Open Project: A Lightweight Framework for Remote Sharing of Mobile Applications

Matei Negulescu[*]
University of British Columbia
mnegules@cs.ubc.ca

Yang Li
Google Research
yangli@acm.org

ABSTRACT
The form factor of mobile devices remains small while their computing power grows at an accelerated rate. Prior work has explored expanding the output space by leveraging free displays in the environment. However, existing solutions often do not scale. In this paper we discuss Open Project, an end-to-end framework that allows a user to "project" a native mobile application onto a display using a phone camera, leveraging interaction spaces ranging from a PC monitor to a public wall-sized display. Any display becomes projectable instantaneously by simply accessing the lightweight Open Project server via a web browser. By distributing computation load onto each projecting mobile device, our framework easily scales for hosting many projection sessions and devices simultaneously. Our performance experiments and user studies indicated that Open Project supported a variety of useful collaborative, sharing scenarios and performed reliably in diverse settings.

Author Keywords
Remote sharing; protocol and architecture design; mobile interaction; computer vision; projection-based interaction.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: User Interfaces: Input Devices and Strategies, Interaction Styles.

INTRODUCTION
Mobile devices grow rapidly in computational power and ubiquity. However, their small form factor, though highly portable, offers a limited interaction bandwidth. In particular, touchscreens, as the major modern input and output medium, suffer from finger occlusion and restricted view areas in spite of their high resolutions (e.g., an Apple iPhone 4s affords a 3.5" touchscreen with a 960x640 resolution). They are awkward to use especially when more than one user wants to interact with the device, e.g., viewing pictures together or playing a multiplayer game.

To address these problems, handheld projectors have shown promise in enlarging the output area of a mobile application [4,5,8,9,15]. However, handheld projectors are often additional hardware to carry and also require special sensors (e.g., a depth camera) for detecting user input in the physical space [8,24]. In addition, projector-based solutions suffer from physical limitations such as hand tremor which makes stabilizing a projection during interaction difficult.

To leverage the intuitiveness of projector-based interaction and overcome the above issues, recent work has explored mobile content sharing using software-based projection (e.g. [3,7,10,16]). For example, Deep Shot allows a user to "post" mobile content onto a remote registered display using a built-in phone camera [10]. More recently, Baur et al.'s Virtual Projection allowed users to share a mobile application by providing continuous feedback on the target display to simulate a projection effect [3]. However, these previous solutions face several challenges.

First, previous systems generally consider predefined target displays for sharing. To set up a target display, the user would need to install the required software component on the display and also register the display before using it. In contrast, we focus on ephemeral sharing scenarios on a variety of displays, especially in a collaborative or public setting, which desire little deployment overhead.

Second, previous solutions conduct computationally intensive tasks such as computer vision on the server or on the target display. This architectural design avoids complex communication protocols and vision processing on low-end mobile hardware. However, it does not scale for hosting many target displays and simultaneous sharing sessions.

Last, previous systems that employ camera-based sharing techniques detect vision features of the screenshot image on the remote display. Although this solution is more seamlessly integrated with the interaction environment, it does not scale for realistic scenarios that often involve sharing from varying distances and angles to remote displays in different lighting conditions.

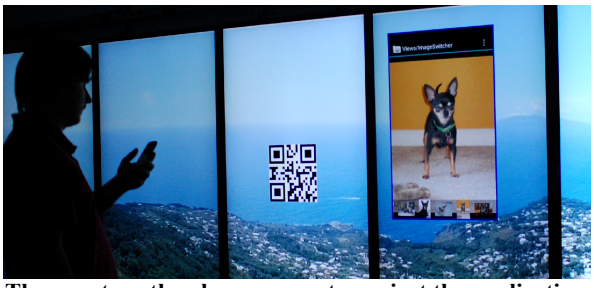
To address these issues, we developed Open Project, a lightweight, highly scalable framework that allows users to leverage the input and output capabilities of other available devices, such as a large display, for interacting with mobile applications. Similar to prior work, it allows a user to share a running mobile application onto a target display by "projecting" it via a mobile phone camera (see Figure 1). However, it goes beyond prior work in several ways.

UIST'13, October 8–11, 2013, St. Andrews, United Kingdom.
ACM 978-1-4503-2268-3/13/10.
http://dx.doi.org/10.1145/2501988.2502030

[*] This work was done while the author was an intern at Google Research.

a) A user starts by aiming the phone camera at the barcode that uniquely identifies the display.

b) The barcode, once identified, changes to a checkerboard marker that continuously shifts as the user orients the phone toward a target position, as if it were projected from the phone.



c) The user taps the phone screen to project the application at the checkerboard position, and interacts with the application via either the phone or its projection on the display. The barcode reappears for other users to project their applications.

Figure 1. A user projects a foreground application running on the phone (e.g., Photo Gallery) onto a wall-size display for a larger view of it. To do so, the user activates Open Project on the phone that (a) identifies a specific display, (b) selects a target position and size, and (c) sharing the application.

First, we developed a web-based framework for supporting multiple users simultaneously sharing content on one or many displays. With the easily-accessible nature of web services, any display can become *immediately projectable* once it accesses the Open Project server in its web browser and can support any number of sharing sessions—deployment is as simple as opening a webpage. A mobile device is paired automatically with a target display once the user identifies the display's 2D barcode, requiring no authentication or pre-registration.

Second, we designed a *decentralized architecture* and communication protocol for scalable mobile sharing. All intensive computation—mainly realizing camera-based projection and executing applications—is performed locally on each individual mobile phone; when sharing, a video feed of the application is transferred to the remote display. This enables our framework to easily accommodate an arbitrary number of projection devices and target displays.

