Virtual Projection: Exploring Optical Projection as a Metaphor for Multi-Device Interaction
Baur, Dominikus; Boring, Sebastian; Feiner, Steven

Publication date: 2012

Document Version
Early version, also known as pre-print

Citation for published version (APA):
Virtual Projection: Exploring Optical Projection as a Metaphor for Multi-Device Interaction

**Dominikus Baur**  
Media Informatics Group\(^1\)  
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 demonstrating the effectiveness of VP in complex interactions with large displays.

**Author Keywords**  
Interaction technique; mobile device; handheld projection.

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

**INTRODUCTION**  
Portable projectors in mobile devices provide a 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 based purely on optics, allowing us to intuitively grasp it, based on our own experience with the physical world. Naturally, the laws of physics also bring drawbacks; for example, that 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 flexibly adjust the parameters 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 present a real-time tracking framework for creating VP applications that do not require external tracking or markers, and works with consumer smartphones on dynamic display content. (3) We describe relevant characteristics of VP and present exemplary applications. (4) We report on a user study comparing VP with both absolute and relative techniques for content placement using a handheld device.

---

1 This work was carried out while the first author was visiting Columbia University.
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 projection 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 handheld in 6 DOF, based only on the device’s camera. Not using an external tracking system avoids 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.

**Terminology**

Throughout this paper, we use the following terms:

- **(Virtual) Projection (VP):** A set of data and its user interface, presented on a secondary display using a metaphor inspired by optical projection.
- **Handheld (Device):** The handheld device from which a VP originates.
- **(Secondary) Display:** A target display that can show a VP.
- **Frustum:** The projection frustum originating from the handheld and aimed at a secondary display.
- **View:** The data and user interface shown on a handheld device and/or within a VP. Views on a handheld are always displayed full-screen.
- **Active View/Active VP:** A view that is currently shown on both handheld and secondary display. Only one view can be active on a handheld at one time, while a secondary display might show additional inactive views.
- **Fixed VP:** A VP that is fixed to a secondary display and, thus, decoupled from the handheld.

**Extending the optical projection metaphor**

By simulating projection, VP enables two 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, 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 is not the source of the projection’s light, so we can completely decouple the virtual projector from its projection at any time. We allow the user to break the coupling, fixing the projection to the secondary display. Users can also regain control when desired. In our implementation, a long-press gesture on the handheld’s touchscreen creates or reacquires a projection. Releasing the touch fixes the projection or deletes it if ‘dropped’ outside 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 select an 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) or on the handheld (e.g., changing the portion of a map displayed on the handheld to correspond to the area). With multiple VPs on the secondary display, the frustum (or its center point) can be used to select a VP and activate it on the handheld.

![Figure 2. Types of projections. VP applies selected aspects of the device's pose to the image transferred to a display: (a) optical-projection transformation with keystone distortion, (b) optical-projection transformation minus keystone distortion, (c) translation and scaling only, (d) translation only, (e) full-screen (display selection only).](image-url)
Point & Shoot
and direct pointing techniques avoid using a display [2]. Jeon et al. use this approach to initially select an object, which can then be manipulated using optical flow analysis [21]. Shoot & Copy eliminates the need for fiducial markers to determine the spatial relationship between the handheld and display through content arrangement [10]. Deep Shot further allows arbitrary screen content by matching the captured image to the remote display’s content [14]. Each of these systems, however, only allows for discrete selection, instead of continuous interaction.

In contrast, Pears et al. allow fluent interaction by continuously tracking a marker on the remote display [28]. Boring et al.’s Touch Projector avoids one movable marker per user by extending Shoot & Copy for continuous tracking [11]. Like DeepShot, Herbert et al. use template matching [18]. While these techniques allow area selection instead of using the center only, they all require a static display subregion. VP builds on this earlier work and represents a major advance. VP (1) tracks the handheld’s spatial relationship continuously, instead of using a single image [1,10,14], (2) needs no additional markers, unlike Pears et al. [28], (3) handles arbitrary, dynamic screen content, unlike all existing systems of which we are aware [11,18], and (4) does not require projected infrared light [39].

Simulated and Optical Mobile Projection
Early systems combined stationary projectors with tracked mobile devices to simulate mobile projection [34]. Blasko et al. [8] prototyped a wrist-watch projector with a floor-mounted projector simulating a wall projection from the tracked wrist. PenLight [32] simulated a pen projector by tracking a pen and projecting on the table where the pen pointed. These systems have a constrained projection canvas, but can conceptually be extended to any surface, in the spirit of Everywhere Displays [29].

