Direct Manipulation Video Navigation on Touch Screens

Cuong Nguyen[1] **Yuzhen Niu**[2,1] **Feng Liu**[1]
[1]Portland State University [2]Fuzhou University
Portland, OR 97207-0751 USA Fuzhou, Fujian 350108, China
{cuong3,yuzhen,fliu}@cs.pdx.edu

(a) DMVN using a mouse (b) DMVN using a finger

Figure 1. Direct Manipulation Video Navigation (DMVN) using a mouse vs. a finger. With DMVN, a user can directly drag the car along its trajectory to the target location using a mouse as shown in (a). While a touch screen can fit DMVN naturally and enhance the directness, the fat finger problem with touch screens can make dragging the car difficult.

ABSTRACT

Direct Manipulation Video Navigation (DMVN) systems allow a user to directly drag an object of interest along its motion trajectory and have been shown effective for space-centric video browsing tasks. This paper designs touch-based interface techniques to support DMVN on touchscreen devices. While touch screens can suit DMVN systems naturally and enhance the directness during video navigation, the fat finger problems, such as precise selection and occlusion handling, must be properly addressed. In this paper, we discuss the effect of the fat finger problems on DMVN and develop three *touch-based object dragging techniques* for DMVN on touch screens, namely *Offset Drag*, *Window Drag*, and *Drag Anywhere*. We conduct user studies to evaluate our techniques as well as two baseline solutions on a smartphone and a desktop touch screen. Our studies show that two of our techniques can support DMVN on touch screen devices well and perform better than the baseline solutions.

Author Keywords

Video navigation; direct manipulation; mobile devices; touch screens.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces

MobileHCI '14, September 23 - 26 2014, Toronto, ON, Canada

http://dx.doi.org/10.1145/2628363.2628365

INTRODUCTION

Capturing, sharing, and watching videos have now become common daily activities. To better support video navigation and browsing tasks, a variety of novel video browsing technologies have been developed [19, 21, 4, 15]. Among them, Direct Manipulation Video Navigation (DMVN) systems allow a user to navigate a video by directly dragging an object of interest along its motion trajectory, as shown in Figure 1 (a), and have been shown well suited for space-centric video navigation tasks [5, 7, 11, 12, 13, 14, 17].

Meanwhile, many of video-capable devices are now equipped with a touch screen, such as smart phones and tablets. DMVN systems can suit these touch-screen devices naturally. As is well known, touch-based interfaces allow a user to interact with an object directly on the screen without any extra input devices and thus reduce the gap between the user input and the system output [22]. For a DMVN system, allowing a user to directly interact with the object of interest can remove the need to use an extra input device like a mouse and further enhance the directness of the video navigation experience.

However, proper touch-based interfaces must be designed for DMVN. The fat finger problem [25] can severely compromise the performance of DMVN on touch screens, as shown in Figure 1 (b). When a user's finger occludes the content behind the finger, it becomes difficult for a user to precisely select an object of interest. A particularly unique challenge with a DMVN system is that the occlusion problem will prevent the user from receiving the feedback from the DMVN system such as the object location and contextual trajectory information, making it difficult to *drag the object* along its trajectory and to the desired location. To our best knowledge,

[1] `http://graphics.cs.pdx.edu/project/TouchDMVN/`

while various DMVN systems have been developed and some were experimented on touchscreen devices or provided an implementation for touchscreen devices, no touch-based interfaces, which are intended for DMVN systems and handle the fat finger problem, have been reported.

In this paper, we report our designs of touch-based interfaces for DMVN systems. We build upon previous research on precise selection techniques for touchscreen, such as Shift [29] and Offset Cursor [20, 23], and extend them to handle object manipulation in a DMVN system on touchscreen devices. Specifically, we design three *object dragging techniques* for touch-based DMVN interaction, including *Offset Drag*, *Window Drag*, and *Drag Anywhere*, which allow a user to drag an object of interest along its motion trajectory without suffering from the fat finger problem. We evaluate these techniques together with two baseline solutions in user studies on both a large desktop touch screen and a small touch screen on an Android smartphone. Our studies show that two of the three touch-based interfaces that we developed enable significantly better direct manipulation-based video navigation than the baseline solutions.

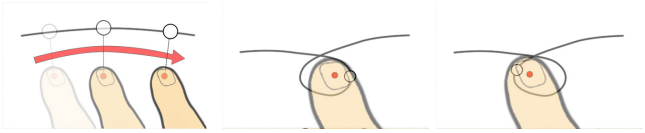
DMVN on Touch Screen

In our work, we implemented a standard DMVN system similar to what was discussed in [11]. To support direct object manipulation, we first used computer vision techniques, including optical flow [28] and feature tracking [24], to estimate the motion trajectories of the objects in an input video. After a user selects an object to manipulate, the object's trajectory is rendered on the screen to help a user drag the object spatially. When a user drags her finger on the screen, the system calculates and navigates to the video frame where the object is spatio-temporally closest to the user's pointer location. Thus the object of interest always follows the user's pointer, creating a feeling of direct manipulation during video navigation. More detail about the DMVN system can be found in [11].

DMVN consists of two steps: object selection and object dragging. On touchscreen devices, both steps can be difficult due to the fat finger problem, which happens when the finger is bigger than the object [25]. This leads to the precision and occlusion problem, compromising the user experience in manipulating a video on a touch screen. Existing methods for precise object selection on touch screens can be used to solve the problem of object selection [20, 23, 1, 3, 18, 29]. They, however, cannot handle object dragging. Below we discuss the effect of the precision and occlusion problem on DMVN, focusing on object dragging.

Precision

A large corpus of touch-based interface techniques have been developed to facilitate the target acquisition task on touch screens [1, 3, 18, 29]. Different from regular target selection tasks, interacting with a DMVN system requires a user to interact with the motion trajectory. In a DMVN system, the original finger contact point on the screen is mapped to the closest point on the trajectory, progressing the video to the corresponding frame in time. This mapping technique is similar to the Bubble Cursor method [8] where the nearest target around the cursor area is acquired.



(a) dragging (b) jumping

Figure 2. The effect of the fat finger problem on DMVN. DMVN maps the finger touch point to its spatio-temporally nearest point. This works well for a simple trajectory as shown in (a). For a complex trajectory, the inaccuracy in dragging objects caused by the fat finger problem can make DMVN map the finger touch points to different segments of the trajectory and results in jumping.