Third, we created *a robust "camera projection" algorithm* for creating a projection effect for users to select sharing positions and dimensions on a target display. The algorithm uses a dedicated checkerboard pattern, instead of relying on screenshots, such that it performs reliably at great ranges, varying lighting conditions and angles. Our algorithm functions when the target screen is blank or has a background image. It performs robustly against on or off-screen distracters. A thorough evaluation of the projection algorithm and the system as a whole provided solid findings for how this kind of system performs in a realistic scenario.

RELATED WORK

In addition to projecting mobile screens onto a physical surface (e.g. [4,5,8,9,15]), prior work investigated mobile sharing to remote displays. One solution of sharing is to send user data or application states to the target machine, e.g., [7,10,12,14,16,19]. The other category of sharing is to stream live copies of UIs—images—to the target machine, and redirect user input to the originating device, e.g., [1,3,7,21,23]. Although transferring files or application states allows better performance on the target machine, it requires the target machine to handle the data or execute the application which introduces security concerns when using a shared or public display—our focus scenario. As a result, we chose to share live screen images but keep data and computation on the mobile device.

XICE [1] allows developers to share live views of mobile apps onto other displays using a custom UI rendering framework. While this approach allows less latency, it is incompatible with existing mobile UI frameworks (e.g., on Android or iPhone). Modifying mobile apps to leverage the framework requires significant developer effort. Open Project allows developers to enable remote sharing of any mobile application with fewer than thirty lines of Java code.

An important step in sharing is to specify a target on a remote display. Previously, using a built-in phone camera for direct pointing has shown promise for WYSIWYG selection of distant targets [2,3,6,10,18]. For example, past work has used a grid of static markers as anchors on the remote screen to estimate where the phone is pointing [2,18]. Open Project uses a single marker, which provides continuous feedback and visual synchronization for targeting.

Open Project is closely related to Virtual Projection [3], which used the projection metaphor to share a smartphone's display via its camera. However, the two techniques are different both in focus and in design. Open Project is aimed at enabling impromptu sharing on arbitrary displays with minimal deployment effort. It is particularly designed for remote sharing on public, shared displays where authentication or pre-registration should be avoided. In particular, Open Project employs a web-based architecture

Figure 2. Multiple users simultaneously projecting their mobile applications (left: Maps and right: Photo Gallery).

to allow lightweight instantaneous deployment and easy access to the projection service.

While Virtual Projection centralizes computation to the server and the target display, Open Project employs a distributed architecture that keeps most of the computation on each individual mobile device; this approach can easily scale for many simultaneous projection devices. In addition, instead of relying on visual features of the screenshot of a target display (e.g., [3,10]), Open Project employs a QR code and a camera-propelled marker to indicate a specific position on a remote display. The marker allows reliable and efficient tracking on mobile hardware, and performs at great ranges with varying lighting conditions and angles or when the target display has no visual features.

Using camera frames to estimate motion is widely used in computer vision. Rather than relying on a static scene (e.g., [22]), Open Project uses detected motion to propel a tracking target to simulate the projection effect. We designed a visual synchronization protocol to coordinate two asynchronous but dependent movement processes.

REMOTE SHARING WITH OPEN PROJECT

A handheld projector allows a user to project content at a specific location and size on a large, physical surface. Open Project leverages the projection metaphor so that users can easily share an arbitrary, native mobile application at a desired position on a target display such as the wall-sized display in Figure 1. Instead of relying on the screenshot of the target display (e.g. [3,10]), we use the phone's camera to drive a marker on the remote display to locate where to project—simulating physical projection. This approach functions even when the screen has no visual features in its background—a blank screen. Once the user identifies a projection region, we send screen pixels to be rendered on the remote display instead of the application itself, which allows the sharing of an arbitrary, native application. As the focus of our system is scalability and multi-user collaborative scenarios, Open Project also implements a visual handshake protocol to register users with any display showing a browser pointed at the Open Project server.

To better describe our interaction flow, consider Angela and Ben, two users at a mall. Angela wants to show Ben her vacation pictures on her smartphone. Instead of giving Ben her phone, Angela decides to use the nearby public display to browse these pictures. To do so, Angela enables Open Project in her Photo Gallery application, which starts a camera view for her to capture the marker. Angela identifies the public display by capturing its QR code (see Figure 1a). Once the QR code is identified, it changes into a checkerboard marker. The blue rectangle centered on the marker represents the target region into which the mobile application will be projected into (see Figures 1b) and moves according to where Angela points her phone's camera. To change the target projection size, Angela slides her finger in a circular motion on the touchscreen (as in [20]): moving counterclockwise to enlarge and clockwise to shrink the projection region.

Once satisfied with the location and size of the projection on the display, Angela taps on the phone's touchscreen to affix the projection, which exits the camera view and returns to the Photo Gallery. Meanwhile, the Photo Gallery is shared at the size and location indicated by the projection region on the remote display. Further movement of the phone will not affect the projection. At this point, the QR code reappears at the center of the display ready for other users' projection (see Figure 1c).

The users can now interact with Photo Gallery through both the application running on the phone and its remote projection. As Angela flicks through the pictures by swiping on her phone, the changes take effect in realtime in the mirrored remote representation. When Ben wants to select an image, he swipes the projection directly on the large touch-enabled display. Ben's remote touch input is relayed to Angela's phone to select the image. Should Angela want to maintain control over her application, she can disable remote input through a menu option.

Meanwhile, a third user captures the QR code using Open Project and shares his Maps application. All users occupy different parts of the display to interact with their applications; all computation occurs on their respective mobile devices (Figure 2). Open Project is able to host multiple concurrent projections on a display and the display space is allocated in a First-Come-First-Serve manner. A projection owns its region on the display—no other projections can be placed on top of it—until the user stops the projection from the phone.

