Virtual Projection: Exploring Optical Projection as a Metaphor for Multi-Device Interaction

Dominikus Baur  
Media Informatics Group  
University of Munich, Munich, Germany  
dominikus.baur@ifi.lmu.de

Sebastian Boring  
Interactions Lab  
University of Calgary, Calgary, AB, Canada  
sebastian.boring@ucalgary.ca

Steven Feiner  
Department of Computer Science  
Columbia University, New York, NY, USA  
feiner@cs.columbia.edu

ABSTRACT
Handheld optical projectors provide a simple way to overcome the limited screen real-estate on mobile devices. We present virtual projection (VP), an interaction metaphor inspired by how we intuitively control the position, size, and orientation of a handheld optical projector’s image. VP is based on tracking a handheld device without an optical projector and allows selecting a target display on which to position, scale, and orient an item in a single gesture. By relaxing the optical projection metaphor, we can deviate from modeling perspective projection, for example, to constrain scale or orientation, create multiple copies, or offset the image. VP also supports dynamic filtering based on the projection frustum, creating overview and detail applications, and selecting portions of a larger display for zooming and panning. We show exemplary use cases implemented using our optical feature-tracking framework and present the results of a user study showing the effectiveness of VP in complex interactions with large displays.

Author Keywords
Interaction techniques, mobile devices, handheld projection.

ACM Classification Keywords
H.5.2 Information interfaces and presentation: User Interfaces—Input devices and strategies, Interaction styles, Graphical user interfaces.

General Terms
Design, Experimentation, Human Factors.

INTRODUCTION
Portable projectors in mobile devices provide a promising way to overcome screen-space limitations on handhelds [17], navigate information [12], or augment reality [5]. One of their appeals is the simplicity of interaction: Aiming at an appropriate surface projects the image, and changing posture and direction adjusts the image’s position and orientation. This behavior is purely based on optics, allowing us to intuitively grasp it based on our own experience with the physical world. However, strict adherence to the laws of physics also has its drawbacks: The intensity of light varies with the projector’s distance to the surface, and the projected image is tightly coupled to the projector’s movement.

In this paper, we apply the metaphor of optical projection to digital surfaces in the environment. We use a handheld device, tracked in 6 DOF, to support Virtual Projection (VP) on one or more displays (see Figure 1). The simulated nature of VP allows us to address some of the limitations of optical projection, avoiding unwanted distortions, jitter, and intensity variations, and eliminating the need to continually point the projector at the surface on which it is projecting. This also frees the frustum so that it can be used for selecting areas, either for navigation or for applying filters.

Our work makes several contributions: (1) We explore the implications of VP as an interaction technique and show how decoupling the projection from the projector and adjusting transformations can improve interaction. (2) We describe relevant characteristics of VP. (3) We present an implemented software framework for creating VP applications for consumer smartphones that does not require external tracking, and show exemplary use cases. (4) We report on a user study comparing VP with both absolute and relative techniques for content placement using a handheld device. Our findings suggest that VP is especially suitable for complex (i.e., translate-scale-rotate) projections.
VIRTUAL PROJECTION
As shown in Figure 1, VP can mimic the behavior of optical projections on digital surfaces. Aiming a tracked handheld device at one of the system’s secondary displays creates a simulated projection; moving the handheld controls the projection’s position, size and orientation—all with a single gesture. To support this, we track the mobile device in 6 DOF, based only on the device’s camera. By not using an external tracking system, we avoid the need for additional environmental infrastructure, allowing VP to run on any suitable mobile device with a live video camera and any available displays running the system.

Extending the optical projection metaphor
Emulating optical projection is powerful in its intuitiveness. However, VP enables two significant extensions to the optical projection metaphor: *adapting perspective transformation* and *decoupling projector and projection*.

Adapting perspective transformation. The tight coupling between projector and projection introduces distortion when the central axis of the projector frustum is not perpendicular to a planar projection surface. This effect is often referred to as *keystoning* and results in a wedge-shaped distortion of the ideally rectangular image in one or both axes. Tracking mechanisms based on cameras and fiducials [5] or other sensors [19] have been used to correct this for optical projectors by determining the transformation between the projector and the room.

