The Design and Evaluation of Multi-Finger Mouse Emulation Techniques

Justin Matejka, Tovi Grossman, Jessica Lo, George Fitzmaurice
Autodesk Research
210 King St. East, Toronto, Ontario, Canada, M5A 1J7
{firstname.lastname}@autodesk.com

ABSTRACT
We explore the use of multi-finger input to emulate full mouse functionality, such as the tracking state, three buttons, and chording. We first present the design space for such techniques, which serves as a guide for the systematic investigation of possible solutions. We then perform a series of pilot studies to come up with recommendations for the various aspects of the design space. These pilot studies allow us to arrive at a recommended technique, the SDMouse. In a formal study, the SDMouse was shown to significantly improve performance in comparison to previously developed mouse emulation techniques.

Author Keywords
Multi-finger input, multi-touch displays, mouse emulation.

ACM Classification Keywords
H.5.2 [User Interfaces]: Interaction styles.

INTRODUCTION
The recent release of several commercial multi-touch systems, such as SMART's Table and Microsoft's Surface, has created a great deal of excitement surrounding multi-touch input in both the public and research communities. Multi-touch interfaces offer a new modality of interaction, providing a unique usage experience [18, 20, 29].

Unfortunately, multi-touch input raises a number of interesting yet difficult challenges when it comes to the design of user interfaces. Multi-touch input generally assumes direct input, where the user interacts with graphical imagery directly under the points of contact. Due to this direct interaction, it suffers from the same drawbacks as traditional single point touch screens. For example, direct interaction may cause fatigue, reduced precision, and occlusions from the user's hand. Research targeting single point touch screens, has attempted to address these issues [1].

More recently, researchers have explored leveraging the extra input provided by multi-touch systems to seek out new solutions and strategies to the difficulties caused by direct touch interaction. For example, Benko et al. used two fingers and various on-screen widgets to improve the precision of the control over a cursor [3].

However, even if the precision, occlusion, and fatigue problems are solved, almost all research to date supporting direct touch interaction has only considered supporting a "left click" event. Receiving less attention is that to properly interact with many graphical interfaces, the right and middle buttons, are also desirable and in some cases essential. Just as Benko argues that a tracking state should be made available from a multi-touch system [3], we argue that right and middle clicks should be supported.

In an effort to address this challenge, we explore "full" mouse emulation techniques, which support the functionality of a 3-button mouse. We first describe design considerations for mouse emulation techniques, and then a design space which allows us to systematically explore various configurations supporting mouse emulation. Through a series of pilot studies, we converge on the *SDMouse* as a recommended mouse emulation technique (Figure 1). In a formal study, SDMouse outperforms previously proposed mouse emulation techniques, while possessing a number of beneficial design properties.

MOTIVATION
In some sense, emulating a mouse on a multi-touch display may seem to defeat the purpose of having a multi-touch system in the first place. However, we feel it is an important issue to address, since it is likely that only a minority of today's end user applications will be completely rethought and re-engineered to provide user experiences specifically tailored to multi-touch input. What is more

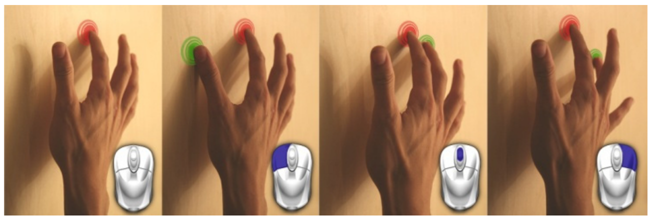
Figure 1. The SDMouse uses multi-finger input to emulate the functionality of a 3-button mouse.

plausible is that while some applications will fully support multi-touch interaction, others will only possess specific modes supporting multi-touch, and some will not change at all, and will consist of only traditional GUI elements.

As an example, consider a user who is scaling and rotating photos using multi-touch interaction, and then decides to email the photos to a friend. The user may be required to switch to their email client, which may not support multi-touch interaction. Or maybe the photo browsing application contains a traditional GUI menu for emailing the photo to a friend. In either case, the user would be left frustrated if they were required to perform this operation without the aid of a mouse emulation technique. TabletPC users could relate to this difficulty, as only a minority of applications which run on TabletPCs have had their user interfaces tailored specifically for stylus input, resulting in numerous frustrating interaction issues [13].

RELATED WORK

A great deal of research has been conducted on multi-point, touch sensitive surfaces [5, 18, 20]. A variety of technologies have been used to sense multi-finger and hand input such as camera based solutions [16, 17, 19, 28, 31] and capacitive sensing [6]. While we have attempted to factor out the sensing technology, it still can impact the robustness of a solution.

Numerous challenges occur when trying to operate a high resolution graphical user interface with fingers that obscure the underlying data. Various research projects attempt to address these problems [1, 3, 24, 27]. Our explorations are compatible with many of these techniques.

The appeal of multi-point input is that it can be used to provide a richer set of inputs to interactive systems. This input can be used to affect and manipulate the data directly or to enhance traditional GUI elements such as modifying the cursor [21, 22]. Multi-point input has been explored within 3D volumetric display [14] and still other research explores combining multi-finger input with pen input [4].

Using multiple hands to control cursor input has also been investigated [3, 22, 29, 30]. Our investigation focuses on providing a one-handed solution so that the other hand can be used for additional interactions.

Some hardware configurations sense the finger position while above the touch surface [7]. For example, the SmartSkin system [25] provides basic mouse emulation by sensing the height of the hand above the touch surface (the tracking state) as well as detecting finger contact which generates a left button-down event. It also explores the use of the palm of the hand as a trigger. While multipoint sensing is possible, they do not describe how to support full mouse emulation. In addition, we want to provide a solution for systems that do not have the ability to sense when fingers are in the tracking state.

Yet other systems attempt to artificially create a tracking state by sensing surface contact area [3] where a small contact area is interpreted as the tracking state and larger contact area as part of the clicking and drag state. Directional finger rubbing has also been studied to trigger more continuous input events [23]. However this would likely interfere with cursor position or drag events.