THE OPEN PROJECT FRAMEWORK

We designed the framework to satisfy several important goals. First, the framework should support common collaborative and sharing scenarios for mobile users. Second, the system should be usable for off-the-shelf mobile devices, without requiring additional hardware or sensors, should require minimal computation and infrastructure, and should be robust in various settings. Third, the system should be easy to deploy and incorporate. To minimize deployment costs, Open Project requires neither the installation of a custom kernel on mobile devices nor additional software on the remote display.

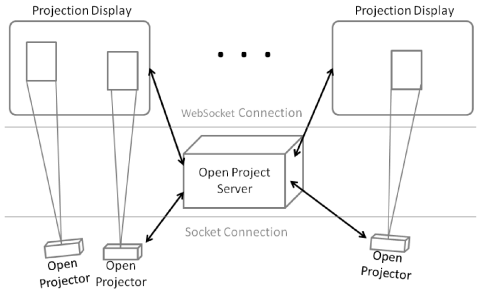


Figure 3. The Open Project runtime architecture.

Instead, Open Project allows developers to project their mobile content by simply adding an Android library while any remote display is share-capable by pointing a browser at our URL.

The Runtime Architecture

The runtime architecture of the Open Project framework consists of three components (see Figure 3): 1) Open Project Server—a web service running on a centralized server for managing projection sessions, 2) Open Projector—a library running on the smartphone for projecting the mobile application, and 3) Projection Display—a web application running in a browser on the display's computer for serving one or multiple projections.

Open Project allows an arbitrary display to become projectable. The owner of the display simply starts a browser on the display's computer and accesses the Open Project Server on the web, i.e., via an HTTP request. This runs Projection Display in the browser. Projection Display starts with a QR code shown at the center of the screen that encodes an ID for identifying the display. Multiple displays can make requests to Open Project Server at the same time and each will be assigned a unique ID.

Note that this lightweight deployment can be as simple as clicking on a bookmark that points to the Open Project Server. The owner of the display can also choose to have a background image or a blank screen. For example, on a public display in a train station, a static background image or dynamic graphics (such as ads) can be displayed as usual, and a projection will be rendered on top of it.

In front of a display, a user activates Open Projector on the phone, e.g., via a menu, while the application of interest is running in the foreground. There are three steps to project a mobile application: *identifying a target display* via the QR code, *adjusting the projection region* on the display, and *sharing the phone screen* on the selected projection region.

Identifying a Target Display

Open Projector starts by invoking the camera functionality on the phone to search for a valid QR code in the scene. It brings up a camera view on the phone for the user to aim the camera at the QR code on the target display. Once Open Projector detects a QR code, it decodes the display ID from the QR code and requests the Open Project Server to create a projection session. The server then notifies both the Projection Display, identified by the decoded ID, and the Open Projector with a session ID. Thereafter, the Projection Display and Open Projector communicate with each other through the Open Project Server using the assigned session ID, which allows messages to be correctly routed when multiple sessions are active (see Figures 2 & 3). Once Projection Display receives a new session ID, it responds by turning the QR code into a checkerboard marker for the user to adjust the projection region.

Adjusting the Projection Region

Once the Open Projector sees the checkerboard marker, it calculates how much the detected marker deviates from the center of the camera view. It then sends the Projection Display the deviation as the amount of adjustment that the marker has to make on the display so that the marker can return to the center of the camera view. As the user moves or orients the phone towards a target position on the display, the marker adjustment is continuously sent to the Projection Display, and the position change of the marker on the display is fed back to the phone visually through the camera. We will elaborate on the details in the next section.

As mentioned earlier, a user can slide her finger on the phone's touchscreen in a circular motion to enlarge or shrink a projection denoted by the blue bounding box. Note that the size and orientation of the marker remain constant. A user may rotate the phone to switch between landscape and the portrait orientations, which will change the orientation of the projection region accordingly.

Sharing the phone screen

Once a user taps the camera view on the phone, the camera view exits and the projection region is affixed. The system now enters the sharing stage. In this stage, the phone's position and orientation become irrelevant to the projection on the display. As the Open Projector has returned control to the application (from the camera view), a user can now interact with the application. Meanwhile, the Open Projector periodically polls the screen image of the application, compresses and sends the image to the display if the screen has been updated. The Projection Display receives the screenshot of the application, scales and renders it according to the determined projection region.

A user can interact with the mobile application on the phone as usual and the UI changes will be reflected on both the local phone screen and the remote projection. In addition, Open Project allows users to directly operate the application via the projection if the display supports user input. When a pointing event occurs, such as a mouse or touch event, the Projection Display verifies if it is within the boundary of an affixed projection, and if so, relays the event to the paired Open Projector using its session ID. This allows the Projection Display to dispatch pointing events to appropriate projections when multiple projections exist on the same display. Additionally, the Projection Display also acts as a window manager and ensures that projections do not overlap, nor are moved off screen.

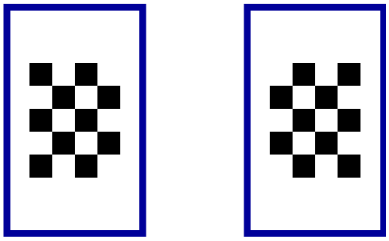


Figure 4. The two checkerboard patterns used for positioning. One is the negation of the other in colors.

The Open Projector receives input events from its projection on the display, and re-dispatches the events to the UI of the application that is being projected. The application responds to the events as if these events were originated locally from the phone, and the UI changes will be displayed on both sides as discussed earlier.

Algorithms for "Camera Projection"

An important step in Open Project is to position a target projection region on a remote display. Open Project emulates a handheld projector through "camera projection", a technique that allows a user to physically target their desired location on the remote display using a built-in phone camera instead of a real projector. To scale for many simultaneous projecting mobile devices and displays, our architecture distributes the computation required for camera projection onto each mobile device.