Since projection is simulated in VP, we can modify the transformation at will, with full control over the results. Our implementation supports five different variations, as depicted in Figure 2: (a) Fully simulated optical projection with distortion is desirable only for a small set of use cases. (b) Considering only 2D position, scale, and rotation of the projected rectangle eliminates keystoning. (c) Discarding 2D rotation (produced by rotation around the central axis of the projection frustum) yields projections whose scale depends on distance to the display, but whose edges are always parallel to the display edges. (d) Ignoring the handheld’s distance to the display turns the handheld into a pure pointing device, controlling the position of a fixed-size rectangular window on the remote display. (e) Taking into account only the identity of the display that the handheld views, selects that display to be filled with the projection.

Which of these variations is best depends on the target display type as well as the application: For example, fixing orientation on a vertical desktop makes sense, as does scaling maps or background images to be full screen size.

Decoupling projector and projection. The rigid coupling between a handheld optical projector and its projection requires that the user aim where the projection is to appear. Combined with hand tremor, this also causes jitter [5,12,17]. In VP, the handheld device is not the source of the projection’s light, so we can completely decouple the virtual projector from its projection at any time. That is, we can allow the user to break the coupling, causing the projection to remain fixed to the secondary display. Users can also regain control when desired. In our implementation, we use a long-press gesture on the handheld’s touch screen for creating or reacquiring a projection. Releasing the touch screen fixes the projection or deletes it if ‘dropped’ outside of the display area. When a user leaves the environment, projections that are still fixed to the display remain behind.

Decoupling projector and projection has two benefits: (1) Users can create multiple projections and place them side-by-side, one at a time. (2) The handheld’s projection frustum can be used to specify a selection area on the secondary display, for interaction. The frustum can manipulate content either in a VP (e.g., applying an image filter to the area in the projection) or on the handheld (e.g., changing the portion of a map displayed on the handheld to correspond to the frustum’s location). With multiple VPs visible on the secondary display, the frustum (or its center point) can be used to select and activate one of them on the handheld.

Resulting Interaction
Figure 3 shows a walkthrough of our VP user interface: (a) To create or change to an existing view, users shake the mobile device to open the view manager. (b) Initially, a user views and interacts only with content on the mobile device. (c) If the device is aimed at a secondary display, a highlighted area is shown indicating the projection area. The highlighted area is tightly coupled to the mobile device and moves accordingly. (d) When long pressing on the mobile display (while aiming the device at the secondary display), local content from the mobile device is ‘projected’. (e) When the user lifts their finger, the projection is fixed on the secondary display and the mobile device can be moved freely without affecting it. (f) Now, both the local content and the projection can be controlled with either of the devices. (g) Alternatively, the projected frustum can be used as input to the secondary display or for navigating the view on the handheld. (h) Users can reposition or remove
the active projection (i.e., the one also shown on the mobile display) by aiming at the secondary display, and dragging the projection away using a long press.

RELATED WORK
VP allows both pointing at and transferring content to secondary displays in multi-display environments. It builds on work in mobile pointing devices, and interacting with real and simulated mobile projection, as well as cross-display interaction techniques.

Pointing with Mobile Devices
Several techniques to point at secondary displays using mobile devices have been proposed. Relative pointing is used to control a cursor on the display in a mouse-like fashion. Sweep uses optical flow analysis of the handheld’s live video [1]. Boring et al. additionally study using either the handheld’s joystick or its built-in accelerometers [9]. Both systems demonstrate the use of mobile cameras as suitable input sensors, but require tracking one’s cursor visually. Absolute pointing techniques avoid the use of pointers [24]. Pointing techniques for mobile devices also make use of such techniques. With Point & Shoot, users take a photo of a digital object they want to interact with [2]. Shoot & Copy eliminates the need for fiducial markers to determine the region pointed at by the device by using the display’s item arrangement instead [10]. Deep Shot further allows arbitrary screen content by matching the captured image to the remote display’s content [14]. None of these systems, however, allows continuous interaction with content.

In contrast, Pears et al. allow fluent interactions by continuously tracking a marker on the remote display [26]. Boring et al.’s Touch Projector avoids one movable marker per user by extending Shoot & Copy for continuous tracking [11]. Similar to DeepShot, Herbert et al. use a template-matching approach [18]. These techniques allow for area selection instead of using the center only. However, they all require a static sub-region on the display. VP builds on these earlier systems and represents a major advance: (1) The handheld tracks its spatial relationship continuously, unlike [1,10,14], (2) without the need for additional markers, unlike Pears et al. [26], and (3) handles arbitrary and dynamic screen content as opposed to the existing systems of which we are aware [11,18].