Comparisons between unimanual direct touch and mouse input indicate roughly equivalent performance times for selection tasks [12]. While some benefits may be realized by adopting a bimanual selection technique, we focus on unimanual solutions.

Most systems that offer partial mouse emulation, do not offer the ability to position the cursor without triggering a left mouse button event [24, 26]. That is, they do not support the input tracking state. The DTMouse [8] does offer partial mouse emulation, but the technique introduces modes and timeout periods which could impede on certain interactions. The now defunct FingerWorks system [9] offers a full mouse emulation design, but requires awkward chordings which may be difficult to learn and awkward to use. We will compare SDMouse to both of these existing emulation techniques.

DESIGN CONSIDERATIONS

There are a number of design goals to consider when developing a multi-touch emulation of a mouse. Here we discuss the functionality which we wish to support, and then the design properties we wish to uphold.

Functionality Support

Despite its simplistic design, the mouse is actually capable of providing numerous forms of input. Here we outline those forms of functionality which we consider.

Tracking State. The technique needs to support cursor positioning without needing to trigger a mouse button.

Three Buttons. We wish to support the left, middle, and right buttons, for both clicking and dragging operations. While some mice have additional buttons, they are rarely required for application use, and used mostly for shortcuts, and so we will not attempt to support them.

Multiple Button Chording. Many applications also require simultaneous use of multiple mouse buttons (chording), which we will make an effort to support.

Scroll Wheel. We also consider the functionality of the scroll wheel to be a shortcut, and do not support it. Furthermore, in many applications the middle button provides scrolling. However, we do discuss ways to augment our designs with scrolling in the future work section.

Design Properties

The mouse possesses numerous subtle properties which have made it such a successful device [2]. In emulating the mouse, we hope to achieve as many of these properties as possible. Doing so on a multi-touch surface introduces further design properties to consider:

Minimal Fatigue. The technique should minimize physical and mental discomfort.

Precision. The technique should allow for precise input.

Visibility Support. The technique should minimize the effect of occlusions caused by the hand.

Edge support. The technique should allow users to position and use the cursor along the display space edges.

Intuitive mapping. The design should relate to the physical mouse layout so that it is easy to initially learn and subsequently remember.

Fewest touch points. The technique should use as few touch points as possible to improve comfort and minimize friction when dragging.

Scale Independent. The technique should work regardless of the size of the user's hand.

Orientation Independent. The technique should work regardless of the hand orientation relative to the display.

Timing Independent. The technique should not rely on timeout periods, which may impede fluidity.

DESIGN SPACE

In this section we present the design space for developing multi-finger mouse emulation techniques. This design space was generated by considering various techniques which have been previously used to accomplish mouse activities. By defining this space of possible designs, we will be able to systematically explore potential emulation techniques.

Mapping

Mapping refers to how the cursor is positioned in relation to the point of contact with the touch screen. The following methods of mapping were investigated:

Direct

A direct mapping is the most traditional form of input for touch screens [5]. The cursor is placed directly where the tracking finger touches the screen. While intuitive, a direct mapping causes the finger to occlude the cursor, and selections may become difficult as targets become smaller.

Offset

An offset mapping positions the cursor slightly above the position of the finger [1, 24, 26]. This prevents occlusions. However, this sometimes causes users to do guesswork with where the cursor will be placed, and the bottom of the screen may not be accessible.

Scaled Absolute

The problem of accessibility of the screen can be solved with a scaled absolute mapping, in which the cursor is mapped from a smaller rectangular portion of the screen onto the entire screen. Each point on the smaller rectangle then has a corresponding point on the full screen, thus allowing for all points of the screen to be accessible.

Relative

Relative mapping also allows for the entire screen to be accessible and may give a sense of familiarity to the user since it works like a mouse. The cursor moves in the general direction of the movement of the finger(s) and is not dependant on where your fingers are on the screen. Clutching is then possible and therefore the cursor can reach anywhere on the screen and the hand can be repositioned to avoid occlusion of the cursor.

Tracking Fingers

Tracking fingers refers to the number of fingers required to move the cursor in the mouse-over status.

One

When using one finger for tracking, the mapping functions are applied to the point at which that one finger makes contact with the touch screen.

Two

An alternative which has been used in both single [8], and bimanual [3] techniques, is to use two fingers for tracking, with the cursor placed at the midpoint of the two fingers. Previous research has implied that using two fingers provides *Visibility Support* and *Precision* [8, 22]. In our work we will only consider single handed techniques, reserving the second hand for other operations.

Button Distinction

Traditional multi-touch systems do not distinguish which finger has made contact with the surface. Thus, to design the mouse emulation techniques, we considered the following characteristics for distinguishing between left, right and middle buttons events:

Chording

Chording refers to using the number of fingers in contact with the screen to delineate which button should be activated. Both FingerWorks and DTMouse utilize chording. Ironically, techniques which utilize chording recognition, cannot support mouse button chording.

Side

The Side technique determines the button based on which side additional fingers are placed relative to the tracking fingers.

Distance

Buttons can also be recognized depending on the distance at which the additional fingers are placed in relation to the tracking fingers.

Gesture

Alternatively, the fingers can be used to perform gestures to specify which button should be activated.

Button Activation

Button activation reflects how the button-down and button-up events are actually initiated.

Momentary

In momentary activation, a button-down event is registered as soon as the user contacts the touch screen with a finger or fingers. By releasing the finger or fingers from the screen, the corresponding button-up event is activated.

Toggle

In toggling, generating a button-down or button-up event occurs after a tap has occurred (the finger touches the screen and is lifted from the screen). If a button has been activated, tapping the screen would register a button-up event. Otherwise, if it has not been activated, tapping the screen would activate a button-down event.

Pressure

A third potential button activation method is to use pressure. Pressing firmly on the display activates a button-down event, and releasing registers a button-up event [3]. Because most multi-touch systems do not provide accurate pressure information, we will not explore the use of pressure.

