SymmetriSense: Enabling Near-Surface Interactivity on Glossy Surfaces using a Single Commodity Smartphone

Chungkuk Yoo*
School of Computing, KAIST
Daejeon, Republic of Korea
ckyoo@nclab.kaist.ac.kr

Inseok Hwang†
IBM Research
Austin, TX, USA
ihwang@us.ibm.com

Eric Rozner
IBM Research
Austin, TX, USA
erozner@us.ibm.com

Yu Gu
IBM Watson Health
Austin, TX, USA
yugu@us.ibm.com

Robert F. Dickerson
IBM Mobile Innovation Lab
Austin, TX, USA
rfdickerson@us.ibm.com

ABSTRACT

Driven to create intuitive computing interfaces throughout our everyday space, various state-of-the-art technologies have been proposed for near-surface localization of a user's finger input such as hover or touch. However, these works require specialized hardware not commonly available, limiting the adoption of such technologies. We present SymmetriSense, a technology enabling near-surface 3-dimensional fingertip localization above arbitrary glossy surfaces using a single commodity camera device such as a smartphone. SymmetriSense addresses the localization challenges in using a single regular camera by a novel technique utilizing the principle of reflection symmetry and the fingertip's natural reflection casted upon surfaces like mirrors, granite countertops, or televisions. SymmetriSense achieves typical accuracies at sub-centimeter levels in our localization tests with dozens of volunteers and remains accurate under various environmental conditions. We hope SymmetriSense provides a technical foundation on which various everyday near-surface interactivity can be designed.

Author Keywords

Near-surface input; computing interfaces; fingertip localization; smartphone; single commodity camera.

ACM Classification Keywords

H.5.2. User Interfaces: Input devices and strategies (e.g., mouse, touchscreen)

*This work was done when the first author was on an internship at IBM Research - Austin.
†The corresponding author.

CHI'16, May 07 - 12, 2016, San Jose, CA, USA

ACM 978-1-4503-3362-7/16/05$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858286

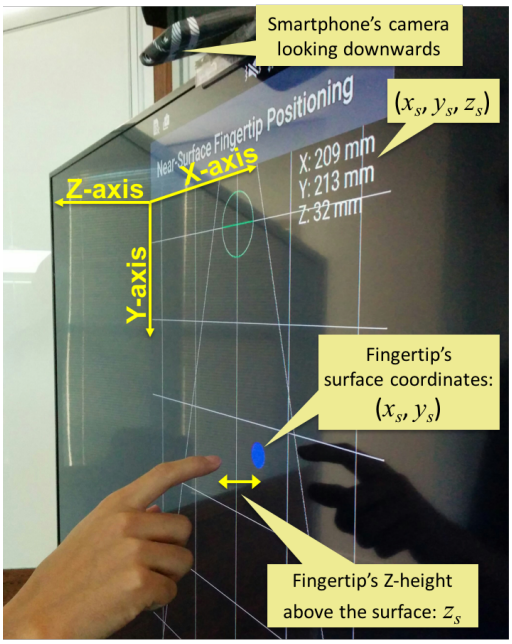


Figure 1: SymmetriSense pinpoints the fingertip's (x, y, z) position near a glossy surface with a single camera.

INTRODUCTION

A major vision of ubiquitous computing is to create seamless computing interfaces throughout the space of our everyday life. In this light, various technologies instrumenting everyday objects with on-surface or near-surface inputs have been proposed [5, 12, 14, 24, 28, 29, 34, 39, 40]. These technologies enable finger or hand movements to provide input either through contact or floating above the surfaces of instrumented target objects. Despite the advantages of these technologies, they require using dedicated hardware which can limit adoption of touch or near-touch interfaces for the user's surrounding objects.