Simulated and Optical Mobile Projection
Projectors, both stationary and portable, extend the available screen size of handheld devices. Early systems utilized stationary projectors in combination with tracked handheld devices to simulate mobile projection. In Hotaru [34], a tracked mobile device controls the projection onto a table from an overhead projector. Blaskó et al. [8] prototyped a wrist-watch projector by tracking the wrist and simulating a projection from it onto a wall, using a floor-mounted projector. PenLight [31] simulated a pen projector by tracking a pen and projecting from a stationary projector onto paper on the table where the pen pointed. Each of these systems uses a limited space as projection canvas, but could conceptually be used on any surface in a room using the Everywhere Display [27]. Pico-projectors are now small enough to embed into handheld devices. Current projects focus on enlarging the handheld’s local content [17] or augmenting external information [30]. Built-in projectors (and their images) are tightly coupled to the device’s motion, introducing jitter and image distortions. Tracking the mobile device and its projector can reduce this [15]. Cameras are used to track a projector in space as well: Raskar et al. [28] and Beardsley et al. [5] demonstrate projected desktop applications and environmental augmentation.

Interacting with Mobile Projections
Several interaction techniques have been created using (simulated) mobile projection systems. MotionBeam makes
use of the projector’s movements for manipulating virtual characters [38]. Cauchard et al. break this tight connection to some extent by adding a controllable mirror [13]. Blaskó et al. utilized the relationship between the projector and projection surface for panning and zooming projected content, or interacting with it [8]. A projection can also be used as an interaction canvas; Similar to Hotaru [34], Cao et al. define virtual information spaces that can be uncovered using a handheld projector [12]. Different projection alignments create different meanings (and states respectively) during the interaction. Bonfire takes mobile projection even further by allowing users to directly interact within the projected image using touch [22]. In all these systems, however, the mobile device’s alignment is crucial at all times. In contrast, VP can decouple the projector and its projection.

**Cross-Display Interaction Techniques**

Projections make possible a new display that can be used to interact with other screens in the environment. Research on such cross-display interactions has been conducted with both additional virtual layers and active mobile devices—often relying on the concept of magic lenses [7, 36]: Benko et al. [6] and Spindler et al. [32] utilize mobile devices as peephole-like displays [39] to create additional information layers for existing displays. SecondLight supports similar interactions by using only passive sheets of paper [20]. Such “tangible views” are especially helpful in information visualization scenarios [33, 37]. VP can be used for the same interactions but does not require external tracking solutions as used by the aforementioned systems. Other projects investigate content transfer between two displays to either shorten the interaction distance [35] or make private information public [16]. Hyperdragging presents a bidirectional technique to transfer information from or to laptops in an indirect fashion [29]. E-conic displays content and cursors in a multi-display environment based on the user’s perspective [25]. These systems either require redirecting local input (e.g., PointRight [21]) or displays that are tracked at all times. Similar to DeepShot [14], VP overcomes this by using a direct pointing mechanism without the need for tracking the mobile device externally.

**CHARACTERISTICS OF VIRTUAL PROJECTIONS**

Virtual Projections can take different forms for different use cases. What they all share is the idea of utilizing available screen space on another display for making the interaction with the handheld more convenient, relying on the metaphor of projection to accomplish this. Based on experience with our implementation, we have attempted to categorize some of the different kinds of virtual projections by identifying the following set of characteristics.

**Content distribution**

A VP can be characterized by the distribution of content between the handheld device and the secondary display. This is also applicable to projector phones or setups where the portable device is connected to a larger screen. We define four approaches: replicated, extended, linked, and independent. When replicated, the VP contains an exact or scaled copy of the handheld’s content. This case is most commonly used with handheld projectors; for example, to show a photo displayed on the handheld. A VP may also show an extended version of the handheld’s content to better utilize the available screen space. For example, the handheld can show a small section of a map (focus), while the secondary display shows this section and its surroundings (context).

Both the VP and the handheld’s view can be linked. That is, both devices show related but different views on the same content. For example, the VP can display a video, while the handheld hosts its controls; the handheld’s frustum can be used as filter for a VP; or the handheld can act as a magic lens [7], displaying content based on its spatial relation to the secondary display.