	Scale Independent	Orientation Independent	Timing Independent	Allows for button chording	Maximum Touch points
Chording	✓	✓	×	×	4
Sides	✓	×	✓	✓	3
Distances	×	✓	✓	✓	2
Gesture	×	×	×	×	2
Side+Chording	✓	×	×	×	3
Side+Distance	×	×	✓	✓	2
Chording+Distance	×	✓	×	×	3

Table 1. Properties of Finger-to-Button Mappings.

FINGER-TO-BUTTON MAPPINGS

Previous emulation techniques [8, 9, 22] have used various finger-to-button mappings, utilizing a combination of different button distinction techniques outlined above. We develop potential techniques in a more systematic nature. By utilizing the above-described button distinction aspect of the design, we developed four techniques using only one button distinction method (Chording, Side, Distance, Gesture) as well as three hybrid methods (Side+Chording, Side+Distance, and Chording+Distance). For simplicity the below descriptions assume single finger tracking, using the index finger, except for the Sides Technique, which requires two finger tracking. Other than this technique, the techniques could all be implemented using one or two finger tracking. Human factors research has shown that movements of the thumb, index finger and little finger are more highly individuated than movements of the middle or ring fingers [15]. If possible, each of the described techniques should be implemented with this in mind. Table 1 summarizes the design properties associated which each technique. Since none of the proposed solutions satisfy all of the desirable design characteristics, the challenge becomes satisfying a suitable subset of them.

Chording Technique

The Chording Technique uses the idea of chording to specify the left, right, or middle buttons. One additional finger specifies a left mouse button event; two additional fingers specify a middle mouse button; and three specify a right button event (Figure 2). The placement of the non-tracking fingers is irrelevant (scale and orientation independent). However, a timeout is needed to determine how long to wait for additional fingers before registering the first one as a left click. We used a timeout of 150ms.

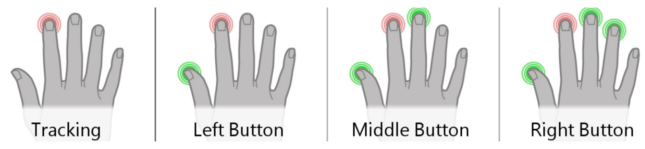

Figure 2. The Chording Technique.

Side Technique

In order to use only side information to determine which button to activate, the index and ring fingers must be used for tracking. The thumb would activate the left button, the pinky would activate the right button, and the middle finger would activate the middle button (Figure 3).

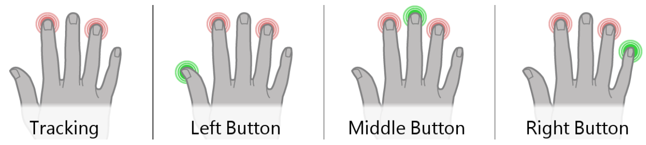

Figure 3. The Side Technique.

Distance Technique

The Distance Technique defines a short (< 150 px), medium (150-250 px) and far (> 250 px) distance to the right of the index finger, for activating the left, middle, and right button. The middle finger, ring finger, and the pinky finger are used to activate the three buttons (Figure 4).

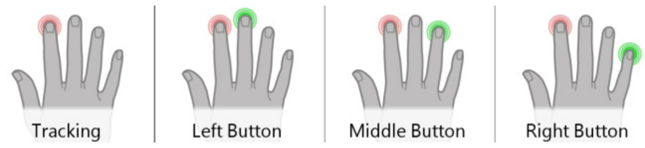

Figure 4. The Distance Technique.

Gesture Technique

This technique uses the index finger for tracking and only the thumb to select the button. Tapping the thumb activates the left button. Pressing down and swiping to the left activates the middle button, and swiping to the right activates the right button (Figure 5). A timeout value (150 ms) is used to determine how long to wait for a gesture before executing the left button event.

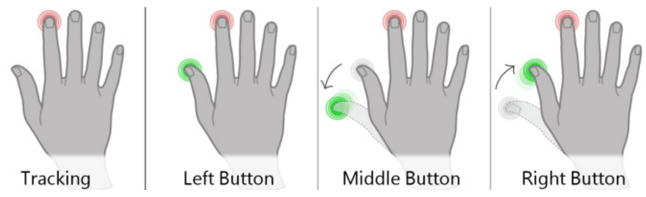

Figure 5. The Gesture Technique.

Side+Chording

The Side+Chording technique uses side information to determine the left and right button state. These fingers are chorded to activate the middle mouse button (Figure 6).

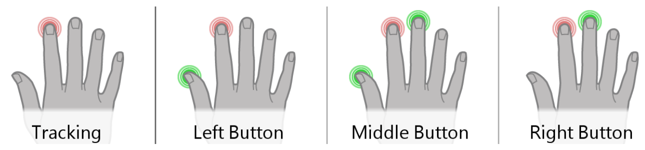

Figure 6. The Side+Chording Technique.

Side+Distance

With the Side+Distance technique the thumb is dedicated for left button activation. Pressing the middle finger, close to the index finger (<150 px) activates the middle mouse button, and pressing a finger further to the right (ring or pinky) (> 150 px) activates the right button (Figure 7).

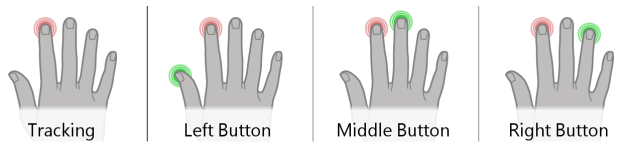

Figure 7. The Side+Distance Technique.

Chording+Distance

This technique is similar to Side+Chording, but we use distance to distinguish between the left (< 150 px) and right (> 150 px) buttons, and chording to execute the middle button (Figure 8).

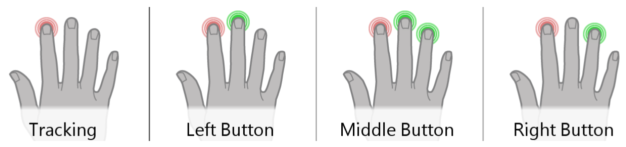

Figure 8. The Chording+Distance Technique.