Within this context, even if the finger contact point can be ambiguous, a user can easily drag the object along its motion trajectory without worrying about inaccurate selection. Figure 2 (a) illustrates this idea, even though it is difficult to select a specific frame with his finger, the object can still be dragged correctly until it reaches the desired frame. If the trajectory is simple enough, a user can also move her finger away from the object during dragging and still maintain a continuous video navigation while being able to observe the occluded target. This behavior is similar to MobileZoomSlider [10] which allows a user to move the cursor away to adjust the dragging speed. When the trajectory is complicated, such as when it forms a loop or an intersection, the inexact selection point of the finger may make the system accidentally map the touch point to a wrong trajectory segment, as shown in Figure 2 (b). While this problem can be alleviated with the temporal continuity constraint discussed in [11, 5], the same problem will still occur for some complicated trajectories.

Occlusion

Compared to the precision selection issue, occlusion from the fat finger problem downgrades the performance of DMVN on touchscreen devices even more. For space-centric video browsing tasks, a user needs to navigate to a frame with some specific visual properties. This requires a user to pay attention to not only the object that she is dragging, but also the motion trajectory and the video content as well. Due to occlusion, it is difficult for a user to make micro-adjustment for frame-accurate video navigation. When the local trajectory segment is occluded by the finger, in combination of its complex shape, and the ambiguity in selection points as mentioned above, the user is more likely to make an unexpected jump between nearby trajectory segments. Finally, the occluded video content prevents a user from observing and appreciating the video while navigating, compromising the user experience of DMVN.

RELATED WORK

Our work is directly related to research on direct manipulation video navigation and research to address the fat finger problem on touch screens.

Direct Manipulation Video Navigation

Direct manipulation video navigation systems allows for effective space-centric video browsing [5, 7, 11, 12, 13, 14, 17]. These systems estimate motion information using computer vision algorithms, such as optical flow [28], visualize the motion trajectory in the video display window, and allow

a user to drag an object of interest along its motion trajectory. Among existing DMVN systems, PocketDragon from Karrer *et al.* was developed to run on Apple iPhone with multi-touch capability [12]. PocketDragon allows a user to navigate an object of interest by directly dragging the object using a finger directly on the screen. To address the occlusion problem when selecting the object of interest to drag, PocketDragon allows a user to pick the most dominant motion trajectory around the finger contact point. While this system provides a good attempt at developing a DMVN system for touchscreen mobile devices, it did not address the fat finger problem with touch screen. Recently, Karrer *et al.* conducted a user study to evaluate their DMVN system in handling spatial-temporal ambiguity problem using a touch screen [13]. Their system does not address the fat finger problem either. Instead, a stylus was provided for a user to manipulate an object of interest.

Precise Selection on Touch Screens

Previous work have been done to improve the selection precision on touch screen devices. Zooming techniques magnify the selection area when an object is very small [1, 3, 18] and can effectively improve the selection precision. These techniques, however, may not suit DMVN well. If we zoom back after selection, the fat finger problem affects dragging. If we do not zoom back and drag while scrolling/panning the screen, video content will move outside the screen. If we use fisheye view warping, the trajectory will be distorted and mislead dragging.

The control-display ratio can also be manipulated to improve the selection precision [1, 3]. These techniques allow a user to modify the control-display ratio - a mapping between the input movement and the system pointer movement. When the control-display ratio is smaller than 1, the pointer will move slower than the input finger. This has two advantages. First, because the pointer moves slower than the finger, the user finger does not occlude the pointer anymore. Second, the slow movement of the pointer allows a user to select a small object more accurately. Although these techniques have been shown effective for small target selection [1, 3], they were not designed for video object dragging in a DMVN system. These techniques either require additional widget manipulation, or need a user to focus on controlling the system pointer to acquire a fix target, which can be inconvenient for object dragging. Our *Drag Anywhere* technique is inspired by the advantages of the CD-ratio manipulation techniques. We map the finger dragging direction and acceleration to the arc-length distance of the motion trajectory. In this way, a user can control the object from anywhere on the screen by dragging her finger according to the motion trajectory.

Occlusion Handling on Touchscreen Devices

Offset cursor by [20, 23] addresses the occlusion problem by allowing a user to select an object without having to position the finger on top of it. When a user touches the interaction screen, this technique displays a cursor on top of the user finger. The orientation and distance between the cursor and the finger contact point are fixed. By maintaining contact with the touch screen, a user can navigate the cursor to the desired location and select the target using the take-off technique: selection is made after lifting the finger from the screen. Offset cursor cannot be directly applied to the task of dragging an object along its motion trajectory as the orientation and the offset distance cannot be dynamically adjusted. This is problematic for DMVN as a fixed orientation can make the finger occlude the motion trajectory when the cursor orientation is similar to the local motion trajectory. We extended the Offset Cursor technique and develop an *Offset Drag* technique for touch-based DMVN. *Offset Drag* allows a user to adjust the orientation and length of the offset cursor dynamically.

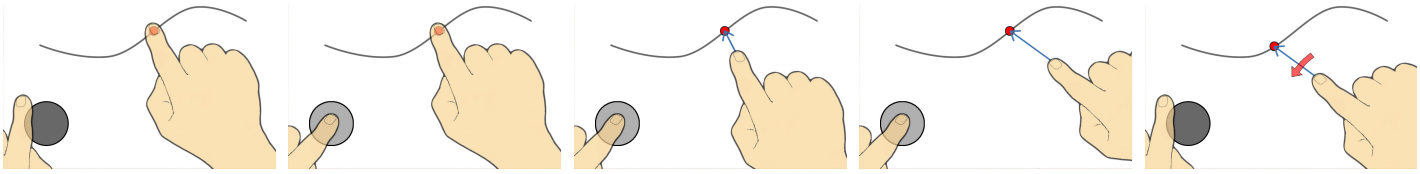
Shift addresses the occlusion problem by displaying the content occluded by the finger in a callout window in a non-occluded area [29]. A dash-line connecting the user finger to the center of the window indicates the current pointer location. A user can observe the occluded content in this window and adjust the selection. Shift also uses the take-off technique to trigger selection. It also determines whether it is necessary to escalate the window, allowing a user to use the direct touch technique for a large object. While Shift works well for object selection, it sometimes cannot support object dragging well. During dragging, a user needs to constantly receive feedback from the video content and the motion trajectory to decide where to go next. When the callout window size is fixed, it may not cover enough feedback information for a user to maintain a fluent dragging experience; otherwise a user might have to shift her/his attention from the callout window and the real motion trajectory, causing unnecessary re-orientation effort. Simply increasing the size of the window does not help but distracts the user even more. We design a *Window Drag* technique that changes the callout window size adaptively according to the current trajectory segment.