Finally, the VP and the handheld’s view can be independent, with no relationship between them. This is the case when the secondary display shows several projections, only one of which is actively displayed on the handheld, the rest being independent.

**Coupling**

This second characteristic determines how closely the behavior of handheld and VP mirror real-world projections. Here we define three different stages of coupling between the handheld and its projection: full, restricted, and decoupled. When full coupling is used, the VP replicates a real projection’s behavior including all distortion (Figure 2a). This may be useful for simulating a real portable projector. Depending on the use case, the coupling can be partially restricted to avoid unwanted transformations (e.g., allowing translation, but ignoring rotation, as shown in Figures 2c–d). For example, this coupling allows for projecting an upright photo while manipulating only position and scale on the secondary display. When the VP and the projector are fully decoupled, the VP is used solely for selecting a display without influencing the content’s transformation (Figure 2e). This can be useful for full-screen presentation of content (e.g., a video).

**Projections as input**

The input characteristic determines if and how the projection can influence the handheld’s content. We define four possible variations: none, remote, snapshot, and continuous. If the VP provides no input at all (none), content on both the handheld and secondary display is independent. This is the case when several independent projections are shown on the secondary display. Alternatively, the VP can be manipulated on a remote display using a touchscreen or external input devices. That is, interaction that happens on the projection is also mirrored on the handheld. One example is using a projected web-browser with the secondary display’s keyboard and mouse (e.g., to change the URL or navigate to links).

The handheld’s frustum also allows for input. In snapshot
input, the intersection of the handheld’s frustum with the secondary display can select a VP’s sub-content by aiming at the projection and subsequently tapping on the mobile device. The selected region of a VP is then brought into focus on the handheld. The server only happens once (cf., 10, 14). An example is navigating within a large map to select a region of interest to be displayed on the handheld. Likewise, the handheld’s frustum can serve as continuous input for the mobile device. That is, while the handheld is aimed at a VP, the handheld’s content adapts to this VP based on its spatial relationship defined by the frustum. For example, this allows for augmented reality applications or magic lenses [7].

Manipulation
The central place for manipulating the VP content is the handheld. This can happen with six different degrees of input: none, indirect, direct, physical, pointing, frustum. When a VP’s content is static and non-interactive, no manipulation (none) takes place. As an example, this may be used for replicated photos. In linked views, the manipulation may happen in an indirect fashion through dedicated controls on the handheld. For example, users can manipulate a projected video with standard controls on their mobile device. In cases where the VP is visually synchronized to the handheld’s content, the VP can be manipulated in a direct fashion, either on the secondary display or on the handheld. The web browsing example mentioned earlier would make use of this type of content manipulation.

Similar to using the projection as input to the handheld, the frustum can be used for manipulation. The position and orientation of the handheld relative to the secondary display manipulates its content by physical motion. This interaction is common for position-aware optical projection, but naturally also works for VPs. Examples include Blasko et al.’s wrist-controlled simulated projector [8], Cao et al.’s information spaces [12], and Willis et al.’s MotionBeam for controlling virtual characters [38]. The handheld can further act as a pure pointing device. This may either happen at all times (e.g., the center of the handheld always creates a cursor) or be triggered through touching the device (similar to TouchProjector [11]). One prominent example is activating a projection on the handheld by tapping. Finally, the entire frustum (controlled by the handheld’s position and orientation) can be used for manipulating a VP. This may be useful to apply imaging filters to a projected photo.

IMPLEMENTATION
We developed a prototype framework with which to investigate the abovementioned characteristics. Our prototype uses a client–server architecture: One secondary display and its computer acts as the server, with multiple clients (Apple iPhone 4) connecting to it. Once connected, a client streams its video data wirelessly to the server, which handles feature tracking and calculation. The client is notified of the current spatial relationship when calculation is done. All client interactions are routed through the server, as they may affect the behavior of a synchronized projection on the secondary display.

Tracking Spatial Relationships to Displays
We wanted to enable our system to work with various displays without modification. For this reason, we chose not to use external tracking hardware (e.g., [23]). Instead, we rely solely on the handheld’s built-in camera (i.e., its live video feed) and the secondary display’s visual content. Most existing approaches for this kind of tracking make use of markers either physically attached around the display or digitally superimposed on the screen’s content [1,26]. Although these approaches can track well, the markers must be visible to the mobile camera at all times and thus clutter the display or its surrounding frame.