The fundamental intuition behind the camera projection algorithm is that a designated pattern on the display (e.g., a checkerboard in our current design), as the projection from the camera, should stay at the center of the camera view. When a user moves or orients the phone camera towards a new location on the display, the pattern will deviate from the center of the camera view. Based on the amount of observed deviation, we can calculate how much the target pattern should be moved on the display so that it is centered in the camera view again.

Designing a Target Pattern

An important issue for the algorithm to work is to design an appropriate target pattern that is easy and reliable to detect by a phone camera. A common solution has been detecting features in a screenshot image (e.g. [3,13]). However, such a solution is dependent upon having an appropriately feature-rich screenshot, is constrained under different viewing conditions and at larger distances, and can become complicated when the screen is highly dynamic (e.g., when other simultaneous projections are constantly updating).

Alternatively, checkerboard patterns have been extensively used in computer vision for camera calibration (e.g. [17]). Checkerboards are distinctive markers that are detectable in many lighting conditions, amongst complex distractors, and at various angles. Additionally, such a marker can be dynamically scaled depending on the size and resolution of the display to enable interaction at large distances. Finally, reliable detection of checkerboard markers can be achieved

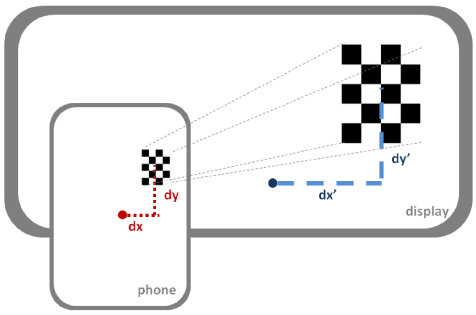


Figure 5. Transforming the offset in camera coordinates, *dx* and *dy*, to an offset in display coordinates, *dx'* and *dy'*.

efficiently on low-powered devices such as smartphones, making it ideal for Open Project.

In our specific design, we use a 5-by-4 checkerboard pattern (see Figure 4) that is resilient of false positives and gives more than enough inner intersection points for calculating a homography matrix. Furthermore, the asymmetric design of the checkerboard helps determine the orientation of the camera with relation to the display.

Detecting a Checkerboard Pattern

Our checkerboard detection algorithm is based on OpenCV, a popular computer vision open source library [25]. In particular, OpenCV provides a set of functions for camera calibration by estimating camera parameters from a series of frames containing a checkerboard pattern. While this algorithm is effective for one-time camera calibration, it is focused on accuracy over speed by continuously iterating its search for quads whose intersections form the inner points of a checkerboard. The algorithm is less suitable when realtime tracking is needed on a smartphone processor. Thus, we tailored the OpenCV algorithm in two ways. First, we eliminate quads quickly by aggressively thresholding based on size and proximity. Second, we eliminate the original algorithm's iterative approach for finding intersection points by expanding quads. A pilot study showed that iterating the search has diminishing returns that do not justify the time cost.

Transforming Deviation from Camera to Display Coordinates

The output of the checkerboard detection algorithm is a set of twelve internal intersection points in camera coordinates. Let *dx* and *dy* be the offsets of the centroid of these points from the center of the camera view on the horizontal and vertical axis (see Figure 5). We need to transform these offsets to display coordinates to find how much the checkerboard must be moved on the display (Equation 1).

$$w\begin{bmatrix} dx' & dy' & 1 \end{bmatrix}^T = H\begin{bmatrix} dx & dy & 1 \end{bmatrix}^T \quad (1)$$

where H is a 3-by-3 projective transformation, i.e., a homography matrix, from camera to display coordinates.

Typically, a homography H can be calculated using a RANSAC approach [11], which iteratively finds the best mapping and the transformation matrix between two sets of points, in our case the 12 checkerboard points detected in

camera coordinates and on the display. However, this process can be simplified for our use. We can determine the correspondences between the two sets of points, by sorting the detected checkerboard points based on the phone's orientation (detected via built-in accelerometers) and their order in the horizontal and vertical axis. Once the correspondences are known, we can find the homography H by calculating the least square solution of a linear system (see Equation 2), which is much faster than RANSAC.

$$T = H\,O \tag{2}$$

where O is a 3-by-12 matrix whose columns are each the homogeneous coordinates of a checkerboard point observed in camera coordinates. T captures the coordinates of the 12 checkerboard points on the display. In fact, we do not need to know the absolute position of the checkerboard points on the display due to the following deduction. Each point can be represented as a relative coordinate, $T_{relative}$, to the checkerboard center, T_{center}. We do the same for the observed checkerboard points O. Thus,

$$T_{center} + T_{relative} = H\,(\,O_{center} + O_{relative}\,) \tag{3}$$

Because $T_{center} = H\,O_{center}$ is redundant for the calculation, we acquire Equation 4.

$$T_{relative} = H\,O_{relative} \tag{4}$$

However, $T_{relative}$ still requires us to know the physical size of the checkerboard on the display which is constant during projection. One solution is to use a predetermined fixed checkerboard size for all displays. However, this is inappropriate for diverse projection constraints, e.g., a large checkerboard might be used for a wall-size display to allow long-range projection while a small one can suffice on a PC monitor. A second solution is to let Projection Display pass the checkerboard size being used to the Projector on the phone at runtime—an extra step. Instead, we can eliminate this requirement by further deduction. We can take the size factor, S, out of $T_{relative}$:

$$S\,T_{unit} = H\,O_{relative} \tag{5}$$

where S is a 3x3 scaling transformation matrix and T_{unit} is the set of checkerboard points at a unit scale, independent of the actual physical size and position of the checkerboard. We then acquire:

$$T_{unit} = S^{-1}\,H\,O_{relative} \tag{6}$$

We can calculate $S^{-1}\,H$ as a whole, but not individually. However, we can use the expression as a whole, based on Equation 1, to transform dx and dy. This gives us the amount of adjustment needed at the unit scale. Once the Projection Display—which knows the actual size of the checkerboard S—receives this adjustment it can recover dx' and dy' by applying S to scale up the unit scale adjustment.