PILOT STUDIES

Fully combining all four aspects of our design space, would results in almost 100 possible "techniques". We narrowed down this design space by using a converging-series design, running a series of four, two participant, pilot studies. Since the pilots only have two subjects, they should not be treated as rigorous experimental findings. While this method would not allow us to examine some of the interesting interactions between different aspects of the design space, we felt that the results of these studies, combined with examining pragmatic issues, would be useful to reduce the candidate techniques down to a smaller number of viable options. Our goal is not necessarily to find the best single technique, but to generate general recommendations and insights.

Apparatus

All studies were conducted on a custom 21" multi-touch monitor with 1600x1200 display resolution (Figure 9). The device is capable of detecting points of contact at a resolution less than 0.25mm. Pressure information was not available. The screen was covered with uncoated glass, and unfortunately this caused an undesirable resistance when dragging fingers along the surface. The display was positioned at an angle of 40° to minimize fatigue [12].

Task

Our task consisted of acquiring a target with the pointer, and then either clicking, double clicking, or dragging the target to a dock location with one of the three mouse buttons (left, middle, right). Before each trial the participant moved the cursor into a start location near the bottom of the screen (Figure 10a). After a 0.5s second delay, the start circle would disappear and the trial would begin by displaying a target square with the task instructions directly above it (Figure 10b-d).

The participants were asked to perform "as quickly and accurately as possible". For the single click tasks, the trial ended when the appropriate button-up event was recorded. Similarly, the double-click tasks ended on the second mouse-up event. When the target was moved within the dock, the color of the target changed from green to blue to indicate that the target was over the dock area. If the target was released outside of the docking area, an error was recorded, and the participant would have to re-acquire the target from its last location and try again to drag it in.

Independent Variables

The design of each individual pilot study varied, using different combinations of independent variables. In some studies, the width of the target was varied. For the drag tasks, the docking region was always 30 pixels wider than the target. The distance of both the length from the start position to the target, and from the target to the docking location, was also varied in some of the below studies.

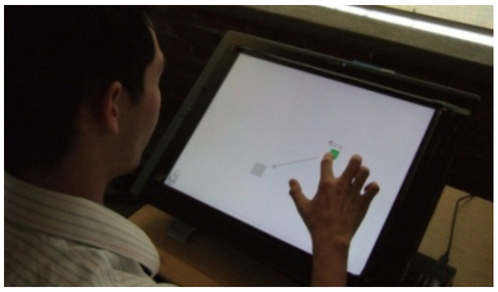

Figure 9. The experiment apparatus used for our studies.

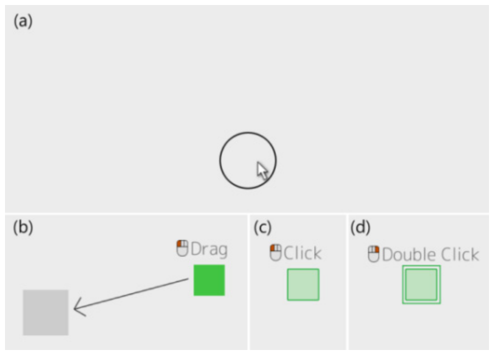

Figure 10. Task appearance and instructions. (a) The start position. (b) Drag task. (c) Single click task (d) Double click task. The mouse icon indicates which button to use (in these examples: left, left, right).

PILOT STUDY 1: BUTTON ACTIVATION

The purpose of the first pilot study was to compare *momentary* button activation (finger down for button-down event, finger up for button-up event) with *toggle* button activation (tap once for button-down event, tap again for button-up event).

Design

Two male subjects aged 24 and 25 participated in this study. Both were right handed and experienced computer users.

The independent variables were *activation type* (momentary and toggle), *task* (click, double click, and drag), and *button* (left, middle, and right). The target size and target distance were both held constant at values of 48 pixels, and 375 pixels respectively. Since our focus was only on button activation, we used the Side+Distance technique, one finger tracking, and a direct cursor mapping, which we felt would be the most intuitive choices for the user.

Each participant performed 8 blocks of trials for each *activation type*, with the order counterbalanced between participants. Blocks consisted of one trial for each of the 9 *task* and *button* combinations presented in random order.

Results
Figure 11 illustrates the completion times recorded from the study. For each task momentary activation was faster, and the overall mean completion times were 1.78s for Momentary and 2.65s for Toggle. This result was somewhat anticipated. The click was faster for momentary mode, as it requires a single "tap" of the finger, while in toggle mode it requires two finger taps (one for button-down, one for button-up). The problem is exaggerated further with the double-click task where momentary activation requires two taps, and toggle activation requires four taps (button-down, button-up, button-down, button-up), as apparent in Figure 11. For the dragging portion of the drag task, we thought toggle may be superior, since momentary activation requires two fingers on the surface, while toggle activation needs only one. With the surface friction of the display, it becomes noticeably less comfortable to drag with each additional finger placed on the display. In the drag task, Toggle activation was closer in performance to Momentary activation, but was still slower. As such, we recommend the use of Momentary activation, and will use this type of activation for our remaining studies.

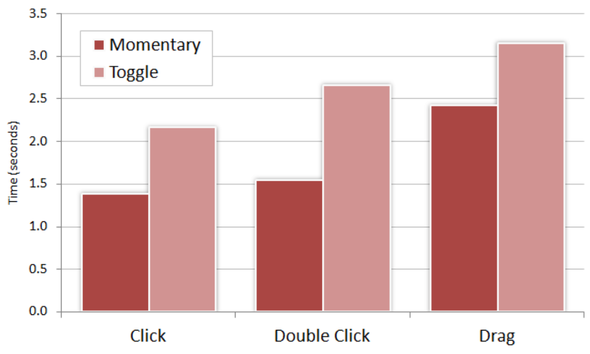

Figure 11. Pilot 1 completion times for the activation modes.