We present *SymmetriSense*, a single camera-based technology that enables near-surface 3-D fingertip localization closely above arbitrary glossy surfaces. SymmetriSense is

uniquely advantageous in that it requires only a single commodity camera such as one from a smartphone to instrument a glossy surface, meaning that the technology is immediately available to almost everyone today. Examples of glossy surfaces include various home or office items such as wall mirrors, television screens with protective glass, or even kitchen countertops with a high gloss finish. By mounting the smartphone at the edge of the target surface as shown in Figure 1, SymmetriSense captures the real-time 3-D position of the user's fingertip within the near-surface space (i.e., the fingertip's X- and Y-coordinates on the surface and its Z-height up to several centimeters from the surface). SymmetriSense achieves a sub-centimeter accuracy on small form-factor surfaces (e.g., 5-inch diagonal) and still retains a centimeter accuracy on much larger surfaces (e.g., 60-inch diagonal).

A key challenge for enabling near-surface fingertip localization with a single regular camera is that an image taken from the camera's field of view lacks the precise distance to the finger. Without the distance, the fingertip's position cannot be uniquely determined with respect to the surface area. Although using cameras with depth-sensing features such as infrared or stereoscopy may resolve this issue [12, 14, 18, 19, 24, 30, 39], such specialized devices are still not as ubiquitous as a regular camera built in every smartphone. Their limited availability could significantly hinder many users from adopting such near-surface input technology throughout their everyday space. SymmetriSense presents a novel approach from the state-of-the-art by utilizing the principle of reflection symmetry and the fingertip's natural reflection cast upon a glossy surface. The glossy surface naturally creates a flipped image of the fingertip at a mirrored position in the camera's field of view. The principle of reflection symmetry provides a geometric condition of the pair-wise positions of the original and reflected fingertips, allowing us to obtain a unique solution for the fingertip's (x, y, z) position near the surface.

We expect SymmetriSense can serve as a platform on which hover, touch, or other kinds of finger gestures can be applied to various everyday items without the addition of new dedicated hardware devices. For example, a user may opt to use her own smartphone to instantly and temporarily enable this interface, or for a more permanent installation, she may use a recycled old smartphone and mount it to a granite countertop or a dressing table mirror. SymmetriSense can then create interfaces like virtual touch, proximity sensing, or drawing a pattern on particular regions of the dressing table mirror. Many applications could be designed to respond to such inputs in cooperation with IoT or connected home technologies, for example illuminating the room or narrating today's weather. SymmetriSense can also turn conventional displays like televisions or desktop monitors into interactive displays which respond to the user's finger inputs. SymmetriSense's sufficiently high localization accuracy allows for an interface design that accommodates multiple interactive buttons or regions, which may be sized and spaced similarly to those in interactive tablets or tabletops.

We present our design and implementation of a working SymmetriSense prototype using an off-the-shelf smartphone as well as a prototype that uses a 60-inch flat panel television. While our implementation and evaluation are focused on the smartphone, it should be noted that SymmetriSense can be implemented on any other platform supporting a regular camera and reasonable computing power. We evaluate SymmetriSense by reporting the basic localization accuracies with a group of users, as well as the impacts on accuracy over diverse environmental factors. To be specific, its finger-localization accuracy has been evaluated with volunteers of different ages, genders, and ethnic groups. Our evaluation also includes a robustness test under different lighting conditions and different surface colors and brightness. It is noteworthy that our particular implementation of SymmetriSense does not rely on parameters specific to human skin colors or textures; it can work with any long-shaped pointer in a finger-like dimension, such as a pen or a finger in a glove. We believe the evaluations outline the capabilities and limitations of SymmetriSense, delivering design considerations for developers to build interfaces on top of SymmetriSense.

The key advantages of SymmetriSense are threefold: (1) it offers a negligible barrier-to-entry because SymmetriSense can be built on single-camera commodity platforms such as a smartphone; (2) it features versatile applicability on various glossy surfaces providing a modest reflection; (3) its fingertip localization typically achieves a centimeter accuracy even on a large form-factor surfaces, enabling a fine-grained, sophisticated surface interface design.

In the next section, we outline related work in ubiquitous touch interactivity and near-surface interaction. We then describe our localization technique, which is inspired by the principle of reflection symmetry. Next we present our implementation process along with various engineering choices, and we then report the evaluation results in terms of localization accuracy and robustness. We conclude after discussing the limitations and remaining issues of SymmetriSense.