Pico-projectors are now small enough to be embedded in handhelds. Current projects focus on enlarging the

Figure 3. VP walkthrough: (a) Shaking the device to create a view. (b) Interacting with a non-projected view. (c–e) Creating a projection by aiming at the secondary display, long pressing, and releasing. (f) Synchronized interaction. (g) Projection frustum can be used for filtering or navigating. (h) Projections can be moved or deleted by aiming, long pressing, and dragging out of the display.
handheld’s local content [17] or augmenting external information [31]. However, built-in projectors (and their images) are tightly coupled to device motion, introducing jitter and image distortion. Tracking the handheld and its projector can reduce this [15]. Cameras are also used to track a projector in space: Raskar et al. [30] and Beardsley et al. [5] demonstrate projected desktop applications and environmental augmentation. Cauchard et al. partially break the tight connection by adding a controllable mirror [13].

**Interacting with Mobile Projections**

Mobile projection systems make possible interesting interaction techniques. *MotionBeam* uses the projector’s movements for manipulating virtual characters [38]. Blaskó et al. utilize the relationship between the projector and projection surface to pan, zoom, and interact with projected content [8]. Similar to *Hotaru* [34], Cao et al. use projections as an interaction canvas: virtual information spaces are uncovered using a handheld projector, with different projection alignments creating different meanings [12]. Bonfire takes mobile projection even further: users directly interact with the projected image using touch [23]. In each of these systems, however, the mobile device’s alignment is always crucial, while VP allows decoupling projector and projection.

**Cross-Display Interaction Techniques**

Projections create a new display that can be used to interact with other displays. Research on cross-display interactions addresses 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. [33] use mobile devices as peephole displays [41] to create information layers atop existing displays. In *SecondLight*, passive sheets of paper allow for similar interactions [20]. VP can be used for the same interactions but differs in that the device tracks itself instead of being tracked by the secondary display.

Other projects investigate content transfer between two displays to shorten the interaction distance [35] or make private information public [16]. Placing handhelds directly on interactive surfaces allows for temporarily displaying personal, mobile content (e.g., [27,40]). These systems either redirect local input (e.g., *PointRight* [22]) or displays that are tracked at all times. Similar to *DeepShot* [14], VP overcomes this by direct pointing.

**CHARACTERISTICS OF VIRTUAL PROJECTIONS**

VP can take different forms for different use cases. What they all share is applying the metaphor of projection to use available screen space on a secondary display to improve interaction with the handheld. Based on experience with our implementation, we have categorized some of the different kinds of VP by identifying the following set of characteristics. Items in each category are mutually exclusive.

**Content distribution**

A VP can be characterized by the distribution of content between the handheld device and secondary display. This is also applicable to projector phones or setups where the portable device is directly 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 typical of real handheld projectors (e.g., the handheld and VP both show a photo). A VP may also show an *extended* version of the handheld’s content to better utilize available screen space (e.g., handheld: focus section of a map, VP: focus section plus context). Both the VP and the handheld view can be *linked*. That is, both devices show related but different views of the same content (e.g., handheld: video, VP: controls; the frustum can also act as filter on a VP or turn the handheld into a magic lens [7]). Finally, the VP and the handheld’s view can be *independent*, with no relationship between them (e.g., the handheld’s view is switched, leaving a disconnected inactive VP on the secondary display).

**Coupling**

This characteristic determines how closely the behavior of handheld and VP are coupled geometrically: full, restricted, and decoupled. In *full* coupling, the VP replicates a real projection’s behavior including all distortion (Figure 2a); e.g., to simulate a real portable projector. Coupling can be partially *restricted* to avoid unwanted transformations (e.g., allowing translation, but no rotation, as shown in Figures 2c–d); e.g., to display an upright photo with flexible position and scale. 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); e.g., to show a full-screen presentation or video.

**Input (Changing the handheld view via the projection)**

This characteristic determines if and how the VP can influence the handheld view. We define four possible variations: none, remote, snapshot, and continuous. The VP might provide no input at all (none) for non-interactive content (e.g., photos) or when a VP shows an inactive view. Alternatively, remote interaction on the projection is mirrored on the handheld (in replicated/extended/linked content distribution); e.g., a projected web browser used with the secondary display’s keyboard and mouse to change the URL.