PILOT STUDY 2: TRACKING FINGERS
The second pilot study was to look at controlling the tracking-state position of the cursor with either one or two fingers. Two finger tracking places the cursor between the two fingers, offering the advantage of the input fingers not obscuring the pointer location. It has also been suggested that using two fingers provides more stability for pointing than does tracking with one finger [3, 8].

Design
Two male subjects aged 25 and 26 participated in this study. Both were right handed and experienced computer users.

The independent variables were *tracking fingers* (one finger and two fingers), *task* (click and drag), *target size* (16 pixels, 64 pixels), and *distance* (250 pixels, 500 pixels). We felt the number of tracking fingers used would have little effect on the actual button activation, so only the left button was used. For the two-finger technique, the thumb and middle fingers were used for tracking, and the index finger was used to activate the left button. The one finger technique was the exact same, except only the middle finger was used for tracking. Momentary activation was used, as per our Pilot Study 1 results. The offset cursor mapping was used so that we could look at the effect of one or two finger tracking on pointing precision without the compounding effect of cursor occlusion.

Each block consisted of one trial for each of the 8 *task*, *target size,* and *distance* combinations, presented in random order. The first participant did all of the one-finger trials first, while the second began with the two-finger trials.

Results
Overall, one-finger tracking performed better than two-finger tracking, with mean completion times of 1.75s and 2.04s respectively. We found that when using one finger tracking subjects were more likely to "leap" to the target, that is, lift their tracking finger off the screen and place it down at the target, than they were with two finger tracking. Two finger tracking performed comparatively better in the small target conditions (Figure 12), but was still slower.

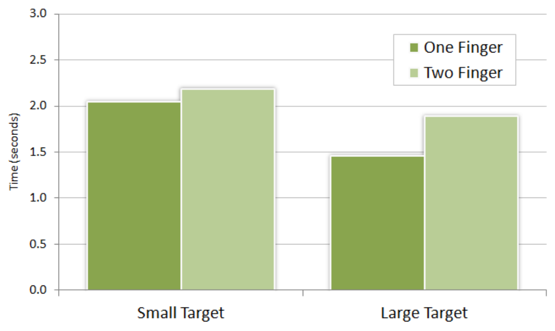

Figure 12. Task completion times for Pilot Study 2: one and two finger tracking modes.

For both subjects, one-finger tracking was preferred over two-finger for both large and small target sizes. Since one finger tracking also addresses the design consideration of using fewer touch points and performed better in this pilot, it is our recommended tracking mode, and will be used for the remaining studies.

PILOT STUDY 3: CURSOR MAPPING
Pilot 3 was designed to determine which of our four cursor mappings (direct, scaled, offset, relative) is most preferable.

Design
Two male subjects aged 24 and 25 participated in this study. Both were right handed and experienced computer users.

The independent variables were *cursor mapping* (direct, offset, relative, and scaled), *task* (click and drag), *target size* (16 pixels, 64 pixels), and *distance* (250 pixels, 500 pixels). As in Pilot Study 2, this study used only the left button. One finger tracking was performed using the index finger, and the left button was activated with the thumb, using *momentary* button activation.

Each participant performed 4 blocks for each of the 4 techniques, with each block consisting of the 8 combinations of *task, target size*, and *distance*. The first participant did the *cursor mapping* techniques in the order of direct, scaled, offset, relative, and the second participant did the reverse order.

Results

Overall completion times were 1.77s, 1.62s, 1.81s, and 2.15s, for direct, offset, relative and scaled respectively. For large target tasks, direct mapping, with the cursor directly under the tracking finger, was 0.13s faster than the next best technique, offset (Figure 13). It was also subjectively preferred as the subjects felt most comfortable pointing directly at the target. They could do so because the large target was larger than the area obscured by their finger.

However, when acquiring small targets, the direct mapping does not fare as well, and was 0.43s slower than the offset technique. The cursor and small target are both occluded by the tracking finger, making it difficult to see if the cursor is over the target. Offsetting the cursor slightly above the tracking finger provides many of the benefits of the direct mapping, while addressing the occlusion problem.

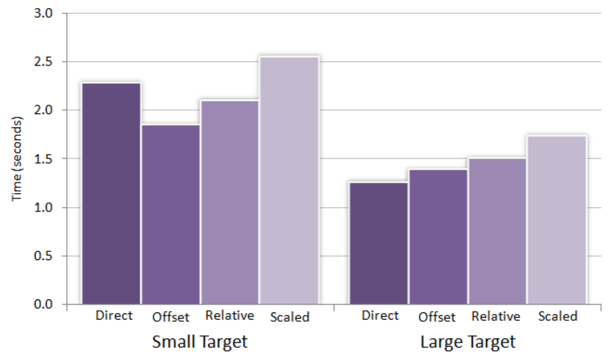

Figure 13. Task completion times for Pilot Study 3: Cursor mapping modes.

Despite its advantage for large targets, the frustration caused by activating small targets causes us to recommend against a direct mapping. The offset mapping performed best overall, but does not possess *edge support*, as the bottom edge will not be accessible. While relative and scaled input did not perform as well, they provide edge support, and further, were reported to reduce fatigue, since the input footprint was lowered. As relative mode performed better than scaled, we recommend the use of a relative mapping when the designer feels it is important to reduce input footprints and provide edge support. Otherwise, our recommendation is to use the offset mapping, and to seek potential solutions to the edge support problem. For example, it could be addressed by altering the mapping near the bottom of the screen, or extending the tracking area beyond the display surface. Since our current investigations are not focused on the edge support problem, we will use the offset mapping for our remaining studies.

PILOT STUDY 4: TECHNIQUES

The purpose of the final pilot study is to investigate the 7 described finger-to-button mapping techniques.

Design

Two male subjects aged 25 and 26 participated in the study. Both were right handed and experienced computer users.

The independent variables were *technique* (Chording, Side, Distance, Gesture, Side+Distance, Side+Chording, and Chording+Distance), *task* (click, drag), and *button* (left, middle, right). The target size and distance were both fixed to values of 48 pixels and 375 pixels respectively.