Specialized Hardware

Hardware-specific techniques have also been designed to address the finger occlusion problem. Back-of-device interaction techniques allow a user to interact with the content on the back of the device, avoiding occlusion completely [27, 31, 32, 2, 9]. A variety of devices have been explored. Wigdor *et al.* designed an interactive touch table that can receive touch inputs from both top and bottom surface [31]. LucidTouch and NanoTouch [30, 2] allow back-of-device interaction for tablet-size and very small-size devices. Back-of-device interaction has also been seen in commercial product such as *Sony PSP Vita* which features a rear touch pad. The recent developed NailDisplay [26] can show the visual content underneath the finger on a small display mounted on the finger. Although these techniques can handle the fat finger problem, they all require specific hardware support. This work addresses the fat finger problem in object dragging during video navigation on general-purpose touch screens.

DESIGN

DMVN consists of two steps: object selection and object dragging. We can use any of the above mentioned state-of-the-art precise selection techniques for the object selection step in DMVN. Thus, we assume that an object of interest has already been selected for manipulation. In this paper, we focus on object dragging and design *Offset Drag*, *Window Drag*,



(a) (b) (c) (d) (e)

Figure 3. Offset Drag for DMVN. By default, Offset Drag works like direct dragging (a). To set up a cursor or change the cursor orientation/length, a user keeps the primary finger touching the object, and uses the secondary finger to press a button to trigger the cursor (b). By holding the secondary finger on the button on the screen, the user can move the primary finger away from the object to define the offset distance (c) and orientation (d). Once the cursor is defined, the user lifts the second finger and uses the primary finger to drag the object using the cursor (e).

and *Drag Anywhere* for non-occluding direct object manipulation/dragging for DMVN.

Design 1: Offset Drag

We first implemented the basic Offset Cursor technique [20, 23] to support our touch-based DMVN system. When a user touches the screen, we provide a cursor at a fixed distance above the finger's contact point. When triggered, the system takes the offset cursor location as input to find the next frame in the video. In this way, we allow dragging while a user maintains contact with the screen and moves the offset cursor along the motion trajectory. Take-off selection technique was not implemented since we did not need to perform object selection during dragging.

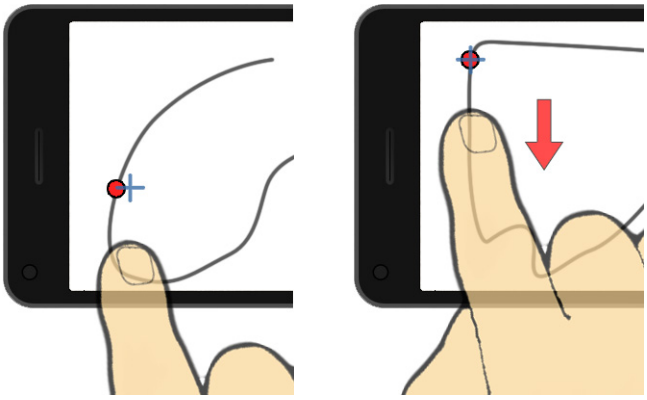
Offset Cursor is good at handling the occlusion problem because the user's finger does not need to be positioned on top of the object. However, this technique still has several limitations when it is applied to touch-based DMVN. First, as discussed in [29], a problem with the Offset Cursor technique is that the user is unable to aim for the actual target when they first interact with the device. When the offset cursor is triggered, it will point to the location above the user's finger, instead of the actual object location. This is problematic for DMVN when the motion trajectory is cluttered. If the cursor is triggered on a different trajectory segment nearby, the object may jump to that part, causing an unexpected scene jump. Second, for DMVN, precise dragging is not always required. Then the effort to trigger the offset cursor can delay navigation. Third, Offset Cursor does not allow a user to access some parts of the screen like the edges as discussed in in [29]. This is because the orientation and length of the offset cursor is fixed. As a result, edge and corner areas become difficult to reach. Moreover, when the cursor orientation is similar to the neighboring motion trajectory segment, only a very short segment ahead of the current position is shown and the rest is occluded by the finger, as shown in Figure 4 (b). This makes it difficult for a user to drag quickly.

To address these issues, we designed an extension of Offset Cursor that satisfies the following requirements,

1. A user should be able to offset the finger selection point to avoid occluding the object of interest.

2. A user should be able to adjust the orientation and length of the offset pointer dynamically.

3. A user needs to trigger the offset cursor only when needed.



(a) Edge problem (b) Occlusion

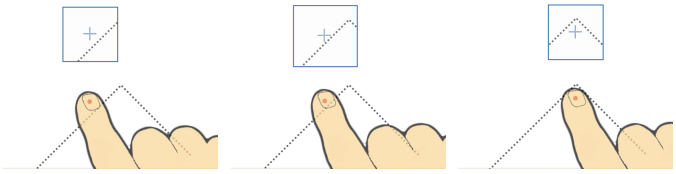
Figure 4. Problem with Offset Cursor in a DMVN system.

We followed the design guidelines of Benko *et al.* [3] to design a bi-manual interaction technique for dragging object in DMVN using an offset cursor. We call it *Offset Drag*. Offset Drag allows a user to trigger the offset cursor only when needed. A user can also interactively adjust the offset cursor's orientation and length during video navigation to allow access to all screen regions and avoid occlusion with the neighboring trajectory segment.

Compared with Offset Cursor, Offset Drag has an extra trigger button, located at the bottom left corner of the screen, for a user to trigger and adjust the cursor. With Offset Drag, a user's primary finger is used as the main finger for dragging. By default, Offset Drag works like direct dragging. While the primary finger is on contact with the object, placing the secondary finger onto the trigger button will trigger a cursor. By holding the secondary finger on the screen, moving the primary finger away from the object will not move the object. Instead, by moving it away from the object, a user will be able to define the offset distance and orientation as illustrated in Figure 3. The cursor is always located at the current object location, and the system renders an arrow to depict the current offset distance. Offset Cursor rendered the cursor as a crosshair. We used an arrow instead to better depict the orientation of the cursor. To continue dragging, a user lifts her secondary finger from the screen. The dragging movement now will be offset with the newly defined offset cursor. Then Offset Drag works in the same way as Offset Cursor. Similarly, a user can dynamically adjust the cursor during navigation.