The frustum created by the handheld can also be used to select regions of the (fixed) VP, either discretely or continuously. In the discrete snapshot case, aiming and tapping on the handheld makes the selected region the focus [10, 14]; e.g., navigating within a large map to select a region of interest for display on the handheld. Likewise, the handheld’s frustum can serve as continuous input; e.g., for augmented reality or magic lenses [7]).

**Manipulation (Changing the projection via the handheld)**

The central locus for manipulating a VP’s content is the handheld. We identify six different ways: none, indirect, direct, physical, pointing, and frustum. When a VP’s content is static and non-interactive, no manipulation (none) takes place; e.g., this may be used for replicated photos. In linked views, indirect manipulation may occur through dedicated controls on the handheld; e.g., users can manipulate a projected video with standard controls on their
handheld. If the VP’s and handheld’s views are synchronized, direct manipulation of the VP can occur on the secondary display or the handheld. The abovementioned web-browsing example uses this type of content manipulation.

Similar to using the projection as input to the handheld, the frustum can be used for manipulation. The handheld’s position and orientation relative to the secondary display can manipulate its content by physical motion, similar to position-aware optical projection \([8,12,38]\). The handheld can further act as a pure pointing device, either at all times (e.g., the center of the handheld always creates a cursor) or when triggered by touching the device (similar to TouchProjector \([11]\)). An exemplary use is activating a projection on the handheld by tapping. Finally, the entire frustum (controlled by the handheld’s position and orientation) can manipulate a VP (e.g., to apply imaging filters to a projected photo).

IMPLEMENTATION

We developed a prototype framework to explore the characteristics we identified, reusing the same client-server architecture as in our work on Touch Projector \([11]\). Overall, our setup contains: (1) the handhelds (Apple iPhone 4), (2) software on all displays in the environment, and (3) a server that manages all connections between the handhelds and secondary displays. All connections use wireless LAN (or wired Ethernet if applicable). In contrast to Touch Projector, however, one display is the server.

As in Touch Projector, each handheld streams its live camera feed wirelessly to the centralized server. Using feature tracking and calculation (see next section), the server (1) detects the targeted display, and (2) calculates the spatial relationship between the handheld and this display. All client interactions (on the handheld and secondary displays) are routed through the server, as they may affect the behavior of a synchronized view.

Tracking Spatial Relationships to Displays

We intended VP for regular, unmodified displays in uninstrumented environments. Thus, we chose to rely solely on the handheld’s built-in camera (i.e., its live video feed) and the secondary display’s visual content, instead of external tracking hardware (e.g., \([25]\)). As stated earlier, most existing approaches for tracking spatial relationships between a mobile device and large display using live video use markers, either physically attached to the display or digitally superimposed on the screen’s content \([1,28]\). Unarguably, these approaches track well, but markers must be visible to the mobile camera at all times and thus clutter the display or its surrounding frame. To avoid this, we decided to use a markerless approach, similar to Herbert et al. \([18]\).

Detecting the target display. We use speeded-up robust features (SURF) \([4]\), which tries to find one or more templates within a given image using feature matching (see Figure 4). Note that the template does not need to be fully present in the image (i.e., parts of the template suffice, provided there are enough features). In practice, we find that 20% of a feature-rich screen is enough to produce at least 10–15 features, allowing for reliable detection. Templates are screenshots of all secondary displays, and the images in which to find the templates are the live video frames of all handhelds. If a template is found, SURF responds with the homography (i.e., transformation matrix) between both image planes.

The server attempts to find the screenshot of each display in every incoming video frame of a handheld. As VP can span multiple displays, the result may contain multiple displays and their corresponding transformation matrices. The server then uses each matrix with its display’s physical size and resolution to calculate the pose (based on Kato et al.’s algorithm \([24]\)), allowing for full 6 degrees of freedom tracking.

Allowing for dynamically changing screen content. Our system modifies the display’s visual content throughout the interaction. For example, even moving a projection changes the screenshot and the image features for later matching. Thus, having only a static screenshot of each display is insufficient and will quickly increase matching and calculation errors. Instead, our framework extends the original idea of matching a screenshot: Each display periodically takes screenshots, calculates the image features, and sends this information to the server. Thus, the server has the most recent screen content available for template matching.