Synchronizing by Alternating Checkerboard Patterns

Generating updates on the phone and updating the checkerboard on the display are two asynchronous but dependent processes. An update is relative and applied to the marker's current position; the updated position is then used by the phone to generate the next update. As a result, when an update is sent to the Projection Display over the network, the phone has to wait for it to be processed by the Projection Display before analyzing a new frame and generating another update. This requires the two processes to be synchronized so that only one of them runs at a time.

To determine if an update has been processed, i.e., the checkerboard has been moved, we cannot rely on the checkerboard's deviation change observed through the camera as the phone is in constant motion even when not intended by the user. Thus, the Display must indicate that an update has been processed explicitly. Though it is possible to send this message to the phone over the network, it complicates the protocol and increases latency.

Instead, we use a visual synchronization mechanism by alternating between two distinct checkerboard patterns that share the same dimension (see Figure 4). When the display moves the checkerboard in response to an update request, it flips the checkerboard pattern to visually acknowledge. Once the phone detects the checkerboard switch, it starts to process a new frame and sends a new update.

Generating an update only when a checkerboard pattern flips synchronizes the processes well. However, it is possible that, before an update is processed or received by the display, the phone is already in a new position or orientation due to system latency and fast motion of the user's hand. To maximally follow the user motion, we choose to constantly generate updates, without waiting for the pattern to switch, but with an important enhancement. Each update is sent with the type of the pattern that it applies to, i.e., either type 1 or 2 (Figure 4). When an update reaches the display, the display checks its pattern type, applies the one that is consistent with the checkerboard pattern being shown and at the same time flips the pattern that will discard subsequent inconsistent updates. This ensures that a correct update is used for the current checkerboard position and is based on the assumption that in general an update would not take too long, i.e., longer than two pattern switches, to arrive at the display. This approach allows a later update to override an earlier unprocessed one, and leverages nondeterministic ordering of messages over the network (e.g. a late update can arrive early).

IMPLEMENTATION

We implemented Open Projector as a library based on the Android 4.0 platform. We employed the Zxing library [26] to detect the QR marker, and customized the OpenCV library to detect the checkerboard marker. Most of the computation for processing a frame and generating updates is written in C++ and performed natively to improve performance. It is then integrated via Android NDK with the rest of the library written in Java. Developers can enable Open Project for their Android application by using the Java API of the library (see Figure 7).

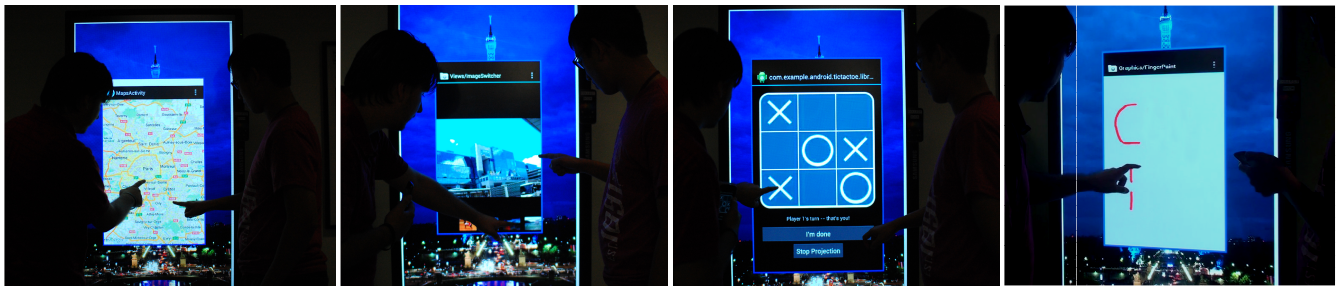


Figure 6. Example applications shared with Open Project: a) Maps, b) Photo Gallery, c) TicTacToe, and d) a drawing pad.

For sharing an application, the library grabs the drawing cache of the application at an empirically determined rate of 10fps to avoid overloading the phone processor. A new screen bitmap is compressed as a JPEG and sent over the network as an encoded Base64 byte stream. User input such as touch or mouse events from the remote display is converted as an Android TouchEvent and dispatched to the application's UI (view) hierarchy. The library communicates with the Open Project Server through a regular socket connection, which can be implemented with Web Sockets in the future.

The Projection Display is implemented as a web application and is automatically deployed on a target display when the browser points to the Open Project Server. We use a full screen HTML5 canvas in the browser for rendering projections and listening for user events. The web application is connected with the server via Web Sockets. Our server is implemented based on Tomcat, an open source web server [27]. The server merely routes messages and image data from mobile devices to screens. With the exception of decoding messages, it does not perform any additional computation, improving scalability. Web applications require no deployment and can easily respond to simultaneous requests from many different users–in this case, mobile devices. Finally, as all the computation is performed efficiently on the individual mobile device, the Open Project Server and the Projection Display need only limited hardware infrastructure for wide deployment.

```
// Instantiate a VirtualProjection instance
VirtualProjection projector = new OpenProject(activity);
// Start a projection
projector.startProjection (new OpenProjectCallback() {
  public void onSharingStarted () {
    // The user has selected a projection region and the sharing
    // stage is started.
  }
  public void onCanceled () {
    // The projection is canceled by the user.
    // before the sharing is started
  }
  public void onFinished () {
    // The sharing is ended by the user.
  }
  public void onFailed () {
    // VirtualProjection failed to find a target display region.
  }
 }
});
```

Figure 7. An example code snippet for enabling Open Project in an application.

ENABLING A MOBILE APPLICATION FOR PROJECTION

A developer can easily hook their mobile application into the Open Project framework. Although our current implementation is based on the Android platform, the API framework can be generalized for other platforms.