Design 2: Window Drag

Our next technique, *Window Drag*, is inspired by the Shift technique [29], which places a callout window near the finger



(a) Default size (b) New size (c) Default size

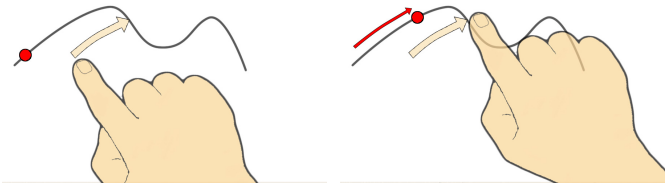
Figure 5. Adaptive window size for *Window Drag*. Two windows with the same size can cover different trajectory lengths. The trajectory segment in (a) is shorter than (c). We adaptively change the window size to maintain a consistent amount of information in the window. For example, we increase the window size from (a) to (b) so more information about the trajectory can be included.

to show a copy of the occluded area underneath the finger. The position of the callout window is calculated to prevent clipping by the screen edges. When a user places a finger on the touch screen, a callout window appears on top of the finger, showing the occluded area under the finger. A crosshair in the center of the window represents the current selection point. The window is displayed as long as a user finger is in contact with the screen. It follows the user's finger to provide additional support for the occluded content. With this window, a user is able to avoid the occlusion problem by observing the content underneath and adjust the current selection during navigation. When a user releases the finger from the touch screen, the window disappears. Note that we made the content inside the callout window fully opaque so that it does not blend into the background and distract the user.

By default, we picked the window size to be slightly larger than the average fingertip size so that it can cover a part of the object and the motion trajectory segment around the object at the current position. For example, the default window size for Google Nexus 4 is $20mm \times 20mm$. The user only needs to observe a part of the motion trajectory around the object to decide where to move next.

Our pilot study showed that this technique allows a user to navigate along the motion trajectory very accurately. However, Shift was originally designed to address the occlusion problem for target acquisition, where the fixed target is usually the main concern of a user. In a DMVN system, in order to drag an object along its motion trajectory, a user needs to pay attention to the shape of the trajectory to follow it. While Shift can display a part of the trajectory in its callout window, due to the fixed window size, there is no guarantee that it can display a consistent amount of motion trajectory segment for a user to navigate the video. Figure 5 shows an example where two trajectory segments with different length are displayed in the same-size windows. Increasing the window size can increase the amount of motion trajectory inside the window and thus reduce the reorientation cost. But a large window is distracting as it occludes too much video content.

We developed a Window Drag technique to address this problem by adaptively changing the window size based on the local motion trajectory information. Window Drag finds such a callout window that contains a trajectory segment that covers at least 20 frames. If the default window does not cover such a trajectory segment, we expand the window to cover it.



(a) Start (b) End

Figure 6. Drag Anywhere for DMVN. Drag Anywhere allows a user to drag on any location and maps the user drag to the object trajectory.

Design 3: Drag Anywhere

When interacting with a touchscreen device, there are two different mapping modes: direct and indirect [16]. Direct mapping requires the pointer to be positioned directly under the finger, which leads to the fat finger problem. The indirect mapping, in contrast, does not require the pointer to be under the finger. Although not as intuitive as direct mapping, indirect touch clearly does not suffer from the occlusion problem. Previous research on touchscreen interaction [1, 3, 6] has already explored indirect mapping interaction techniques that adjust the control-display ratio to increase or decrease the pointer speed. This has been shown to improve precision in target acquisition tasks.

Inspired by Moscovich and Hughes [16], we designed an indirect mapping navigation technique for our touch-based DMVN system. Moscovich and Hughes argued that designing indirect mapping techniques needs to ensure the perceptuomotor compatibility of the interaction task, which is the degree to which the user's physical actions are similar to the system's visual feedback. Interestingly, this is also the case for designing the DMVN interface. As Dragicevic *et al.* mentioned, the directness of a DMVN interface also depends on the perceptuomotor compatibility [5].

Thus, when a user drags on the touch screen, we map the finger dragging motion to the object trajectory, as shown in Figure 6. Specifically, we compare the finger dragging direction to the tangent of the motion trajectory at the object's position to determine whether the object should be moving forward or backward in time. Then we map the dragging distance to the arc-length of the trajectory to determine how far the object should move along the trajectory. By using the orientation and distance as the mapping parameters, we ensure that the dragging movement closely matches the object movement, thus resulting in a smooth dragging experience without having to put the finger on top of the object.

One drawback of this technique is that a user is unable to quickly skim through the video, a common task in video browsing. This is because the object moves exactly the same distance as the user pointer. We overcome this problem by adaptively adjusting the CD ratio between the finger and the object of interest. This is similar to the classic "mouse acceleration" interface seen in Apple Mac Operating System. When a user moves the finger more quickly, the dragging distance is made larger so that the object can travel quicker along the motion trajectory. Our system keeps track of the finger dragging speed and uses this speed to compute a magnification coefficient to modulate the dragging distance. Our

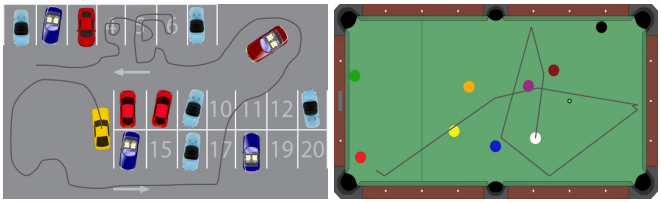


Figure 7. Sample frames of testing videos with trajectories.

system uses the following function to compute the magnification coefficient

$$m = \sqrt{s * a + 1}, \quad (1)$$

where m is the magnification coefficient for the dragging distance, s is the dragging speed, and a is a parameter that be adjusted by a user to control the scaling factor. Our pilot study shows that for the smartphone (Google Nexus 4) and the desktop touch-screen, good values for a are 0.25 and 6.67 respectively.

EXPERIMENTAL EVALUATION

We conducted a user study to evaluate our touch-based DMVN techniques: *Offset Drag*, *Window Drag*, and *Drag Anywhere* on both a large and small touch screen. Our study asked participants to perform spatial video navigation tasks: locate a frame with some specific visual event in a video.

For the baseline techniques, we used the basic DMVN system and the standard timeline slider. For simplicity, we use DMVN and Timeline to denote these two techniques, respectively. DMVN interprets the finger contact point as the pointer location and thus provides no additional support to handle the fat finger problem. Timeline is similar to the timeline interface used in YouTube Video Player on iPhone. It allows a user to offset the cursor on the timeline to address the occlusion problem. We want to understand the performance of DMVN-based techniques against Timeline for space-centric video navigation tasks on touchscreen devices. Early studies have shown that DMVN performs better than the standard timeline slider for space-centric tasks using a mouse or stylus as input; however, no study between them has been reported that directly uses a finger as input on touchscreen devices.

Note, the first step of direct manipulation video navigation is to select an object of interest to manipulate. As precise object selection has been well studied in previous research, we focus on dynamic object dragging during video navigation. Accordingly, an object of interest has already been selected for a user to manipulate beforehand. In the study, we only show the trajectory of the selected object and map the pointer movement to the nearest position on the trajectory.