RELATED WORK

There exists a large body of pre-existing work to enable surface interactivity. In spite of this, SymmetriSense offers a unique advantage by allowing any user with a single commodity camera, such as a smartphone, to enable near-surface 3-D fingertip localization on various surfaces. Below we present a summary of major related works and differentiate SymmetriSense from them.

Surface Instrumentation for Ubiquitous Touch Interfaces

Bringing computing interfaces out of a few dedicated boxes to everyday space has been a major interest of ubiquitous computing and human-computer interaction. Along with the recent proliferation of touch-screen devices, instrumenting various everyday surfaces to respond to users' touch-like inputs has received much attention. A number of works addressed ubiquitous surface instrumentation by localizing the pointer on various surfaces. PlayAnywhere [39] uses infrared light sources and cameras with infrared pass filters to locate touches and hovers of fingertips on arbitrary surfaces. OmniTouch [14] and PointPose [19] use a depth camera oriented towards a surface to detect a larger set of finger input types

like hover, touch, tilt, and rotation. Inverted FTIR [12] uses a sheet of special acrylic glass with a laterally mounted infrared light source and takes advantage of frustrated total internal reflection to sense multi-touch locations on a display. WorldKit [40] makes a large ordinary surface instantly touch-interactive by deploying a paired projector and depth camera above the target. SMART Board [5] implements a touch-sensitive whiteboard which localizes the pointer by four cameras at each corner and determines a touch event by analyzing the pointer's on-surface reflection. Several other works instrumented surfaces for different types of surface interactivity not relying on fine-grained pointer locations: Touché [29] determined single- or multi-touch on conductive objects, Scratch Input [15] and Toffee [41] sensed on-surface scratch inputs, SurfaceLink [13] sensed on-surface directional motion of a pointer, and uTouch [7] sensed touch and hover over an LCD display by tapping on the powerline.

Different from the ubiquitous surface instrumentation works above, SymmetriSense presents a novel technology requiring only a single commodity camera to localize the user's near-surface fingertip with respect to an arbitrary reflective surface. Importantly, SymmetriSense is deployable as a smartphone app, making it highly practical for many end users.

Above-surface Input Technologies

Input technologies above a surface extend the input space beyond the discrete boundary of a conventional 2-D touch area. Diverse technologies have been proposed, including those designed for near-surface interactions in close proximity to the surface to others that support relatively distant interactions from the surface. For the space up to a few centimeters above a surface, SmartSkin [28] determines the depth information from a user's hand to the surface by a surface equipped with capacitive sensor grids, while Z-depth [34] senses the depth by multi-layered laser planes over the surface. Very recently a few smartphone models come with built-in hover-sensing screens like Samsung Air View [37] or Sony Floating Touch [36]. By using different types of touchscreen panels and/or different capacitance calibrations, they enable the phones to detect a hovering fingertip close to the screen. FlexAura [20] supports proximity sensing from an arbitrary object by wrapping the target object with a flexible-PCB sensor circuit. For relatively higher altitudes like several tens of centimeters above a surface, RetroDepth [18] uses a pair of stereoscopic infrared cameras to enable precise inputs on and above a retro-reflective material. Similarly, LeapMotion [24] uses an infrared transceiver device to detect multi-finger gestures in the space above the device. Sharp et al. [30] present a Kinect-based system tracking the detailed mid-air motion of a user's hand. Medusa [1] installs 138 proximity sensors around the perimeter of a tabletop to enrich the tabletop interaction with the user's body and arm location contexts.

SymmetriSense targets the near-surface space, i.e., a few centimeters above the surface. Within this space, SymmetriSense pinpoints the fingertip in terms of its spatial coordinates with respect to the surface area, at an accuracy comparable to those provided by the commercial devices with special hover-sensing screens such as Samsung Air View. SymmetriSense also provides the fingertip's height information above the surface which is not supported by those commercial devices. Later in the paper, we show the accuracy of SymmetriSense in comparison with Samsung Air View.