Based on the recommendations from the three previous pilots, the button activation mode was momentary, the tracking mode was one finger (except for the Side Technique), and the cursor mapping was offset. Each block consisted of one of each of the 6 combinations of *task* and *distance*, and each participant performed 6 blocks of trials for each *technique*. The first participant performed the *techniques* in the above order, and the order was reversed for the second participant.

Results

The best performing techniques were Side+Distance (1.73s), and Distance (1.82s) (Figure 14). Both techniques support our intuitive mapping design property. Side+Distance has the advantage of using the thumb for left button input, which the subjects preferred.

Our own experience, in addition to the feedback received from the subjects, suggests that the techniques that require chording (Chording, Side+Chording, and Chording+ Distance) are awkward to use, as they are time dependent, and introduce additional fingers during dragging operations. The Side Technique requires two-finger tracking with the index and ring fingers and as such, is ergonomically uncomfortable to use. Users liked the gesture technique, but it took longer, and is timing dependent.

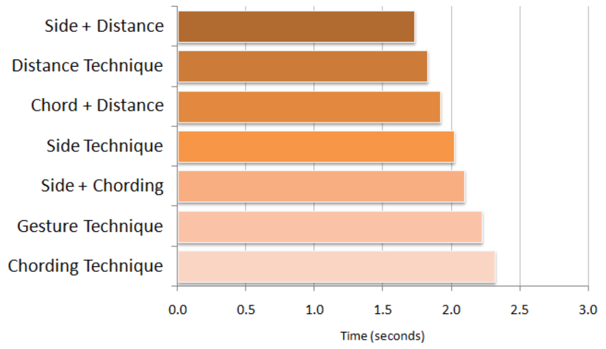

Figure 14. Task completion times for finger-to-button mapping techniques.

The results of the experiment show that all of these techniques are viable. However, considering together the quantitative data, subjective feedback, and qualitative observations, our recommendation is to use the

Side+Distance mapping. It is intuitive, minimizes touch points, is timing independent, and comfortable to use.

PILOT STUDIES SUMMARY: THE SDMouse

Our pilot studies have provided valuable insight into the design of multi-finger mouse emulation techniques. While it is difficult to suggest a single design, we are motivated to evaluate one possible technique against existing solutions. As such, we propose the *SDMouse* where:

- Button activation is *momentary*
- Tracking fingers is *one*
- Cursor mapping is *offset*
- Finger to button mapping is *Side+Distance* (hence SDMouse)

As discussed earlier, any emulation technique (including the SDMouse) will not satisfy all desirable design characteristics, and the above dimensions could all be modified if desired. For example, if edge support is desired, the cursor mapping could be relative.

EXPERIMENT

The goal of this experiment is to compare the performance of the SDMouse to previously suggested techniques for mouse emulation. Furthermore, we believe it is important to understand how close these techniques are to being optimal; that is, matching the performance of an actual physical mouse. Below we briefly describe the techniques which *SDMouse* will be compared against.

Techniques

Comparison Technique 1: FingerWorks

FingerWorks [9] uses two fingers in the tracking mode and was actually designed for indirect multi-touch devices, so it uses a relative cursor mapping (Figure 15). Since FingerWorks was not meant to be used in a direct touch configuration, our results do not reflect an overall assessment of the technique. Three fingers are required for the emulation of all the buttons. The buttons are distinguished by the arrangement of the fingers, and activated *momentarily*. For the left mouse button, the index, middle, and ring fingers are tapped. For the middle mouse button, the thumb, index, and middle fingers are used. Lastly, for the right button, the thumb, ring, and pinky fingers are used with the hand slightly spread.

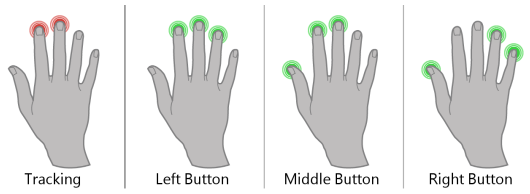

Figure 15. The FingerWorks technique.

Comparison Technique 2: Fluid DTMouse

Fluid DTMouse supports tracking and the left and right button, but does not support the middle mouse button. It supports both one-finger and two-finger tracking, with a direct cursor mapping. A toggle button activation protocol is used. To create a left button-down or button-up event, the user tracks with two fingers, and then taps the index finger. For the right button, the user tracks with a single finger, and taps with a second (Figure 16).

The original Fluid DTMouse supports a mode that treats a single touch point as a left click, but supporting this would disable a user's ability to position the cursor before clicking the right button. This would make it very hard to be precise with the right button, and so we do not enable this mode.

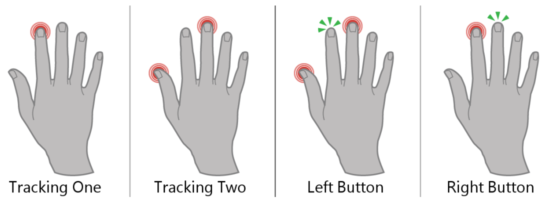

Figure 16. The Fluid DTMouse Technique.

Comparison Technique 3: Actual Mouse

To provide an empirical comparison to an actual mouse, we used a 3-button 400 dpi optical mouse with ballistic pointing enabled and set to a comfortable level such that mouse clutching was never needed. Previous work [26] has shown that direct pointing can surpass the use of an indirect mouse, however, this result only applied to single finger usage. We expect the overhead cost of introducing tracking and multiple button support will cause the emulation techniques, both previous and our own, to be slower than a physical mouse. Our goal here is not to match the performance of the mouse, it is only to empirically determine the magnitude of the introduced cost.

Task and Apparatus

The task and apparatus which we used was identical to the pilot studies, with the exception of a physical mouse, described above.

Participants

Twelve volunteers (10 male, 2 female), aged 21-36 participated in this experiment. Participants were all right handed, experienced computer users without any multi-touch experience.