To avoid this, we employ a markerless approach similar to Herbert et al. [18]. We use speeded-up robust features (SURF) [4] to find the secondary display’s visual content (i.e., its screenshot) in a given video frame from a mobile device. When the content is found, SURF responds with the homography (i.e., transformation matrix) between both image planes. Our algorithm then uses this matrix together with the display’s physical size and resolution to calculate the pose, allowing for full 6 degrees of freedom. Since our system significantly modifies the display’s content throughout the interaction, we had to extend this basic approach: The display continuously updates its image features (by periodically taking screenshots and calculating image features) for later template matching using SURF. As wireless transmission of video frames introduces an additional delay, devices have to be synchronized and screenshots and video frames mapped using time stamps. A queue stores screenshots (and calculated image features) of at least the past two seconds to handle delayed video frames. When a new frame arrives, the display selects the screenshot closest to the current frame to minimize the offset. As a result, our system represents a novel real-time markerless tracking approach that supports continuous feedback and dynamic screen content.

Performance and Limitations
Our current implementation uses an iPhone 4, 160×120 color photos (scaled on the phone), and transmission via Wi-Fi to a MacBook Pro (2.4 GHz Core Duo, 4GB RAM) server running Windows 7. It achieves ca. 10 frames per second (fps) for transmitting an image and 8 fps for tracking, resulting in a delay of 100–130ms. This performance will increase with new transmission technologies, as well as faster processors in both mobile devices and desktop computers. Several drawbacks result from the restrictions of a consumer device. First, feature-based tracking requires display content that contains rich features [26]. Through experimentation we found that this is rarely an issue in practice, provided we avoid roughly uniformly colored backgrounds. Second, our current implementation captures up to 12 screenshots per second. If content is changing more rapidly (e.g., a video with fast changes in scenes),
calculation errors are introduced. Third, the wide-angle lenses in today’s mobile devices limit the operational range of our system to less than 10’ on a standard 24” diagonal display. We expect that future improvements in mobile devices will transcend each of these limitations.

VIRTUAL PROJECTION EXAMPLES

Using our framework, we prototyped exemplary VP use cases. These can be used to create synchronized projected views by aiming the handheld device at a display and long pressing (approx. 1 second) the touchscreen. Switching between views and navigating within them happens by aiming at a view on the secondary display and tapping. Alternatively, users can shake the device to open up the view manager showing all available views. This manager also enables users to create a new projection (e.g., Figure 4 left). Here (and in the video), we show these example use cases, which represent starting points to demonstrate the potential of VP. Each can be classified by the characteristics we discussed earlier, as summarized in Table 1.

<table>
<thead>
<tr>
<th>Example</th>
<th>Content</th>
<th>Coupling</th>
<th>Input</th>
<th>Manipulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notepad</td>
<td>Replicated</td>
<td>Restricted</td>
<td>None</td>
<td>Direct</td>
</tr>
<tr>
<td>Photo Viewer</td>
<td>Extended</td>
<td>Restricted</td>
<td>Snapshot</td>
<td>Frustum-based/Direct</td>
</tr>
<tr>
<td>Web Browser</td>
<td>Extended</td>
<td>Restricted</td>
<td>Snapshot/Remote</td>
<td>Frustum-based/Direct</td>
</tr>
<tr>
<td>Video Player</td>
<td>Linked</td>
<td>Decoupled</td>
<td>None</td>
<td>Indirect</td>
</tr>
<tr>
<td>Maps</td>
<td>Extended</td>
<td>Restricted</td>
<td>Continuous</td>
<td>Frustum-based/Direct</td>
</tr>
<tr>
<td>Photo Filters</td>
<td>Extended</td>
<td>Full</td>
<td>Snapshot</td>
<td>Frustum-based/Direct</td>
</tr>
</tbody>
</table>

Table 1 - VP characteristics of the example use cases.

**Photo Viewer.** Instead of simply cloning the visible content, views that do not fit on the small display can be extended—especially when details should be preserved. In this scenario, the entire content (not just the visible portion) of the mobile device is cloned to the large display. The projection also denotes the area that is currently visible on the mobile device using a thin white frame (Figure 5 left). Users can interact with this projection in two ways: Manipulating (i.e., panning or zooming) the view on the mobile device updates the selection frame on the large screen, and using the projection frustum selects a new region to be shown on the mobile device.