We hypothesized that:

1. Techniques that can handle the occlusion problem (Timeline, Offset Drag, Window Drag, and Drag Anywhere) would outperform the basic DMVN for small-size objects.
2. The basic DMVN system would perform similarly to the DMVN systems supported by fat finger problem-handling techniques (Offset Drag, Window Drag, and Drag Anywhere) for big objects.

Screen	Car videos (V1, V2, V3)			Ball videos(V4, V5, V6)		
Small	1.4×3.2	5.8×12.8	9.7×21.3	1.5×1.5	9.6×9.6	15.5×15.5
Large	3.1×7.3	9.4×20.8	18.7×41.6	2.5×2.5	11.3×11.3	50.2×50.2

Table 1. Summary of object sizes (in *mm*) in the testing videos.

3. Enhanced DMVN techniques (Offset Drag, Window Drag, and Drag Anywhere) will be faster for dragging than Timeline for space-centric tasks on touch screen devices.

Apparatus

We used a large touchscreen display and a smartphone equipped with a touch screen. For the large display, we used a 23-inch Acer T232HL bmidz capacitive touch screen monitor. The monitor has a 1920×1080 resolution (95.8 pixels per inch). The size of each pixel is about 0.26 *mm*. We take a regular finger size to be 15 *mm* wide [3], which is about 58 pixels on this monitor. For the small display, we used a Google Nexus 4 smartphone running Android 4.3.3. This smartphone comes with a capacitive touch screen. Its active display area resolution is 1280×768 (320 pixels per inch). The size of a pixel is roughly 0.08 *mm*. The width of a regular finger on this small display is about 188 pixels.

Task and Stimuli

For each touch screen, we ask a participant to perform six tasks, each involving a different video. We then have six videos for each display. These six videos cover three object sizes (small, medium, large) and two types of scene content (parking lot and pool ball). The object motion for the parking lot videos is complex and that for the ball videos is simple, as shown in Figure 7. Since the two touch screens have different resolutions, the six videos used for the big screen are different from those for the small screen. We report the object sizes of these videos for each screen in Table 1. The videos range from 305 to 900 frames (10.2 to 30.0 seconds) and were shown at a display window of 1280×768 pixels on both screens. In each task, a participant was asked to use each of the five techniques to navigate a given video and locate a target frame with specific visual content as fast and as accurate as possible. Like the previous research [5], the starting frame in each task is the first frame.

Experimental Setup

There were 15 participants in our study, including 6 female and 9 male current or former students in our university. These participants come from a range of academic departments, including Computer Science, Business, Arts, Architecture, etc. The ages of these participants range from 20 to 30 years. All of them were familiar with touchscreen devices and were right handed. One participant was left handed but preferred to use his right hand for everyday work. We conducted a within-subject study with these users. We tested 5 techniques (Timeline, DMVN, Offset Drag, Window Drag, Drag Anywhere), 2 devices (large and small display), and 6 tasks. This resulted in a total of 60 data points per user.

Our study consisted of two stages. Each participant first performed all the tasks on one screen and then performed the tasks on the other screen. To reduce the carry over effect, after a participant finished studies on one screen, we let the participant rest for around 10 minutes. We used a 2×2 latin

square to counterbalance the study order of the large screen and small screen. At each stage, a participant performed the same task using the above mentioned five techniques. In order to reduce the learning effect, we counterbalanced the order of the techniques using a 5×5 latin square.

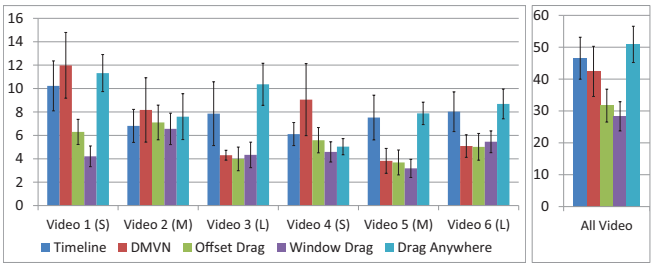
For each task, a participant watched the video twice and was shown the target frame together with the textual instruction to understand the task beforehand. The participant was given a maximal navigation time to finish the task. This maximal navigation time was set to be 2 times of the video length. The task session terminated if the participant successfully achieved the goal or ran out of time. We added a button in our testing interface to allow the participant to submit the navigation result. If the right frame was not reached, the participant was alerted to refine the navigation result by the system. The timer started counting when the participant first touched the screen and stopped counting when the submit button got hit and the right frame was reached. If a participant could not finish the task within the given time, we took the given maximal navigation time as the actual navigation time. We recorded not only the overall navigation time, but also the time a user took to reach the pre-defined neighborhood of the target frame, and the micro-navigation time a user spent inside the neighborhood around the target frame. The target neighborhood was defined to be slightly larger than a typical finger size. Specifically, the neighborhood radii for the small screen and big screen were 8 *mm* and 13 *mm*, respectively.

Results

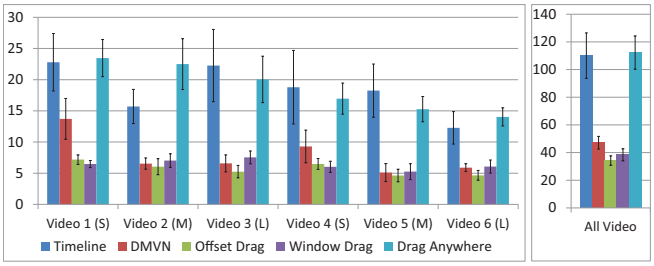
In our studies, all the participants successfully finished the navigation tasks on the large screen. On the small screen, 4 participants failed to reach the target frame within the given time on Video 1 using DMVN, 1 participant failed on Video 2 using DMVN, 1 participant failed on Video 3 using Timeline, and 2 participants failed on Video 4 using DMVN. We analyzed the temporal measurements with repeated measures ANOVA analysis and pairwise comparisons using t-tests with Bonferroni correction for post-hoc analysis.