Design

A repeated measures within-participant design was used. The independent variables were *technique* (SDMouse, DTMouse, FingerWorks, Mouse), *task* (click, drag), *button* (left, right), *distance* (250 pixels, 500 pixels), and *size* (16 pixels, 64 pixels). We did not include the middle mouse button since DTMouse does not support it. A fully crossed design resulted in 64 conditions. Each participant performed the experiment in one session lasting approximately 60 minutes. The session was broken up by the techniques, with 5 blocks of trials completed for each technique. In each block participants would complete each of the 8 combinations of *task, button, distance,* and *size*.

Before using each technique, participants completed a short warm-up session, to familiarize themselves with the technique. The ordering of techniques was counterbalanced using a Balanced Latin Square design.

Results

Completion Times

Completion time was calculated in the same manner as the pilot studies. The analysis included trials in which errors occurred. However, we used the median completion time to correct for typical skewing of the time data and to remove outliers. The median was calculated across the 5 blocks.

Repeated measure analysis of variance showed main effects for *technique* ($F_{3,33} = 74.7$, $p < .0001$), *task* ($F_{1,11} = 284$, $p < .0001$), *button* ($F_{1,11} = 23.3$, $p < .005$), *distance* ($F_{1,11} = 17.2$, $p < .005$), and *size* ($F_{1,11} = 80.6$, $p < .0001$). Overall completion times were 1.58 for the mouse, 2.41 for SDMouse, 3.16 for FingerWorks, and 4.25 for DTMouse. Pairwise comparison using Tukey adjustment showed that all of these values were significantly different ($p < .001$).

The interaction between *technique* and *task* was significant, ($F_{3,33} = 5.0$, $p < .01$). The effect is illustrated in Figure 17. SDMouse comes closer to approaching the performance to the mouse in the clicking task, with a difference of 0.54s. In the dragging task the difference is 1.1s. This is consistent with previous comparisons of direct touch with the mouse [12]. The interaction between *technique* and *button* was also significant ($F_{3,33} = 10.36$, $p < .0001$). Figure 18 shows that the mouse and SDMouse have equal times for left and right click, while for FingerWorks and DTMouse right button times were much higher. The *technique* x *size* and *technique* x *distance* interactions did not reach significance.

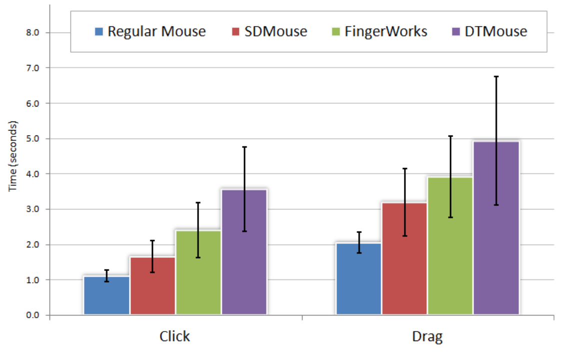

Figure 17. Completion times by task.

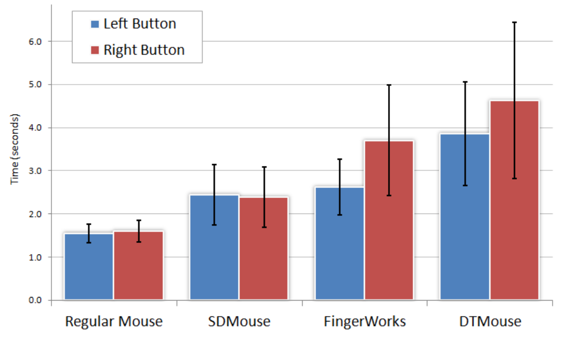

Figure 18. Completion times by button.

Error Rates

Repeated measure analysis of variance showed that *technique* also had a significant effect on error rates ($F_{3,33} = 25.5$, $p < .0001$). The average error rates were 8.5% for the mouse, 21.8% for SDMouse, 27.8% for FingerWorks, and 42.8% for DTMouse. A main cause of errors came from the dragging task, due to the friction created with the surface ($F_{1,11} = 27.9$, $p < .0001$). The display surface material was slightly sticky, which caused the fingers to sometimes skip. This would be registered as an error. However, it should be noted that users were still able to complete the task, and our analysis of movement time incorporated these errors. Considering the pointing task only, error rates were 6.2% for the mouse, 14.8% for SDMouse, 23.5% for FingerWorks, and 35.9% for DTMouse. We believe the high error rates for FingerWorks and DTMouse were caused by the difficulty learning the finger mappings.

DISCUSSION & FUTURE WORK

Designing a multi-touch mouse emulation technique introduces numerous issues to consider. While performance and error rates are important measurements of success, we have found that additional factors play into the appeal of a technique. For example, how easy is it to remember? How well does it map to a physical mouse? Is it comfortable?

Offering graphical widgets such as Microsoft's Vista TabletPC mouse or a TrackingMenu [10] version are two examples of approaching the problem from a different angle. An interesting solution to explore would be to present a graphical mouse next to the index finger when it is in contact with the sensing surface and is stationary. The user would then use their thumb and mouse graphic to trigger a mouse button or perform a mouse action.

The mouse is often used in conjunction with modifier keys (Ctrl, Shift, etc). We believe this can be achieved utilizing the other hand and is another area of future research.

In addition, providing mouse scrollwheel support is another area we wish to investigate. Two potential solutions are to use repeated middle finger swiping or thumb rotation (Figure 19). Repeatedly swiping the middle finger up or down while the tracking finger is stationary would signal scrollwheel up/down events. Alternatively, with thumb rotation, a clockwise circular gesture with the thumb translates to scrollwheel up events while counterclockwise gestures map to scrollwheel down events. Our initial implementations of the swiping technique seemed to work but required a rate-based mapping.