Designing Near-surface Interaction

With the emergence of near-surface input technologies, new interfaces and interaction guidelines incorporating on- and near-surface space have been studied. Hilliges et al. [16] explored interactions seamlessly switching between on a tabletop surface and above it. Spindler et al. [33] and Wacharamanotham et al. [38] studied the vertical thickness guidelines to design mid-air gestures within the near-surface space. Marquardt et al. [21] addressed the notion of continuous interaction space, treating the space on and above the surface as a continuum, and categorized various interactions making use of the space. Detailed interface examples include selective information exposure upon a hovering event [3], preselection guides to improve touch accuracy on small targets [25], hover-responsive dynamic user interfaces [8], and so on.

We clarify that crafting or assessing user interfaces designed on top of SymmetriSense is beyond the scope of this paper. We introduce the technical foundations of SymmetriSense and analyze its performance metrics to benefit future interface designs and usability assessments.

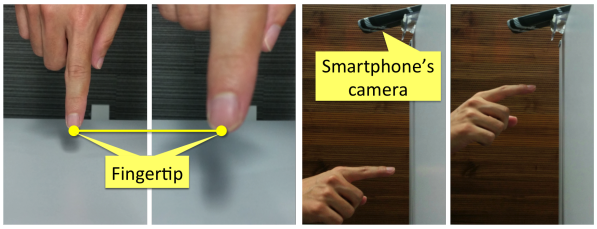
CHALLENGES AND KEY APPROACH

In this section, we first address the key challenge of localizing the fingertip with respect to the surface using only a single regular non-depth camera. Then we present our novel technique utilizing the natural reflection of the fingertip appearing on a glossy surface. The principle of reflection symmetry ensures the original image and its reflected image appear symmetric with respect to the plane of reflection, which is the interacting surface in our case. We will describe how we leverage this property to derive the 3-D spatial coordinates of the fingertip from a single 2-D image from the camera's field of view (FoV hereinafter).

Single Vision Challenge in Fingertip Localization

Figure 1 shows a possible placement of the camera (i.e., the smartphone) with respect to the target surface. The phone is simply placed at an edge of the television's bezel with a simple acrylic mount. This placement allows the phone's camera to see the perspective horizontal view of the surface, the fingertip, and the near-surface space in between. Also, placing the phone right at an edge of the target surface makes SymmetriSense easier and more intuitive to use, rather than placing the phone at a distant location such as on the ceiling. For the rest of the paper, we use the definitions of X-, Y-, and Z-axes as shown on Figure 1.

Figure 2 shows that fingertips at two different positions near a matte surface appear as the same points in the camera's FoV. Figure 3 depicts its sagittal cross-section diagram without the notion of reflection, explaining the uncertainty of the true position of the fingertip along the line of sight passing from the camera to the fingertip. One might wonder if the use of the fingertip's shadow casted on the surface would help. It may



(a) Viewed from the camera. (b) Viewed from the side.

Figure 2: Fingertips at two different locations appear at the same points in the camera's FoV.

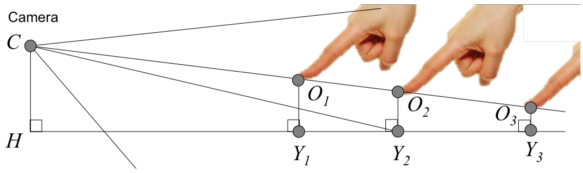


Figure 3: Uncertainty in localizing fingertips.

narrow the area in which the fingertip possibly exists, but it still does not pinpoint the fingertip's position as the shadow can be arbitrarily displaced depending on the fingertip's Z-axis height and the directions of external lighting sources.

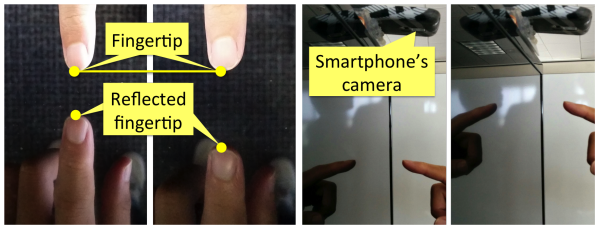
Pinpointing Fingertip by Leveraging Reflection Symmetry

Figure 4 demonstrates how the reflected fingertip makes it possible to pinpoint the fingertip's location. The figures are taken from our high-gloss 60-inch LCD TV screen under the illuminance of 150 lux, a typical office lighting condition [23]. Basically, the fingertips at two different locations along the camera's line of sight still appear at the same points in the camera's FoV. However, their reflected counterparts appear at different points from each other. This property can be described by a set of geometric equations, from which we can pinpoint the true location of the fingertip. Note that glossy surfaces with reasonable reflection are very common in our everyday space. Figure 5 demonstrates reflected fingertips on four different surfaces, i.e., a restroom mirror, an office window, a granite kitchen countertop, and a white board.