To add Open Project to an application, a developer instantiates the *OpenProject* class by passing it a reference to the current activity—a screen on the Android platform (see Figure 7). Open Projector will extract the root of the UI hierarchy for capturing the UI rendering cache and dispatching events from the remote display. A developer can select how to start projecting e.g., through a menu item. Remote input can be disabled on demand through a function call to maintain user control over the application.

To start a projection, a developer calls *startProjection* with *OpenProjectCallback* which allows the developer to handle a status change in the projection process. These callbacks give the developer the option to update the UI or signal the user, e.g., beeping when the sharing stage is started.

To demonstrate our API's ease of use, we added Open Project support to several sample applications provided by the Android SDK (see Figure 6): a Maps application, a Photo Gallery, a TicTacToe game, and a Drawing application. Based on these examples, we estimate that developers can add Open Project to their Android applications with less than 30 lines of Java code.

EVALUATIONS

In addition to demonstrating the usefulness of Open Project in a set of sample mobile applications, we evaluated Open Project in two ways. We first investigated the performance of our checkerboard updating algorithms. We then performed a user study to gauge user reactions to the tool.

Performance Experiments

We intended to find out, under different conditions, the effective range of the checkerboard marker detection and how fast we can update its position.

Apparatus

We used a Thinkpad T420s as our server (running Ubuntu Linux 10.04), a Galaxy Nexus smartphone for running Open Projector (with 11.8cm diagonal size at a resolution of 1280x720), and a PC connected to five Samsung displays (each is 68x120 cm) tiled by Windows 7 (see Figure 1). The PC showed a fullscreen Chrome browser running Projection

Display. Each display had a resolution of 900x1600, totaling 4500x1600 over five screens. The width of the two bezels between adjacent displays is 5cm. All devices were connected to a Belkin N750DB router's Wi-Fi network.

Experimental Design & Procedures

To find the effective range of our checkerboard detection and the time duration for updating the checkerboard position, we evaluated the system at various distances, angles, background distractors, and lighting (see Figure 8).

The major factors for the checkerboard detection include the checkerboard marker's size, the viewing distance and angle from the phone camera. The angle is critical due not only to perspective distortion but also foreshortening. Therefore, we tested our system for three angles: 90°, 60° and 30°. Note that at an acute angle of 30° we consider a range that is closer to the display (140-240cm) to take into account the significant perspective foreshortening. For each angle, we used six distances that cover the available space in the laboratory room. We used a fixed checkerboard marker—a 5x4 array of 3cm black and white squares. The small marker size was selected so as to detect the threshold distance within our small testing environment.

We also took into account two common factors for a computer-vision system: background distracters and lighting. For distracters, we tested three representative conditions: the marker is shown on a white screen, the marker is displayed on top of a full-sized background image, and the marker is surrounded by three distracter projections (a Maps app, a Contacts app, and a YouTube app). Two lighting conditions were tested: on and off.

Measures

In each condition, we mounted the phone such that the center of its camera view sat at a height of 115cm, perpendicular to the ground in portrait mode, with its center aligned with that of the checkerboard. The checkerboard marker was fixed on the display to maintain the designated spatial configuration. We then calculate two measures by processing 200 consecutive camera frames. We define the *detection rate* as the percentage of frames in which the algorithm finds the marker. This reflects how reliably our algorithm can detect the marker. We also measure the *update time* by looking at the duration between two adjacent checkerboard pattern switches detected by the algorithm—an update cycle. The update time includes the frame processing time, and other time costs such as the time to update the camera view, transmit an update over the network and render the new pattern on the display. Note that the update and detection rate are two correlated measures, i.e., when the detection rate is low, it takes a longer time to generate an update, and vice versa.

Results

We summarize Open Project's detection rate and update time performance under different conditions in Figure 8. When the update time is above 500ms, our algorithm is too slow to catch up to a regular user's motion, and we mark

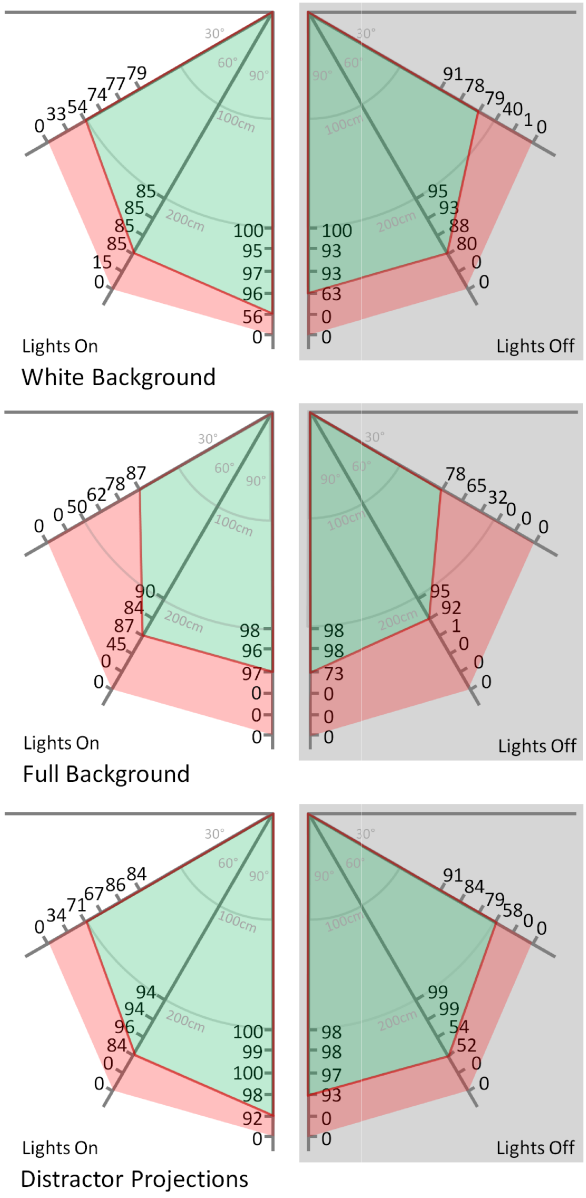


Figure 8. The detection rate (%) and update times in different conditions (green represents less than 500ms update time). Distances at 90° and 60° range from 200cm to 300cm. Distances at 30° range from 140cm to 240cm. They all have even increments of 20cm.