**Web Browser.** Similar to the Photo Viewer example, users can project web pages on the secondary display. The visual representation remains the same; a white border denotes the mobile’s visible sub-region (Figure 6 right). While conceptually similar, the web browser can be controlled on both displays: on the mobile device by navigating it as if there would not be a projection, and on the secondary screen using a mouse and keyboard. When users click on links (and enter a new URL in the address bar), the website is updated synchronously on both devices.

**Video Player.** This example makes use of content distribution. Users are able to project their video onto the large display. However, the video is then no longer shown on the mobile device. Instead, the mobile device presents controls to manipulate the video (i.e., basic playback, seek, and volume). Thus, the handheld turns into a temporary remote

**Notepad.** A very simple, yet powerful use case is to clone the mobile device’s visible content. This resembles the core idea of mobile optical projection. However, decoupling the projection from the mobile device allows for subsequent interactions. In this example, users can create Post-it-like notes (Figure 4 right). That is, they can write messages and place them on the large screen. Furthermore, as projections are synchronized, users can change the content after the note is projected, which is immediately visible on both the mobile device and the secondary display.

**Figure 5.** (left) Creating new views. (right) Notepad view.

**Figure 6.** (left) Extended photo view. (right) Using the frustum to apply image filters.

**Figure 7.** (left) Video with remote controls. (right) Web view.

**Figure 8.** (left) Extended map view. (right) Magic lens for map.
control, as long as the projection remains on the large display (Figure 6 left). The only content that can be actively distributed by users is the movie’s volume: the output is either on the large display’s speakers, the mobile device (e.g., on attached earphones), or both.

Maps. The rather small display of mobile devices requires a large number of panning and zooming operations within maps. In this example, the entire map is projected onto the display with the mobile device again showing a sub-region in higher detail (Figure 7). In contrast to previous extended views (i.e., Photo View, Web Browser), the handheld is now used as a magic lens [7]. That is, the mobile view is continuously updated in real-time by moving the handheld in front of the projected map. Zooming is then achieved by moving the handheld closer to or further away from the display. This resembles a behavior similar to focus (mobile) plus context (large display) screens [3].

Photo Filters. Instead of changing the content on the mobile device through motion, a projection’s visual appearance can also be changed. In this example, users can select an image filter (e.g., color, grayscale, invert, and alpha) on the mobile device and apply it to a photo shown on the secondary display. The filter’s area is defined by the projector’s frustum and updated in real-time (Figure 5 right). It affects all projections that allow filtering once it overlaps the projection. That is, if the filter reaches into two different projections, both may be filtered accordingly. Multiple filters can be composed by placing them above each other.

USER STUDY
We designed a user study to determine how VP compares with existing techniques for selecting a portion of a display using a handheld device. As VP represents an area-pointing technique (i.e., it selects an area rather than a single point), we decided to use a targeting task, and measure speed and accuracy.

Independent Variables: Technique and Projection
We developed four different Techniques: two variants of VP and two controls (one absolute and one relative technique). VP (Pure) places a source quadrilateral, using the position and orientation of the handheld device relative to the large display. While aiming, a yellow border is shown on the target display. A participant must do a long-press and release (lift) to place the projection. A second VP technique, VP (Thumb) adds touch-based interaction to VP (Pure): Once a participant puts a finger (typically the thumb) on the handheld screen, they can additionally change the translation of the projection by moving the finger. Placement happens when the finger is lifted. Minimap shows a miniature representation of the large display on the handheld and uses absolute mapping. The source initially appears in the center of the large display and can be dragged using one finger or rotated/scaled using a two-finger pinching gesture. A double-tap confirms the placement of the projection. Touchpad is a relative input technique in which the entire handheld touchscreen can be used for dragging (single finger) or rotating/scaling (pinching) the source. As with Minimap, a double-tap confirms placement of the source.

We decided to test three different Types of Projections: Translate-Scale-Rotate (Figure 2b), Translate-Scale (Figure 2c) and Translate (Figure 2d). For each type of projection, only the specified transformations are applied to the projected source.