Our goal is to compute the near-surface coordinates of the fingertip (x_s, y_s, z_s) (denoted in Figure 1) with respect to the surface using the on-camera coordinates (x_{co}, y_{co}) and (x_{cr}, y_{cr}), the coordinates of the original and the reflected fingertip seen from the camera's FoV, respectively (denoted in Figure 7). Note that we presume a few constants of known values, such as the vertical and horizontal angles of the camera's FoV (denoted by Θ_v and Θ_h, respectively), its mounting angle (denoted by Θ_{HM}), the camera's offset distances from the surface edges (denoted by D_x and D_y, respectively), the Z-axis height of the camera above the surface, and so on. We discuss how to derive those constants in Discussion section.

Determining the Y-axis coordinate

For easier understanding, we begin with a simpler case where x_{co} and x_{cr} are zero, i.e., the fingertip is on a straight line from the camera in parallel with Y-axis. Figure 6 depicts the



(a) Viewed from the camera. (b) Viewed from the side.

Figure 4: Fingertips at two different locations still appear at the same points in the camera's FoV but their reflected counterparts do not appear at the same locations.

Figure 5: Fingertips reflected on (from left to right): a restroom mirror, an office window, a granite kitchen countertop, and a white board.

sagittal cross-section diagram of such a case, along the line from the camera to the fingertip. Now the problem is to find y_s, which can be expressed from Figure 6:

$$y_s = \overline{HC} \tan(\Theta_{HM} + \theta_{MY}) - D_y \tag{1}$$

where $\overline{HC}$ is the height between the camera and the surface.

To obtain the unknown angle θ_{MY}, we first use the property given by reflection symmetry: $\overline{OY} = \overline{RY}$ in Figure 6. This means that $\frac{1}{2}\left(\overline{RY_C} - \overline{OY_C}\right) = \overline{RY_C} - \overline{YY_C}$, which can be expressed by:

$$\frac{1}{2}\left(\tan\theta_{RY_C} - \tan\theta_{OY_C}\right) = \tan\theta_{RY_C} - \tan\theta_{YY_C} \tag{2}$$

$$\text{where } \theta_{YY_C} = \frac{\pi}{2} - (\Theta_{HM} + \theta_{MY}),\ \theta_{RY_C} = \frac{\pi}{2} - (\Theta_{HM} + \theta_{MR}),$$

$$\theta_{OY_C} = \frac{\pi}{2} - (\Theta_{HM} + \theta_{MO}) \tag{3}$$

Equation (2) is equivalent to:

$$\tan\theta_{YY_C} = \frac{1}{2}\left(\tan\theta_{RY_C} + \tan\theta_{OY_C}\right) \tag{4}$$

By using the definition of θ_{YY_C} in (3) on (4),

$$\theta_{MY} = \frac{\pi}{2} - \Theta_{HM} - \arctan\left(\tan\theta_{RY_C} + \tan\theta_{OY_C}\right) \tag{5}$$

Therefore, we can evaluate y_s in Equation (1) by using (5) and (3). Note that θ_{MR} and θ_{MO} can be obtained from y_{cr} and y_{co}, respectively. If we assume an ideal pinhole camera without a nonlinear distortion [11], it gives the following relation for θ_{MR}:

$$\frac{y_{cr}}{Y_{FOV}/2} = \frac{\tan\theta_{MR}}{\tan(\Theta_v/2)}$$

$$\text{Or equivalently, } \theta_{MR} = \arctan\left(\frac{y_{cr}}{Y_{FOV}/2}\tan\frac{\Theta_v}{2}\right) \tag{6}$$