Since wireless image transmission introduces additional delay (from both secondary displays and handhelds), synchronization is necessary. To accomplish this, we timestamp screenshots (and their calculated features), as well as video frames. Since the server has no knowledge if a received video frame is as recent as the screenshots from all displays, it stores screenshots and features of at least the past two seconds. When a new video frame arrives, the server picks the screenshot closest to that frame to minimize offset and calculation errors. Our system thus represents a novel real-time markerless tracking approach that supports continuous feedback and dynamic screen content.

Performance

Our current system uses an iPhone 4, 160×120 pixel color frames (pre-scaled on the phone), and Wi-Fi transmission to a MacBook Pro (2.4 GHz Core Duo, 4GB RAM) running Windows 7. It achieves ~10 frames per second (fps) for
image transmission and ~8 fps for tracking, with a delay of 100–130 ms. Accuracy depends mainly on the handheld’s distance to the display (i.e., on the display’s size in the image). Recent tests with an iPhone 4S and a more powerful server (3.4 GHz Intel i7) achieve the same speed with unscaled 320×240 pixel frames. While overall speed (slower transmission, but faster calculation) remains constant, accuracy increases due to the larger, better quality image. Faster client and server processors, along with better transmission technologies will produce further improvements.

Our implementation supports multiple simultaneous clients, whose number is limited by the server’s calculation speed (we find that the best trade-off is to calculate one image per processor core). Likewise, speed decreases as the number of secondary displays grows. Although these displays calculate features on their own before sending them to the server, the server must match each feature set to incoming frames. In addition, more displays increase the chance of two displays having very similar content (> 75% of a region in the phone’s viewfinder), which will cause calculation errors.

Limitations
Several drawbacks result from the restrictions of a consumer handheld and the computer-vision approach: (1) Feature-based tracking requires display content containing rich features [28]. Our experience shows that this is rarely an issue in practice, provided we avoid roughly uniformly colored backgrounds. (2) The captured region of a display must be unique. (3) Our current implementation captures up to 12 screenshots per second. If content changes more rapidly (e.g., a video with fast-paced scene changes), calculation errors are introduced. (4) The wide-angle lenses in today’s handhelds limit their operational range to < 10° on a standard 24” diagonal display. We expect that future improvements in server and handheld speed will transcend the last two limitations. However, other solutions must be found for the first two limitations.

VIRTUAL PROJECTION EXAMPLES
Our framework can be used to create synchronized projected views by aiming the handheld device at a display and long-pressing (~ 1 sec.) the touchscreen. Switching between views and navigating within them happens by aiming at a view and tapping, or by shaking the device to open the view manager showing all available views. The view manager also enables users to create a new projection (Figure 5a). Here (and in the video), we show example use cases that demonstrate the potential of VP. Table 1 classifies each case by the characteristics we discussed earlier. Note that we focused on creating compelling applications, instead of implementing all possible combinations of characteristics.

**Notepad**. A very simple, yet powerful, use case is to clone the handheld’s visible content. This resembles the core idea of mobile optical projection. However, decoupling the projection from the handheld allows for subsequent interactions. Figure 5b shows how users can create Post-it-like notes, writing messages and posting them on the large screen. Since projections are synchronized, users can change a note’s content after it is projected, which is immediately visible on both the handheld and secondary display.

**Photo Viewer**. Instead of cloning only visible content, views that do not fit on the small display can be extended. In this scenario, the entire content of the handheld (not just its visible portion) is cloned to the large display. The area that is currently visible on the handheld is shown on the projection with a thin white border (Figure 1a). Users can interact with the projection in two ways: Manipulating (i.e., panning or zooming) the handheld’s view updates the selection frame on the large screen, and changing the projection frustum determines the region shown on the handheld.

**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 handheld’s visible sub-region (Figure 5d). While conceptually similar, the web browser can be controlled on both displays: using touch input on the handheld, and mouse and keyboard on the secondary screen. When users click on links (and enter a new URL in the address bar), the website is updated synchronously on both devices.

**Video Player**. Users are able to project video onto the large display. However, the video is then no longer shown on the handheld. Instead, the handheld presents controls to manipulate the video (i.e., playback, seek, and volume). Thus, the handheld turns into a temporary remote control, linked to

<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</td>
<td>Linked</td>
<td>Decoupled</td>
<td>None</td>
<td>Indirect</td>
</tr>
<tr>
<td>Filters</td>
<td>Extended</td>
<td>Restricted</td>
<td>Continuous</td>
<td>Frustum-based/Direct</td>
</tr>
</tbody>
</table>

Table 1. VP characteristics for the example use cases.