![Figure 9. User study compared four different techniques with three different types of projections for speed and accuracy.](https://via.placeholder.com/150)

Task
We used a basic targeting task inspired by existing Fitts’s Law tests (e.g., ISO 9241-9). However, we did not have a distinct target point, but a target quadrilateral: We asked participants to try to align the projection as well as possible to this target, while being as fast as possible. Boring et al. introduced the term docking offset for the error created in such targeting scenarios, defining it as the “percentage of the object located outside the target area” [11]. We computed docking offset, the distance between the centroids of the projection and target, and the differences in scale and orientation as error measures.

At the beginning of each task, participants had to press a start button on the mobile device. They then placed the object inside the target area using the given technique and type of projection. Once the target was placed (i.e., either by lifting the finger in both VP conditions or performing a double-tap in the other two conditions), the handheld switched back to the start button. We measured task time from pressing the start button till placement.

All types of projection were tested using one technique before switching to the next technique. After each technique, participants answered a short questionnaire about the technique’s applicability to the task and their fatigue.

Design & Hypotheses
We used a within-subjects design, counterbalancing Technique across participants using a Latin Square. For each combination of technique and type of projection we had four blocks. A block contained four trials, each of which had a different target position and shape or orientation respectively. These transformations were randomized within a block and the first block was used as training for the participants and discarded from the analysis. We therefore had a total of $4 \times 3 \times 3 \times 4 = 144$ trials. Dependent variables were speed and accuracy.

We had three hypotheses: (H1) For complex projections (Translate-Scale-Rotate) we expected the VP techniques to
outperform the other two conditions in terms of task time due to less required movements. (H2) In terms of coarse positioning (time to approach the target), we hypothesized that both VP techniques would outperform the other two techniques. (H3) We further expected that VP (Thumb) would result in more accurate placement compared to VP (Pure) due to fine-grained control by the thumb.

**Apparatus & Participants**
We used an Apple iPhone 4 (3.5” diagonal, 960×640 resolution) as handheld and an LC-Display (24” diagonal, 1440×900 resolution) as secondary display (Figure 8). For the VP conditions, we used our feature-based tracking approach. To assist the tracking algorithm, we used a display-filling, feature-rich background image. We logged all interaction with the handheld device (i.e., all touch points) and all tracking results for each frame on the server.

We recruited a total of 12 participants (3 female, 11 right-handed, ages 22–30, average age 26.8 years) from our institution. Each participant took about 45 minutes to complete the study including post-questionnaires.

**RESULTS**
We compared overall task time and docking offset as well as coarse positioning with separate repeated ANOVA measures. For pair-wise post hoc tests we compared against an α of 0.05 that was Bonferroni-corrected for repeated measures based on the number of repetitions. All unstated p-values are p < 0.05.

![Figure 10. Results for time (left) and accuracy (right)](image)

The reason for this interaction can be seen in Figure 9 as the increasing “complexity” of the projection type (from Translate to Translate-Scale-Rotate) influences task times. While all techniques performed nearly equally for Translate (all p > 0.05), task times of both Minimap and Touchpad gradually increase with Projection complexity. For Translate-Scale, VP (Pure) was faster than Minimap (p < 0.002), and for Translate-Scale-Rotate, both VPs were faster than Minimap (all p < 0.001) and Touchpad (all p < 0.005). All p-values were compared against a Bonferroni-corrected α = 0.0083.

Overall, VP (Pure) was fastest (M=6426 ms, SD=473 ms), followed by VP (Thumb) (M=6525 ms, SD=544 ms), Touchpad (M=6994 ms, SD=605 ms), and Minimap (M=8172 ms, SD=432 ms). However, only Minimap differed significantly from the two VP conditions.

**Docking Offset**
We aggregated docking offsets across blocks and performed a 4 × 3 (Technique × Projection) within-subjects ANOVA. We found significant main effects for Technique (F_{2,33} = 15.870, p < 0.001, only Translate differed from the other types of projections with p < 0.001). Furthermore, we found a significant interaction between Technique and Projection (F_{3,66} = 9.309, p < 0.001).

![Figure 11. Times to reach 50%, 25%, and 20% docking offset (= error) for the three different types of projections](image)

Both VPs are slightly less accurate than Minimap and Touchpad for almost all Projections. However, all interfaces still achieved very high accuracies of over 92% (docking offset < 10%), which is sufficient for many real-world uses. In total, Touchpad had the lowest docking offset (M=2.5%, SD=0.7%) followed by Minimap (M=4.7%, SD=0.8%), VP (Pure) (M=5.8%, SD=0.5%), and VP (Thumb) (M=7.1%, SD=0.8%). Explanations might be that the tracking sometimes was inaccurate and that the VP techniques did not allow correcting a placement after lift, unlike the other two.