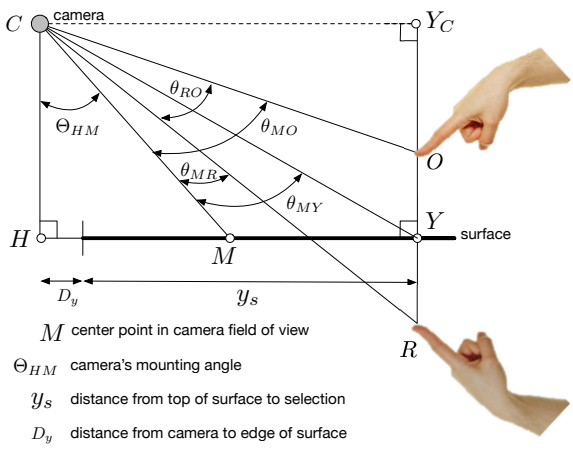


Figure 6: Sagittal cross-section diagram of a near-surface finger and its reflection.

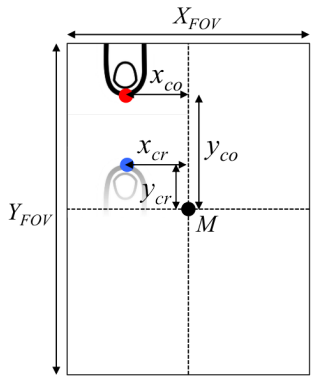


Figure 7: Notations to denote the on-camera coordinates of the original fingertip (x_{co}, y_{co}) and the reflected fingertip (x_{cr}, y_{cr}) as seen by the camera, with respect to the center (M) of the camera's FoV.

where Y_{FOV} is the vertical length of the camera's full FoV, as denoted in Figure 7.

The expression of θ_{MO} is analogous to (6). We applied a polynomial radial distortion correction in our actual implementation.

Determining the X- and Z-axis coordinates

The general case with nonzero x_{co} and x_{cr} is depicted in Figure 8. Once we obtain y_s, it is rather straightforward to compute the remaining near-surface coordinates, i.e., x_s and z_s. Note that z_s is equal to $\overline{OY} = \overline{YY_C} - \overline{OY_C}$. In turn, this difference is represented by the difference between two tangent values:

$$z_s = (y_s + D_y)\left(\tan\theta_{YY_C} - \tan\theta_{OY_C}\right)$$

which can be evaluated with (3), (5), and θ_{MO}'s version of (6). Finally, x_s is given by the Pythagorean theorem:

$$x_s = \sqrt{\overline{HC}^2 + (y_s + D_y)^2}\tan\theta_{YX} + D_x$$

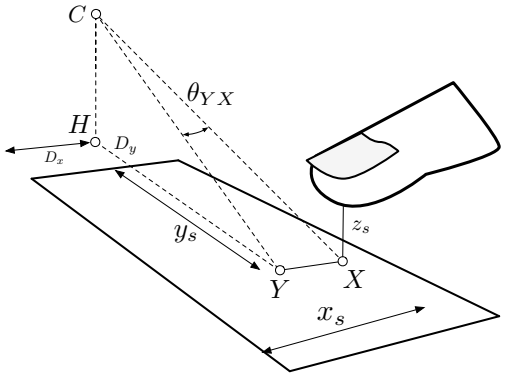


Figure 8: General case of localizing the fingertip in 3-D space.

Figure 9: Finger extraction. (a), (b): Raw images seen by the camera at two successive frames; (c): Finger extraction by applying skin color ranges; (d): Finger extraction by inter-frame differentials; (e): Computing convex hulls.

where the expression of θ_{YX} is analogous to (6), in terms of x_{cr}, X_{FOV}, and Θ_h.

IMPLEMENTATION

We present the issues and considerations in implementing a working prototype of SymmetriSense. While our prototype is built on Samsung Galaxy S5 with Android 5.0, it could be built on other phones or various off-the-shelf mobile platforms having a single regular camera combined with a embedded computer such as Arduino [2] or Rasberry Pi [27].