Figure 5. VP examples (from left to right): (a) Creating new views, (b) notepad view, (c) video with remote controls, (d) web view.
the secondary display, as long as the VP remains on it (Figure 5c). Audio can be played on the secondary display’s speakers, the handheld (e.g., on its earphones), or both.

Maps. A small handheld display typically requires many panning and zooming operations within maps. Here, a larger section of a map is projected onto the secondary display, with the handheld showing a subregion in greater detail (Figure 1c). In contrast to previous extended view examples (Photo Viewer and Web Browser), the handheld is now a magic lens [7]—the mobile view is continuously updated in real-time by moving the handheld in front of the projected map. Zooming is achieved by moving the handheld closer to or farther from the display. This is an example of focus (handheld) plus context (secondary display) views [3].

Photo Filters. Users can select an image filter (e.g., color, grayscale, invert, and alpha) on the handheld and apply it to a photo on the secondary display. As in the magic lens Maps example, the handheld’s frustum defines the filter’s area, which is updated in real-time (Figure 1b). Filters are placed by long pressing and affect all VPs they overlap. Filters are composed by placing them above each other.

USER STUDY
Exploring the complete space of VP characteristics in a study would have been unmanageable, so we decided to evaluate the central underlying aspect: Controlling the handheld’s frustum on another display with different types of projection. To estimate how well VP worked for each, we measured speed and accuracy for a targeting task.

Independent Variables: Technique and Projection
We compared four different Techniques: two variants of VP and two controls (one absolute and one relative placement 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. We chose a second VP technique, VP (Thumb), to evaluate the impact of more fine-grained touch-based interaction on VP: 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. Lifting the finger places the projection. 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 with one finger or rotated/scaled using a two-finger pinch. A double-tap confirms placement of the projection. Touchpad is a relative input technique in which the entire handheld screen can be used to drag (single finger) or rotate/scale (pinch) the source. As with Minimap, a double-tap confirms placement.

We decided to test three different Projection types: Translate-Scale-Rotate (Figure 2b), Translate-Scale (Figure 2c) and Translate (Figure 2d). For each Projection, only the specified transformations are applied to the source.

Task
We used a targeting task inspired by existing Fitts’s Law tests (e.g., ISO 9241-9). However, we had a target quadrilateral instead of a distinct target point: We asked participants to try to align the projection as well as possible to the target, while being as fast as possible. Boring et al. introduced the term docking offset for error created in such targeting scenarios, defining it as “percentage of the object located outside the target area” [11]. We computed as error measures docking offset, distance between centroids of the projection and target, and scale and orientation differences.

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

All projection types 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 projection type, we had four blocks. A block contained four trials, each of which had a different target position and shape or orientation respectively. Transformations were randomized within a block and the first block was used for training and discarded from the analysis. We thus had a total of 4 (Technique) × 3 (Projection) × 3 (Block) × 4 (Trial) = 144 trials. Dependent variables were speed and accuracy.

We formulated three hypotheses: (H1) For complex projections (Translate-Scale-Rotate), the VP techniques would be faster than the control techniques, due to fewer required movements. (H2) In terms of coarse positioning (time to approach the target), both VP techniques would outperform the two control techniques. (H3) VP (Thumb) would result in more accurate placement than 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. For the VP conditions, we used our feature-based tracking approach, with a display-filling, feature-rich background image to assist tracking. We logged all interaction with the handheld (all touch points) and all tracking results for each frame.

We recruited a total of 12 participants (3 female, 11 right-handed, ages 22–30, \( \bar{x} = 26.8 \) years) from our institution. Each participant took about 45 minutes to complete the study including post hoc 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.

Before the main analysis, we performed a 4 × 3 × 3 (Technique × Projection × Block) within-subjects ANOVA and found a significant effect for Block (F_{2,22} = 5.876, p < 0.009) on task time, indicating the presence of a learning effect. Subsequent pairwise comparison showed that participants were significantly slower in Block 1 compared to the other two for all techniques and variations except Translate (Block 0 had been removed beforehand). As well, for Translate, our techniques showed lower task times. Thus, we also discarded Block 1 for all following analyses.