such a condition red in Figure 8. Otherwise, we mark it green. The checkerboard detection algorithm is most reliable when the lights are on, when there is less distraction at more straightforward angles. For example, with our very small marker size at 90° or 60°, a user can comfortably stay at a distance of 280cm and experience update times well below 500ms. We expect distance to scale linearly with the physical size of the rendered checkerboard marker which can be dynamically modified.

User Studies

We intended to find out 1) if a user can use our camera projection technique to adjust a target projection region on

the display, and 2) how mobile users react to our system as a whole. We recruited 10 participants (2 females, 8 males, with a mean age of 22.4) from an IT company, all highly experienced smartphone users. Only one participant had experience with handheld physical projection systems.

Experimental Design

To answer these questions, our study consisted of two sessions. First, a participant moved and resized the projection region on the five-display setup used in our performance experiment. After a warm-up session, a participant starts the task by enabling Open Project on a test application. Starting from the center of the display, the participant must move the projection region to a given location, randomly selected from two categories: those nearby, or at most 68cm away; and those far away, or greater than 170cm. The initial projection dimensions are 22x37cm. A participant needs to scale the projection region up to one of two scales: 31x52cm, and 44x74cm (i.e. 1.4 times or twice the original dimensions). We marked each target location and size on the display. Participants confirm the final projection region and complete a trial by clicking on a button on the smartphone. Participants performed the task 12 times for each setting.

In the second session, participants were asked to project three Open Project-enabled applications onto a 68x120cm touch-sensitive display running at 900x1600 resolution. For this session, we used a citywide free wi-fi network instead of a dedicated network as previous, to experience a realistic network configuration. A participant first played a game of Tic-Tac-Toe with the computer. Then, the participant used the Gallery to find an image of a dog randomly positioned from a list of 15 distractor images. Last, the participant used Maps to find downtown Seattle and Vancouver from San Francisco using only panning and zooming. For a good understanding of the interaction, participants performed the tasks twice: once interacting on their smartphone, and once using the touch-sensitive large display. Participants were asked to think aloud and give qualitative feedback.

Results

Using a within-subjects analysis of variance with distance and scale as factors, we found both factors have a significant effect on the task completion time ($p < 0.05$). Participants took longer to position the projection farther away (M=12s, SD=4s) than closer to the starting position (M=11s, SD=3s). Tasks including scaling to twice the dimensions took 12s on average (SD=4s) while those requiring minor scaling took 11s (SD=4s). Task completion times include acquiring the Stop button for each trial.

To find out how accurately participants positioned the projection region, we calculated the distance from the final projection to the target. Participants were able to accurately position the projection both nearby (offset M=2.1cm, SD=1.5cm) and far away (offset M=2.6cm, SD=2.6cm), i.e., a mean error of 3% of the total distance traveled.

We evaluated how accurately participants were able to scale the projection to match the target size, by considering the percentage amount the participant's projection differed from the target dimensions. Similar to positioning, participants were highly accurate in scaling. When scaling to 1.4 times the original dimensions, participants resized their projection to within 5.3% of the target size (SD=4.6%). When scaling to twice participants had a mean error of 4%. (SD=3.4%).

Qualitative Feedback

Overall, participants were highly positive regarding Open Project for sharing media, collaborate with others or expanding the output space. On a five-point Likert scale, participants rated Open Project highly useful for applications such as presentations (M=4.4), collaborating tasks (M=3.9), and for sharing media with others (M=4.4). Participants rated Open Project as very easy to understand (M=4.3) and easy to use (M=3.7). Participants saw Open Project as somewhat useful for mobile gaming (M=3.6).

In terms of qualitative feedback, participants saw Open Project as a useful addition to their mobile experience. Participants praised the ability to share information quickly and efficiently on a nearby display.

"It's difficult to share stuff regularly. Only one person can view the screen at a time. It's a turn-based experience." [P2].

In terms of the specifics of Open Project, participants found the physical positioning with the camera projection technique to be *"a natural instinct"*. Participants also saw Open Project as a way to control the privacy of their mobile device while still sharing some information.

"I am nervous to hand over my phone. With [Open Project], I don't have to hand over my phone that has a lot of private data. […] It's nice to have control over what is shared." [P10]

However, our participants did notice the performance was lacking for our current implementation.

"In terms of mobile gaming the performance needs to improve. However the concept is useful. With a better frame rate I would change my score accordingly." [P1]

Finally, participants suggested using Open Project to enable the smartphones as a central repository of data and computation, streaming information to various outputs. Participants thought of our web framework as effectively replicating common hardware used to stream music or video to remote devices with minimal deployment effort.

DISCUSSION

Open Project performs as much computation as possible on the mobile device, maintaining a lightweight server that simply acts as a router. Our current implementation achieves a similar image transmission frame rate to previous techniques (e.g. [3]). However, performance suffers in dynamic applications such as video sharing or realtime gaming. To improve the frame rate, future work can take better advantage of mobile hardware (e.g. using the GPU for marker detection) and use dedicated libraries

for content streaming such as VNC. As VNC is a point-to-point protocol, the Open Project Server will have to set up direct connections between a display and its remote clients.