Extracting Finger Parts

The approach presented in the previous section assumes that we can extract the original and reflected finger parts out of the image seen from the camera's FoV. Our first attempt was to extract finger parts whose color values belong to the representative color range of the human skin [32]. We further specified the YCbCr color range between (0, 77, 133) and (255, 127, 173) [26]. However, we observed this technique is highly susceptible to background objects similar to skin colors; Figure 9c demonstrates this issue for the given raw image of Figure 9a. It is also unable to support non-finger pointers such as a hand in glove or pen.

To overcome the limitations of the color-based technique, we adopted alternative techniques based on frame differences [22]. When the user moves her finger near the surface, we extract the differentials between successive frames. Figure 9d demonstrates the differentials evaluated from two successive

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

Figure 10: Pen extraction. (a), (b): Raw images at two successive frames; (c) Inter-frame differentials; (d) Computing convex hulls around the pair of pen images.

frames shown in Figure 9a and 9b. We set empirically-determined threshold values to classify a motion at a certain position on the FoV. To suppress false-positives from non-finger motions in the background, we find a contour of the differential area between the frames and compute a convex hull enclosing the contour as shown in Figure 9e. We classify the differential area as a finger only for a convex hull larger than a threshold size. We used the Android OpenCV library to find contours and compute convex hulls. Another advantage in this technique is the ability to extract not only a fingertip but also many different long-shaped pointers, as shown in Figure 10.

Note that finger extraction is not a part of the novelty we claim; we integrated well-known, relatively simple image processing techniques to build a working prototype of SymmetriSense. Rather than simply using frame differences, there may be more advanced object-tracking techniques proposed in the field of image processing [42, 9, 31]. In particular, Cucchiara et al. [9] proposed a method to precisely extract a moving object without its shadow included, which may be applied to SymmetriSense to further improve the finger extraction performance. However, such advanced techniques are beyond the scope of this paper.

Detecting Fingertips from Finger Parts

In our camera-mounting configuration, the original finger appears in the top part of the view, while the reflected finger appears in the bottom part. We first find the original fingertip by searching for the bottom-most tangential point in the original finger part (indicated by the red arrow in Figure 9e). Starting from this point, we define a narrow rectangular space stretching downwards to search for the reflected fingertip.

To reduce computation, we leverage the locality of the moving fingertip. We define a region of interest (ROI) centered at the previously detected location on the FoV, only in which we perform fingertip detection. We empirically found an appropriate camera resolution to ensure real-time responsiveness and practical localization accuracy. We used 320×240 pixels for the camera resolution and its quarter size for the ROI. We apply these detected locations into the equations in the previous section to localize the fingertip's (X, Y, Z) coordinates. Detection is done at 30 frames per second.

Suppressing Image Noises and Localization Errors

Under low brightness conditions, we observed random noises on images regardless of a moving object due to the camera's ISO settings. We applied a blur filter for higher noise-robustness. To ensure sufficiently large differentials, we compute the difference between the current frame and a past one four frames behind. We also performed morphology operations (dilate and erode) to acquire well-connected finger shapes.

Due to the relatively low camera resolution of 320×240, a single pixel error often results in momentary hopping of a fingertip's locations. To alleviate such behaviors, we applied a 10-frame weighted moving average to smooth the fingertip's trajectory. Given the 10 points from those frames, we further rule out the largest outlier to improve accuracy. All combined, our localization pipeline takes 8.7 ms to compute each frame (not including the moving average filter lag). Due to the inherent causality of the moving average filter, real-time localization introduces a slight noticeable lag. Note that quadrupling the resolution would halve the impact of single pixel error, allowing a shorter moving average to do the equivalent smoothing effect, and thereby shorten the response lag. However, all these improvements are at the cost of extra computation. We will discuss these details in the Discussion section.

We observed subtle localization errors which tend to grow at outer regions of the camera's FoV. This is known as radial distortion [11] due to the non-ideal curvature of the camera's lens. We implemented additional correction based on the polynomial approximation model [11, 6]. As the correction parameters are variables in terms of the focal length, we disabled Android's auto-focus feature. In fact having a constant focal length is necessary in localizing the fingertip as it effectively fixes the camera's FoV.