Task Completion Time
We aggregated task completion times across blocks and performed a 4 × 3 (Technique × Projection) within-subjects ANOVA. We found significant main effects for Technique (F_{3,33} = 9.657, p < 0.001, only Minimap differed from the two VP conditions with p < 0.003) and Projection (F_{2,22} = 69.178, p < 0.001, only Translate differed from the other projection types with p < 0.001). Furthermore, we found a significant interaction between Technique and Projection (F_{1,66} = 9.309, p < 0.001).

The reason for this interaction can be seen in Figure 6 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), Minimap and Touchpad task times 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_{3,33} = 15.870, p < 0.001, Minimap differed from all other interfaces with p < 0.01) and Projection (F_{2,22} = 13.277, p < 0.001, Translate differed from the other two Projections with p < 0.016). We further found an interaction between Technique and Projection (F_{3,438,66} = 2.908, p < 0.041).

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 a cost of slightly higher offsets, we additionally evaluated “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). 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 × 3 × 3 (Technique × Projection × Offset) within-subjects ANOVA. We found a significant main effect for all independent variables (all p < 0.001). We also found significant interactions for all combinations (all p < 0.001). From Figure 7, we see that time until an Offset is reached increases with Projection complexity for Minimap and Touchpad. Simple Projections did not influence homing speed of different techniques for all Offsets (all p > 0.05). In contrast, post hoc multiple means comparisons (against corrected α = 0.0083) of medium-complex Projections (Translate-Scale) showed VP (Pure) outperformed both Minimap and Touchpad (all p < 0.002), whilst VP (Thumb) only showed a significant difference compared to Minimap for 25% and 20% (all p < 0.002). For further increase in complexity, post hoc tests revealed both VPs are faster than Minimap (all p < 0.001) and Touchpad (all p < 0.004) for all Offsets.
Qualitative Feedback

Participants were asked for their opinion after each technique. VP Pure was most commonly described as ‘intuitive’ or ‘easy to learn/use’ (S of 12). While similar comments were made about VP Thumb, its distinctive feature of using the thumb was only lauded by three. For both VP techniques, participants mainly criticized the inaccurate tracking and difficult precise interaction (VP Pure: 6, VP Thumb: 7). Opinions on Touchpad were less consistent, but participants liked it in general, for example because it relied on a ‘known gesture’ (3), even though some found it ‘too slow’ (3). Finally, Minimap elicited mixed comments. While using it for a Translate projection was considered easy (3) and the visual feedback was liked (3), half the participants criticized it for difficulties in rotating/scrolling (6) or found the visual feedback too small (4). Most complaints were grounded in the ‘fat-finger’ problem and subsequent difficulties in accurate placement and confirmation. One participant joked that Minimap was ‘perfect for gnomes.’

DISCUSSION

Our quantitative results show that for complex Projections, both VP techniques significantly outperform Minimap and Touchpad, supporting H1. Overall, for complex Projections, VP (Pure) / (Thumb) 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 VPSs significantly outperformed the other two techniques for complex transformations, as we had expected (H2). Our assumption is further supported by both VPSs outperforming Minimap for Translate-Scale at all Offsets. We also assumed that there would be not much difference among the techniques when the projection type allows only simple transformations (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).

In overall docking offset, the VPs perform slightly worse (5.8–7.1%) than the other techniques (2.5–4.7%). Most surprisingly, 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, both of which 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 longer task times. While this may improve 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 theme of decoupling the projection from the handheld device allows using the frustum for interactive filtering or selecting display subregions 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, which we plan to explore 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, which we would like to explore. As handhelds stay synchronized to VPs, interaction could even happen remotely. Additionally, interacting with a handheld device brings with it implicit user identification. Another direction to explore is 6DOF device tracking. While we have only projected onto 2D surfaces, we could create interaction techniques that use the full 3D frustum to place 3D objects, cut through 3D scenes, or control avatars.

ACKNOWLEDGEMENTS

We would like to thank our reviewers and Raphael Wimmer for their insightful comments, our participants and Doris Hausen for their time, and Andreas Butz for his support. This research is partially funded by the State of Bavaria, the iCORE/NSERC/SMART Chair in Interactive Technologies, Alberta Innovates Technology Futures, NSERC, and SMART Technologies Inc.

REFERENCES