Overall Navigation Time

We conducted one-way ANOVA analysis to test whether different methods result in different overall navigation times on both the large and small screen and found a significant effect of method on overall navigation time on both the large screen ($F(4, 56) = 94.13$, $p < 0.001$) and small screen ($F(4, 56) = 16.74$, $p < 0.001$). Figure 8 provides a summary of the performance analysis with respect to the overall navigation time for each of the five techniques. According to the navigation time, the relative performance between (basic) DMVN and Timeline method depends on the object size. On the small screen, DMVN actually took participants longer than Timeline on the small touch screen when the object was significantly smaller than the finger. According to the feedback from the participants, occlusion from the fat finger problem made it difficult for them to navigate to the exact location. When the object was big, DMVN then performed better than Timeline on the small screen. For all the videos on the large screen, they were longer than those on the small screen. At the same time, their objects are larger, as reported



(a) Small screen



(b) Large screen

Figure 8. Navigation time summary (seconds). Left: the average navigation time and 95% confidence interval on each video. Right: the sum of the navigation time on all the six videos for each technique.

in Table 1. As we will discuss in detail in the next subsection, the performance of DMVN, particularly considering the time to reach the neighborhood of the target location, depends less on the video length than Timeline. Then for all the videos on the large screen, DMVN was consistently faster. Looking at the sum of the navigation time on the large screen, the mean times for DMVN and Timeline were 47.1 seconds (SD=9.0) and 110.1 seconds (SD=32.5), respectively. The difference between DMVN and Timeline was strongly significant ($p < 0.001$).

The relative performance between two of our extended DMVN methods (Offset Drag and Window Drag) and DMVN also depended on the object size. When the object was small, Offset Drag and Window Drag could improve DMVN. For example, the average navigation times for DMVN, Offset Drag, and Window Drag for the small object size video (Video 1 on the small screen) were 12.0 seconds (SD=5.5), 6.3 seconds (SD=2.1), and 4.2 seconds (SD=1.8). The difference between DMVN and Offset Drag were strongly significant ($p < 0.001$), so was the difference between DMVN and Window Drag. When the object sizes were comparable to or bigger than the finger size, such as those in Video 3 and 6 on both screens, these three methods performed similarly and their differences were not significant.

On both the small and large screen, Offset Drag was slower than Window Drag when the object was small. We observe that when the object was small, participants tended to use the cursor, which took some time to first set up the cursor. In contrast, Window Drag did not have this cost. When the object became large, Offset Drag was faster. The differences between these two techniques were insignificant in both cases.

According to the sum of the navigation time, Drag Anywhere (M= 50.9, SD=11.3) performed consistently worse than both Offset Drag (M=31.7, SD=10.2) and Window Drag (M=28.3,

SD=9.1) on the small screen. The differences were strongly significant ($p < 0.001$). On the big screen, Drag Anywhere was also significantly slower than Offset Drag and Window Drag. We observed that using Drag Anywhere, participants needed to drag the object all along the trajectory carefully, while Offset Drag and Window Drag could both enable participants to quickly move the object through the trajectory or even jump to the neighborhood of the target location. The performance difference between Drag Anywhere and Timeline was insignificantly small. This is reasonable in that both techniques required participants to move through the trajectory until the neighborhood of the target location was reached.

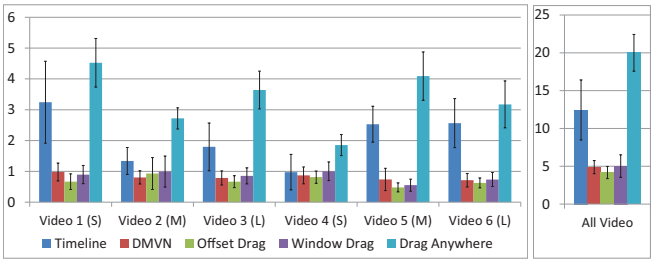
Time Needed to Reach the Target Neighborhood

It is interesting to look into how each technique supports two navigation tasks: quickly reaching the target frame neighborhood and carefully navigating in the neighborhood to reach the target frame. Figure 9 provides a summary of the performance analysis with respect to the time needed to reach the neighborhood of the target frame for all the five techniques. The neighborhood size is 100 pixels (8 *mm*) and 50 pixels (13 *mm*) for the tasks on the small screen and the large screen, respectively. The ANOVA analysis found a significant effect of method on the time needed to reach the target neighborhood on both the large screen ($F(4, 56) = 111.2$, $p < 0.001$) and small screen ($F(4, 56) = 45.58$, $p < 0.001$).

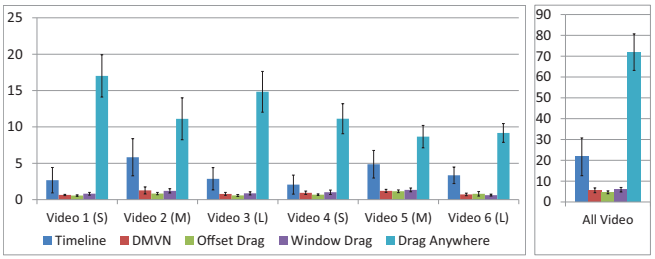
The three variations of the DMVN techniques, including DMVN, Offset Drag, and Window Drag, enabled a participant to quickly navigate to the neighborhood of the target location. For all the videos on both the small and large screen, these three techniques consistently took a participant around one second to reach the target neighborhood. This clearly showed the advantage of DMVN that allows for quickly dragging an object of interest to the target neighborhood.

The performance of Timeline depended on the temporal location of the target. If the target was temporally close to the starting point, such as those in Video 2 and 4 on the small screen, it took comparable time to the three DMVN methods, as shown in Figure 9. Otherwise, it took much longer to reach the neighborhood. For example, the time differences between Timeline and any DMVN methods were strongly significant on Video 1, 5, 6 on the small screen and Video 5 and 6 on the large screen ($p < 0.001$).

Drag Anywhere required a participant to literally go through the trajectory from the starting point until the target neighborhood was reached. When the spatial arc distance along the trajectory between the target location and the starting point was large, it took Drag Anywhere a long time to reach the target neighborhood. The time differences between each of the three other DMVN techniques (DMVN, Offset Drag, and Window Drag) and Drag Anywhere were all strongly significant with $p < 0.001$ according to the sum of the time needed to reach the target neighborhood, as shown in Figure 9. During the study, we also observed that when the trajectory was straight, participants could quickly move along the trajectory using Drag Anywhere by either flicking on the touch screen very frequently or using a quick long-range dragging action.



(a) Small screen



(b) Large screen

Figure 9. Time needed to reach the target neighborhood (seconds). Left: the average time and 95% confidence interval on each video. Right: the sum of the time on all the six videos for each technique.

But around the sharp corners, the participants often got stuck and sometimes even navigated in the opposite direction.

Micro-navigation Time in the Target Neighborhood

We are also interested in how each technique supports a user to perform micro-navigation in the neighborhood of the target frame. Figure 10 provides a summary of the performance analysis with respect to the time needed for micro-navigation in the target neighborhood. The ANOVA analysis found a significant effect of method on the micro-navigation time in the target neighborhood on both the large screen ($F(4, 56) = 50.6$, $p < 0.001$) and small screen ($F(4, 56) = 7.367$, $p < 0.001$).