The projection metaphor allows users to easily access any part of the display with radial movement of the arm and wrist. However, physical projection suffers from poor stability and angle-dependent distortion dependent on viewing angle. Open Project's marker-based positioning embodies a partial projection metaphor that takes advantage of the intuitive physical targeting but stabilizes the projection when the user interacts with the projected app. We can strengthen the projection metaphor, e.g., by adding a live view of the projected content to the marker while positioning. We can also enable multiple phones to simultaneously position their content by embedding a unique identifier (e.g., a QR code) into the marker.

Finally, the Open Project framework enables vision-based digital sharing at a large scale with minimal effort. The web-based architecture means that any display capable of showing a browser can be turned into an output device without installation. It is also possible to implement Open Projector (the mobile component in our framework) as a built-in function of a mobile platform for improved performance and simple invoking, e.g., via a hard button.

CONCLUSION

We presented Open Project, an open, web-based framework for enabling mobile sharing and collaboration at a large scale. It employs an intuitive, projection-based metaphor for a user to easily share a mobile application by projecting it onto a target large display. Open Project can turn any computer display projectable instantaneously and without deployment, and its "camera projection" algorithm, for simulating a projection effect, performs reliably in various viewing conditions. Developers can add support for Open Project in native mobile apps by simply linking our library, requiring no additional hardware or sensors. Our study participants responded highly positively to Open Project-enabled applications for mobile sharing and collaboration.

REFERENCES

1. Arthur, R. and Olsen,Jr., D.R. XICE windowing toolkit: Seamless display annexation. *ACM Trans. Computer-Human Interaction. 18*, 3 (2011), 14:1–14:46.
2. Ballagas, R., Rohs, M., and Sheridan, J.G. Sweep and point and shoot: phonecam-based interactions for large public displays. *CHI '05 Extended Abstracts*, 2005, 1200–1203.
3. Baur, D., Boring, S., and Feiner, S. Virtual projection: exploring optical projection as a metaphor for multi-device interaction. *Proc. of CHI'12*, ACM (2012), 1693–1702.
4. Beardsley, P., van Baar, J., Raskar, R., and Forlines, C. Interaction using a handheld projector. *Computer Graphics and Applications, IEEE 25*, 1 (2005), 39 –43.
5. Blasko, G., Feiner, S., and Coriand, F. Exploring Interaction with a Simulated Wrist-Worn Projection Display. *Proc. of Wearable Computers*, IEEE Computer Society (2005), 2–9.
6. Boring, S., Jurmu, M., and Butz, A. Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays. *Proc. of OzCHI'09*, 2009, 161–168.
7. Bragdon, A., DeLine, R., Hinckley, K., and Morris, M.R. Code space: touch + air gesture hybrid interactions for supporting developer meetings. *Proc. of ITS'11*, 2011, 212–221.
8. Cao, X. and Balakrishnan, R. Interacting with dynamically defined information spaces using a handheld projector and a pen. ACM Press (2006), 225.
9. Cao, X., Forlines, C., and Balakrishnan, R. Multi-user interaction using handheld projectors. *Proc. of UIST'07*, ACM (2007), 43–52.
10. Chang, T.-H. and Li, Y. Deep shot: a framework for migrating tasks across devices using mobile phone cameras. *Proc. of CHI'11*, ACM (2011), 2163–2172.
11. Fischler, M.A. and Bolles, R.C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395.
12. Greenberg, S. and Rounding, M. The notification collage: posting information to public and personal displays. *Proc. of CHI'01*, ACM (2001), 514–521.
13. Herbert, L., Pears, N., Jackson, D., and Olivier, P. Mobile Device and Intelligent Display Interaction via Scale-invariant Image Feature Matching. *PECCS*, 2011, 207–214.
14. Izadi, S., Brignull, H., Rodden, T., Rogers, Y., and Underwood, M. Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. *Proc. of UIST'03*, ACM (2003), 159–168.
15. Molyneaux, D., Izadi, S., Kim, D., et al. Interactive Environment-Aware Handheld Projectors for Pervasive Computing Spaces. *Pervasive*, (2012), 197–215.
16. Rekimoto, J. and Saitoh, M. Augmented surfaces: a spatially continuous work space for hybrid computing environments. *Proc. of CHI'99*, ACM (1999), 378–385.
17. Remondino, F. and Fraser, C. Digital camera calibration methods: considerations and comparisons. *Photogrammetry, Remote Sensing and Spatial Information Sciences,* (2006).
18. Rohs, M. Real-world interaction with camera phones. *Proc. of UCS'05*, Springer-Verlag (2005), 74–89.
19. Shen, C., Everitt, K., and Ryall, K. UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. *Proc. of UbiComp 2003*, (2003), 281–288.
20. Smith, G.M. and schraefel, m. c. The radial scroll tool: scrolling support for stylus- or touch-based document navigation. *Proc. of UIST'04*, ACM (2004), 53–56.
21. Tan, D.S., Meyers, B., and Czerwinski, M. WinCuts: manipulating arbitrary window regions for more effective use of screen space. *CHI'04 extended abstracts*, 2004, 1525–1528.
22. Wang, J., Zhai, S., and Canny, J. Camera phone based motion sensing: interaction techniques, applications and performance study. *Proc. of UIST'06*, ACM (2006), 101–110.
23. Wigdor, D., Jiang, H., Forlines, C., Borkin, M., and Shen, C. WeSpace: the design development and deployment of a walk-up and share multi-surface visual collaboration system. *Proc. of CHI'09*, ACM (2009), 1237–1246.
24. Wilson, A.D. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. *Proc. of UIST'10*, ACM (2010), 273–282.
25. OpenCV. http://opencv.org/.
26. zxing - 1D/2D barcode image processing library. http://code.google.com/p/zxing/.
27. Apache Tomcat http://tomcat.apache.org/.