**Coarse Positioning**
As the VP techniques were faster than both Minimap and Touchpad at the cost of slightly higher offsets, we addition-
ally evaluated the “homing speed” (i.e., the relationship of time and docking offset). We suspected that both VP techniques would reach a coarse position even faster than the control techniques, but take longer for fine adjustment (thus resulting in a smaller overall speed advantage). In order to test this, we defined Offset (50%, 25%, and 20%) as predefined docking offsets and measured the time from beginning of a trial until each offset was reached. With this, we performed a $4 \times 3 \times 3$ (Technique $\times$ Projection $\times$ Offset) within-subjects ANOVA. We found a significant main effect for all independent variables ($p < 0.001$). We also found significant interactions for all combinations ($p < 0.001$). From Figure 10, we see that the time until an Offset is reached increases with Projection complexity for both Minimap and Touchpad. Simple Projections did not influence homing speed of different techniques for all Offsets ($p > 0.05$). In contrast, post hoc multiple means comparisons (against corrected $\alpha = 0.0083$) of medium-complex Projections (Translate-Scale) showed that VP (Pure) outperformed both Minimap and Touchpad ($p < 0.002$), whilst VP (Thumb) only showed a significant difference compared to Minimap for 25% and 20% ($p < 0.002$). For further increase in complexity, post hoc tests revealed that both VPs are faster than Minimap ($p < 0.001$) and Touchpad ($p < 0.004$) for all Offsets.

DISCUSSION

Our results show that both VP techniques significantly outperform Minimap and Touchpad for complex Projections, supporting H1. Overall, for complex Projections, the VPs were 19.2%/23.2% faster than Touchpad and 28.9%/32.5% faster than Minimap. The noticeably higher performance of Touchpad compared to Minimap can be explained by participants’ familiarity with touchpad devices, and the display’s resolution and resulting fat-finger problem on the handheld’s display (Minimap).

In terms of coarse positioning, both VPs significantly outperformed the other two interfaces for complex transformations, as we had expected (H2). Our assumption is further supported by the fact that the virtual interfaces outperformed the Minimap interface for Translate-Scale projection types at all Offsets. We also assumed that there would be not much difference among the interfaces when the projection type allows only simple transformations (i.e., Translate). However, we note that with less complexity, Minimap and Touchpad perform equal to or even slightly better than both VPs. Nevertheless, using only Translate will rarely be the case (e.g., when projecting on a tabletop).

Overall docking offset revealed that the VPs perform slightly worse (5.8–7.1%) than the other interfaces (2.5–4.7%). Most surprising, however, is that VP (Thumb) performed worse than VP (Pure), although fine-grained positioning would have been possible. Thus, we rejected H3. One explanation may be the interference between device and finger movement, which both controlled the result (i.e., absolute and relative positioning are performed simultaneously).

A sequential approach (e.g., doing absolute positioning only until the finger moves on the handheld) may overcome this at the expense of higher task times. While this may increase accuracy, we believe that, at least in the presented use case, accuracies > 92% are still acceptable.

CONCLUSIONS AND FUTURE WORK

We presented Virtual Projection, an interaction metaphor for information transfer from handhelds to digital surfaces, inspired by optical projection. The central aspect of decoupling the projection from the handheld device allows using the frustum for interactive filtering or for selecting subregions of the display for navigation. We implemented an optical-feature–based tracking framework for smartphones and used it to apply VP to a wide range of static and dynamic screen content. A user study showed the effectiveness of VP for performing complex placements with a single gesture. Our software and tracking framework allows quick creation of additional virtual views, so we plan to explore this concept further: Our prototype already supports multiple clients, but only some views (e.g., image filters) allow meaningful collaboration. Mobile projection can benefit multi-user scenarios, so using VP there is worth exploring. As handhelds stay synchronized to projections, interaction could even happen remotely. Additionally, interacting with a handheld device brings with it implicit user identification for CSCW. Another direction to explore is 6DOF device tracking. While we have only projected onto 2D surfaces, we could create interaction techniques relying on the full 3D frustum. The frustum could be used to place 3D objects, cut through 3D scenes, or control avatars.

REFERENCES