When the object was small, it took participants longer to navigate to the final target location using DMVN than Timeline. For example, the object in Video 1 on the small screen was very small ($1.4 \times 3.2mm^2$), DMVN (M=11.0, SD=5.7) was significantly slower than Timeline (M=7.0, SD=3.2). The difference was significant ($p = 0.02$). Participants complained that it was difficult to use DMVN for micro-navigation due to the finger occlusion. For videos with big objects, such as V3, V5 and V6 on the small screen, DMVN performed better than Timeline. According to the sum of the micro-navigation time in the target neighborhood, DMVN (M=41.5, SD=8.2) was faster than Timeline (M=88.4, SD=29.7) with $p < 0.001$ on the large screen. While the objects in V1 and V4 on the large screen were smaller than the finger and the occlusion still compromised the user performance using DMVN, Timeline on the big screen caused participants to switch their attention frequently between the timeline and the object they were dragging according to their feedback after the study. This re-orientation cost compromised the user performance with Timeline. Besides, participants also reported that when they were dragging the timeline slowly and carefully, the friction between their fingers and the (big) touch screen often pre-

vented them from smoothly and continuously navigating the video at small steps. On the small touch screen on the smartphone, the friction was less an issue.

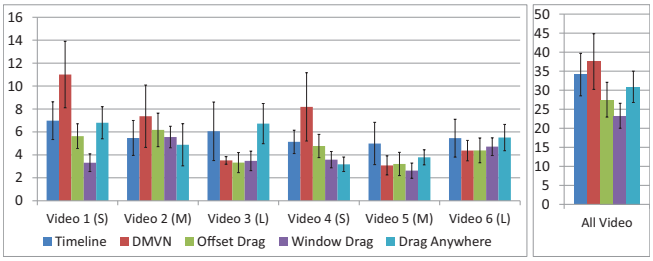
As shown in Figure 10, Offset Drag and Window Drag could both improve DMVN on videos with small objects, such as V1 and V4 on the small screen and V1 on the large screen. The time differences between these two techniques and DMVN on these videos were significant ($p < 0.05$). For videos with big objects that are comparable or larger than the finger, such as V3 and V6 on both screens, these three techniques performed similarly.

According to the sum of the micro-navigation time as shown in Figure 10, Drag Anywhere took less time than DMVN and Timeline, and took more time than Offset Drag and Window Drag on both screens. The time differences between Drag Anywhere and the others were not significant except that between Drag Anywhere and Timeline on the big screen. Looking at the data on each video, the performance of Drag Anywhere for micro-navigation varied and was highly dependent on individual video property. If the trajectory in the target neighborhood was smooth, users actually reported Drag Anywhere was very helpful as it did not have the occlusion problem, required no extra widget, and allowed users to drag on any location that they felt comfortable. In practice, a user can drag in an area which is close to the object, having the feeling of directness without any occlusion, distraction from extra widgets, or the re-orientation cost. However, when the trajectory has a sharp angle or a high curvature, Drag Anywhere often confused participants by accidently moving to the wrong direction. In many of the testing videos, the object trajectories were very complex, which contributed to the relative poor performance of Drag Anywhere compared to Offset Drag and Window Drag.

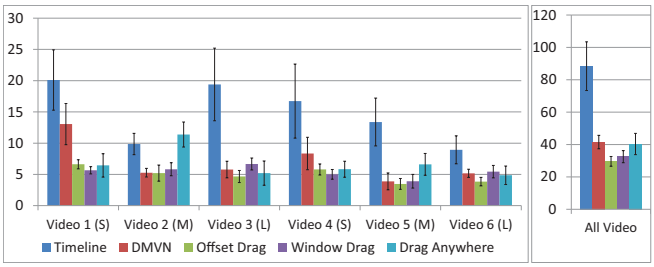
Summary

DMVN and two of our enhanced DMVN techniques (Offset Drag and Window Drag) can quickly navigate to the neighborhood of the target location without regarding to the object size. When the object is small, the occlusion problem compromises the performance of the basic DMVN for micro-navigation tasks. Offset Drag and Window Drag can effectively handle the occlusion problem. Overall, Offset Drag and Window Drag can better support space-centric video navigation tasks than Timeline for both small and big objects. In contrast, the basic DMVN method is helpful to the space-centric tasks only for videos with big objects. A common concern with Window Drag from participants is that the call-out window is sometimes distracting.

The performance of Drag Anywhere depends on the trajectory shape and the distance between the target location and the starting point. Around the trajectory segment that has a high curvature or sharp angle, Drag Anywhere is not convenient for users to drag an object. Accordingly, when the target is faraway from the starting point, Drag Anywhere cannot well support quick navigation to the target neighborhood as a user needs to literally move the object through the trajectory until reaching the target location. For a simple trajectory, Drag Anywhere can support quick navigation well. Participants also reported that Drag Anywhere has advantages, such as handling occlusion without any extra widget and allowing for dragging at any location on the screen.



(a) Small screen



(b) Large screen

Figure 10. Micro-navigation time in the target neighborhood (seconds). Left: the average time and 95% confidence interval on each video. Right: the sum of the time on all the six videos for each technique.

Discussion

We would like to note that this paper only studied the performance of each method for object dragging, although DMVN consists of two steps, object selection and dragging. As object selection on touch screens has been well studied, our study omitted this step. Although it will be certainly helpful to involve the object selection step in our studies, we feel that skipping this selection step will not severely jeopardize our study as all the DMVN methods covered in this paper require this step, which can be achieved using any of the state of the art precise selection methods. Thus our conclusion from this study can be applied to the complete DMVN procedures. Meanwhile, we need to clarify that the comparisons between DMVN and Timeline are affected by this omission, as Timeline does not require an object to be selected at all for navigation. We considered this affect during our design of this study and chosen our current design for the following reasons. First, object dragging is a unique problem of DMVN. Therefore, we wanted to focus on this step and make this already complex study controllable. We can borrow from the previous research for object selection performance analysis if needed. Second, this omission will not affect the comparison between different DMVN methods at all. Third, object dragging typically contributes to the majority of time cost of a DMVN task. So in general, our conclusion is still valid between the comparison between DMVN and Timeline although for some particular videos, the results for object dragging may have to be revised to apply to the whole DMVN procedure.

It could also have been nice that we could provide a performance curve that shows the effect of the object size on each individual technique in our study, like those in previous work on precise selection techniques. However, we found that this

is a very challenging task for our work on DMVN. Besides the object size, many factors of a video, such as the target location and the trajectory shape, can affect the performance of a video navigation technique. We could have designed a set of videos that differ from each other only in object sizes while keeping others the same. But it will introduce a very strong learning effect, which is difficult to eliminate as we need to use each of the video in the set to test five different methods and other similar videos in the set will be used to test these five techniques again by the same participant. We therefore only discuss the effect of the object size mostly in the context of comparing the relative performance between different methods and we feel that our findings from the current study is still interesting, and is helpful for us to understand the challenges of designing DMVN for touchscreen devices and evaluate our techniques.