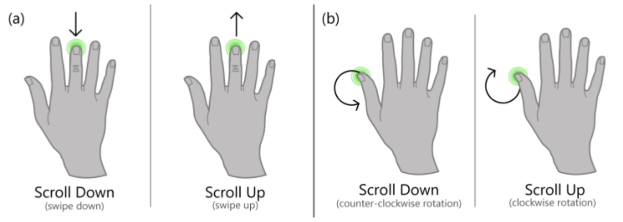

Figure 19. Mouse Wheel Techniques: (a) swiping (b) rotation.

We did not study very precise (e.g., single pixel) positioning or selection tasks. While two-finger designs aim to offer more precise cursor positioning by positioning the cursor between the two finger points, we believe the single finger tracking design with cursor offset can be similarly accurate as the multi-touch sensing technology improves.

Our approach to exploring the design space utilized pilot studies to examine individual design issues. We note that this method does not allow us to examine some of the interesting interactions between different aspects of the design space. Also, the pilot studies were not meant to lead us to absolute truths or performance measurements; instead they were meant to lead us to potential design solutions.

CONCLUSIONS

Multi-touch systems offer the potential of very rich input. However, it is unreasonable to expect that every application will be rewritten to support a multi-touch interaction mode. Users will need to be able to interact with the traditional user interfaces using multi-touch input, much like TabletPC users are forced to interact with traditional GUI's with a stylus. More importantly, full mouse emulation should be supported in multi-touch environments to effectively interface with traditional applications which have minimal or no multi-touch capabilities. Our research contribution is the systematic exploration of the solution space of one handed, multi-touch, full mouse emulation techniques. By constructing a design space and conducting controlled studies we have been able to generate a set of viable designs, such as the SDMouse.

REFERENCES

1. Albinsson, P. and Zhai, S. (2003). High precision touch screen interaction. *ACM CHI.* 105-112.
2. Balakrishnan, R., Baudel, T., Kurtenbach, G., and Fitzmaurice, G. (1997). The Rockin'Mouse: integral 3D manipulation on a plane. *ACM CHI*. 311-318.
3. Benko, H., Wilson, A., Baudisch, P. (2006). Precise Selection Techniques for Multi-Touch Screens. *ACM CHI*. 1263-1272.
4. Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. (2008). Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. *AVI*. 154-161.
5. Buxton, W., Hill, R., and Rowley, P. (1985). Issues and techniques in touch-sensitive tablet input. *ACM SIGGRAPH*, 215-224.
6. Dietz, P. and Leigh, D. (2001). DiamondTouch: a multi-user touch technology. *ACM UIST*. 219-226.
7. Echtler, F., Huber, M., and Klinker, G. (2008). Shadow tracking on multi-touch tables. *AVI*. 388-391.
8. Esenther, A. and Ryall, K. (2006). Fluid DTMouse: better mouse support for touch-based interactions. *AVI.* 112-115.
9. Fingerworks, Inc. (2008). User's Guide. http://www.fingerworks.com/gesture_guide_mouse.html
10. Fitzmaurice, G., Khan, A., Pieké, R., Buxton, B., and Kurtenbach, G. (2003). Tracking menus. *ACM UIST.* 71-79.
11. Forlines, C., Vogel, D., and Balakrishnan, R. (2006). HybridPointing: fluid switching between absolute and relative pointing with a direct input device. *ACM UIST*. 211-220.
12. Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. *ACM CHI.* 647-656.
13. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., and Balakrishnan, R. (2006). Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. *ACM CHI*. 861-870.
14. Grossman, T., Wigdor, D., and Balakrishnan, R. (2004). Multi-Finger Gestural Interaction with 3D Volumetric Displays. *ACM UIST.* 61-70.
15. Hager-Ross, C., Schieber, M. (2000). Quantifying the Independence of Human Finger Movements: Comparisions of Digits, Hands, and Movement Frequencies. *J. of Neuroscience*, 20(22), 8542-8550.
16. Han, J. Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. *ACM UIST* 115-118.
17. Letessier, J. and Bérard, F. (2004). Visual tracking of bare fingers for interactive surfaces. *ACM UIST.* 119-122.
18. Malik, S. (2007). An Exploration of Multi-Finger Interaction on Multi-Touch Surfaces. Univ. of Toronto.
19. Malik, S., Ranjan, A., and Balakrishnan, R. (2006). Interacting with large displays from a distance with vision-tracked multi-finger gestural input. *ACM UIST*. 43-52.
20. Moscovich, T. (2007). Principles and Applications of Multi-touch Interaction. Brown University.
21. Moscovich, T. and Hughes, J. F. (2006). Multi-finger cursor techniques. *Graphics Interface*. 1-7.
22. Moscovich, T. and Hughes, J. F. (2008). Indirect mappings of multi-touch input using one and two hands. *ACM CHI*. 1275-1284.
23. Olwal, A., Feiner, S., and Heyman, S. (2008). Rubbing and tapping for precise and rapid selection on touch-screen displays. *ACM CHI*. 295-304.
24. Potter, R. L., Weldon, L. J., and Shneiderman, B. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. *ACM CHI.* 27-32.
25. Rekimoto, J. (2002). SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. *ACM CHI*. 113-120.
26. Sears, A. and Shneiderman, B. (1991) High Precision Touch-Screens: Design Strategies and Comparison with a Mouse, *Int. Journ. of Man-Machine Studies*, 43(4), 593-613.
27. Vogel, D. and Baudisch, P. (2007). Shift: a technique for operating pen-based interfaces using touch. *ACM CHI.* 657-666.
28. von Hardenberg, C. and Bérard, F. (2001). Bare-hand human-computer interaction. *ACM PUI.* 1-8.
29. Wu, M. and Balakrishnan, R. (2003). Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. *ACM UIST*. 193-202.
30. Wu, M., Shen. C., Ryall, K., Forlines, C., Balakrishnan, R. (2006). Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces. *IEEE TableTop*. 183-190.
31. Zhang, Z., Wu, Y., Shan, Y., and Shafer, S. (2001). Visual panel: virtual mouse, keyboard and 3D controller with an ordinary piece of paper. *ACM PUI.* 1-8.