, B. A study in two-handed input. In *Proc. CHI 1986*, ACM Press (1986), 321–326.
6. Cutler, L. D., Fröhlich, B., and Hanrahan, P. Two-handed direct manipulation on the responsive workbench. In *Proc. I3D 1997*, ACM Press (1997), 107–114.
7. Epps, J., Lichman, S., and Wu, M. A study of hand shape use in tabletop gesture interaction. In *Ext. Abstracts CHI 2006*, ACM Press (2006), 748–753.
8. Guiard, Y. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. In *Journal of Motor Behavior*, vol. 19 (1987), 486–517.
9. Hinckley, K., Cutrell, E., Bathiche, S., and Muss, T. Quantitative analysis of scrolling techniques. In *Proc. CHI 2002*, ACM Press (2002), 65–72.
10. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. Manual deskterity: an exploration of simultaneous pen + touch direct input. In *Ext. Abstracts CHI 2010*, ACM Press (2010), 2793–2802.
11. Kabbash, P., Buxton, W., and Sellen, A. Two-handed input in a compound task. In *Proc. CHI 1994*, ACM Press (1994), 417–423.
12. Lank, E., Ruiz, J., and Cowan, W. Concurrent bimanual stylus interaction: a study of non-preferred hand mode manipulation. In *Proc. Graphics Interface 2006*, CIPS (2006), 17–24.
13. Li, Y., Hinckley, K., Guan, Z., and Landay, J. A. Experimental analysis of mode switching techniques in pen-based user interfaces. In *Proc. CHI 2005*, ACM Press (2005), 461–470.
14. Lu, H., and Li, Y. Gesture avatar: a technique for operating mobile user interfaces using gestures. In *Proc. CHI 2011*, ACM Press (2011), 207–216.
15. Matejka, J., Grossman, T., Lo, J., and Fitzmaurice, G. The design and evaluation of multi-finger mouse emulation techniques. In *Proc. CHI 2009*, ACM Press (2009), 1073–1082.
16. Olwal, A., Feiner, S., and Heyman, S. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *Proc. CHI 2008*, ACM Press (2008), 295–304.
17. Roth, V., and Turner, T. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proc. CHI 2009*, ACM Press (2009), 1523–1526.
18. Roudaut, A., Lecolinet, E., and Guiard, Y. Microrolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proc. CHI 2009*, ACM Press (2009), 927–936.
19. Siio, I., and Tsujita, H. Mobile interaction using paperweight metaphor. In *Proc. UIST 2006*, ACM Press (2006), 111–114.
20. Wu, M., and Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proc. UIST 2003*, ACM Press (2003), 193–202.
21. Yee, K.-P. Two-handed interaction on a tablet display. In *Ext. Abstracts CHI 2004*, ACM Press (2004), 1493–1496.