Finally, when the object is small, all the DMVN methods in this paper require an object selection step, which typically involves a take-off action to indicate the selection of the object. This breaks DMVN into two separate steps and comprises the directness of DMVN. In future, we plan to eliminate this take-off action and combine object selection and dragging into a single continuous procedure to make DMVN smoother.

CONCLUSION

This paper discussed the effects of the fat finger problem on DMVN systems for touchscreen devices and developed three touch-based interface techniques to support DMVN, including Offset Drag, Window Drag, and Drag Anywhere. Our study showed that Offset Drag and Window Drag could effectively support DMVN on touchscreen devices and enable DMVN to outperform the traditional timeline based video navigation method for space-centric video browsing tasks. While Drag Anywhere did not perform as well as the other two techniques, it offered a few encouraging features.

Acknowledgments. The video and images in this paper are used from YouTube user MSPDawgs, Flickr user John Baer, Wikimedia Commons under Creative Commons licenses and from public domain (http://openclipart.org). This work was supported in part by NSF grants IIS-1321119 and CNS-1205746. Yuzhen Niu was supported by NSFC 61300102.

REFERENCES

1. Albinsson, P.-A., and Zhai, S. High precision touch screen interaction. In *ACM CHI* (2003), 105–112.
2. Baudisch, P., and Chu, G. Back-of-device interaction allows creating very small touch devices. In *ACM CHI* (2009), 1923–1932.
3. Benko, H., Wilson, A. D., and Baudisch, P. Precise selection techniques for multi-touch screens. In *ACM CHI* (2006), 1263–1272.
4. Cheng, K.-Y., Luo, S.-J., Chen, B.-Y., and Chu, H.-H. Smartplayer: user-centric video fast-forwarding. In *ACM CHI* (2009), 789–798.
5. Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. Video browsing by direct manipulation. In *ACM CHI* (2008), 237–246.
6. Forlines, C., Vogel, D., and Balakrishnan, R. Hybridpointing: fluid switching between absolute and relative pointing with a direct input device. In *ACM UIST* (2006), 211–220.
7. Goldman, D. B., Gonterman, C., Curless, B., Salesin, D., and Seitz, S. M. Video object annotation, navigation, and composition. In *ACM UIST* (2008), 3–12.
8. Grossman, T., and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *ACM CHI* (2005), 281–290.
9. Hasan, K., Yang, X.-D., Liang, H.-N., and Irani, P. How to position the cursor?: an exploration of absolute and relative cursor positioning for back-of-device input. In *ACM MobileHCI* (2012), 103–112.
10. Hürst, W., Götz, G., and Welte, M. Interactive video browsing on mobile devices. In *ACM Multimedia* (2007), 247–256.
11. Karrer, T., Weiss, M., Lee, E., and Borchers, J. Dragon: a direct manipulation interface for frame-accurate in-scene video navigation. In *ACM CHI* (2008), 247–250.
12. Karrer, T., Wittenhagen, M., and Borchers, J. Pocketdragon: a direct manipulation video navigation interface for mobile devices. In *MobileHCI* (2009), 47:1–47:3.
13. Karrer, T., Wittenhagen, M., and Borchers, J. Draglocks: handling temporal ambiguities in direct manipulation video navigation. In *ACM CHI* (2012), 623–626.
14. Kimber, D., Dunnigan, T., Girgensohn, A., III, F. M. S., Turner, T., and Yang, T. Trailblazing: Video playback control by direct object manipulation. In *IEEE ICME* (2007), 1015–1018.
15. Matejka, J., Grossman, T., and Fitzmaurice, G. Swifter: improved online video scrubbing. In *ACM CHI* (2013).
16. Moscovich, T., and Hughes, J. F. Indirect mappings of multi-touch input using one and two hands. In *ACM CHI* (2008), 1275–1284.
17. Nguyen, C., Niu, Y., and Liu, F. Direct manipulation video navigation in 3d. In *ACM CHI* (2013), 1169–1172.
18. Olwal, A., and Feiner, S. Rubbing the fisheye: Precise touch-screen interaction with gestures and fisheye views. In *ACM UIST* (2003), 83–84.
19. Pongnumkul, S., Wang, J., and Cohen, M. Creating map-based storyboards for browsing tour videos. In *ACM UIST* (2008), 13–22.
20. Potter, R. L., Weldon, L. J., and Shneiderman, B. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *ACM CHI* (1988), 27–32.
21. Ramos, G., and Balakrishnan, R. Fluid interaction techniques for the control and annotation of digital video. In *ACM UIST* (2003), 105–114.
22. Sears, A., Plaisant, C., and Shneiderman, B. Advances in human-computer interaction (vol. 3). 1992, ch. A new era for high precision touchscreens, 1–33.
23. Sears, A., and Shneiderman, B. High precision touchscreens: design strategies and comparisons with a mouse. *Int. J. Man-Mach. Stud.* (1991), 593–613.
24. Shi, J., and Tomasi, C. Good features to track. In *IEEE CVPR* (1994), 593–600.
25. Siek, K. A., Rogers, Y., and Connelly, K. H. Fat finger worries: how older and younger users physically interact with pdas. In *Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction*, INTERACT'05 (2005), 267–280.
26. Su, C.-H., Chan, L., Weng, C.-T., Liang, R.-H., Cheng, K.-Y., and Chen, B.-Y. Naildisplay: bringing an always available visual display to fingertips. In *ACM CHI* (2013), 1461–1464.
27. Sugimoto, M., and Hiroki, K. Hybridtouch: an intuitive manipulation technique for pdas using their front and rear surfaces. In *ACM MobileHCI* (2006), 137–140.
28. Sun, D., Roth, S., and Black, M. Secrets of optical flow estimation and their principles. In *IEEE CVPR* (2010).
29. Vogel, D., and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. In *ACM CHI* (2007), 657–666.
30. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. Lucid touch: a see-through mobile device. In *ACM UIST* (2007), 269–278.
31. Wigdor, D., Leigh, D., Forlines, C., Shipman, S., Barnwell, J., Balakrishnan, R., and Shen, C. Under the table interaction. In *ACM UIST* (2006), 259–268.
32. Wobbrock, J. O., Myers, B. A., and Aung, H. H. The performance of hand postures in front- and back-of-device interaction for mobile computing. *Int. J. Hum.-Comput. Stud.* (2008), 857–875.