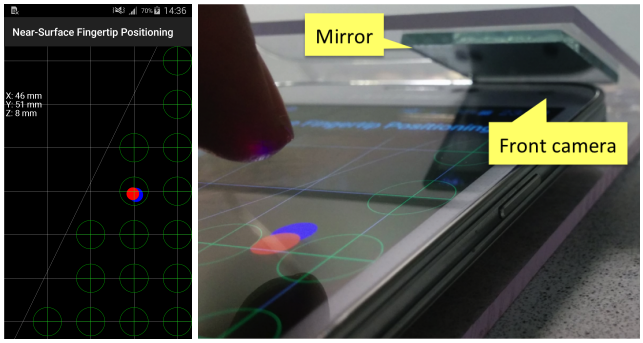
EVALUATION

The goal of our experiments is to report the basic fingertip localization accuracies of SymmetriSense under a wide range of participants as well as environmental conditions. We believe this provides numerical implications or guidelines for future interface designs on top of SymmetriSense. First, we describe our evaluation settings and then present the results.

Evaluation Settings

We perform three classes of experiments: a fingertip localization accuracy experiment on a smartphone, a fingertip localization accuracy experiment on a 60-inch television, and finally a set of microbenchmarks to analyze the accuracy under a variety of environmental conditions.

While SymmetriSense supports a wide array of arbitrary glossy surfaces, we include a smartphone experiment to compare against an alternative near-surface localization technology in the market, i.e., Samsung Air View. The Samsung Galaxy S5's Air View is a built-in hover sensing feature enabled by extended capacitive sensing supported in its specific display hardware. The experiment is a set of finger-hovering tasks on targets shown on different positions of the display. Upon a participant's fingertip above a target, we have SymmetriSense and Air View independently compute the fingertip's location. We consider the locations from Air View as the ground truth, and evaluate the fingertip localization accuracies of SymmetriSense by measuring the offset distance between SymmetriSense-computed locations and



(a) Test app screenshot on phone.

(b) A phone case with mirror.

Figure 11: Phone experiment settings.

the ground truth locations from Air View. One might wonder why we do not simply measure the offset distance between SymmetriSense-computed locations and the corresponding target locations shown on the display. The reason is to rule out possible human errors; the participants may not have perfectly positioned their fingertips on the targets. By measuring SymmetriSense-vs-Air View differences, we evaluate the accuracies of SymmetriSense comparative to those of Air View and irrespective of human errors.

In the TV experiment, we report only the localization results from SymmetriSense with respect to on-screen targets due to the absence of an alternative built-in localization technology. Therefore, the localization errors in TV experiments may include not only the errors from SymmetriSense technology itself but also human errors. Given that the phone experiment has shown the localization accuracies free from human errors, the TV experiment mainly demonstrates the efficacy of SymmetriSense on much larger surfaces.

To incorporate diverse finger shapes, skin colors, and pointing poses in the smartphone and TV experiments, we recruited 18 participants for each experiment. To be specific, the participants of the phone experiments include 4 females out of 18 and were 38.3 years old on average (std: 10.5 years). Those of the TV experiments include 4 females out of 18 and were 40.2 years old on average (std: 9.6 years). Our participants pool consists of left- and right-handed users, and mixed ethnicity of Asians, African Americans, Hispanics, and Whites.

Lastly, the microbenchmarks demonstrate if SymmetriSense is susceptible to various environmental conditions. To be specific, a series of small case studies were performed over varying conditions of surface colors, ambient lighting, and screen brightness. These experiments were conducted on the phone to use Air View as the ground truth.

Detailed setup for smartphone experiment. A Samsung Galaxy S5 phone was placed on a desk in a large corporate office. To retrieve the localization results from both technologies at minimal time differences, we slightly engineered the phone's exterior to make SymmetriSense localize a fingertip near the phone's own surface, where Air View is running as well. Figure 11b shows our phone on a custom acrylic mount with a mirror above its front camera. This setup produces geometrically similar effects as having the camera mounted at an edge of the target surface (as depicted in Figure 8).

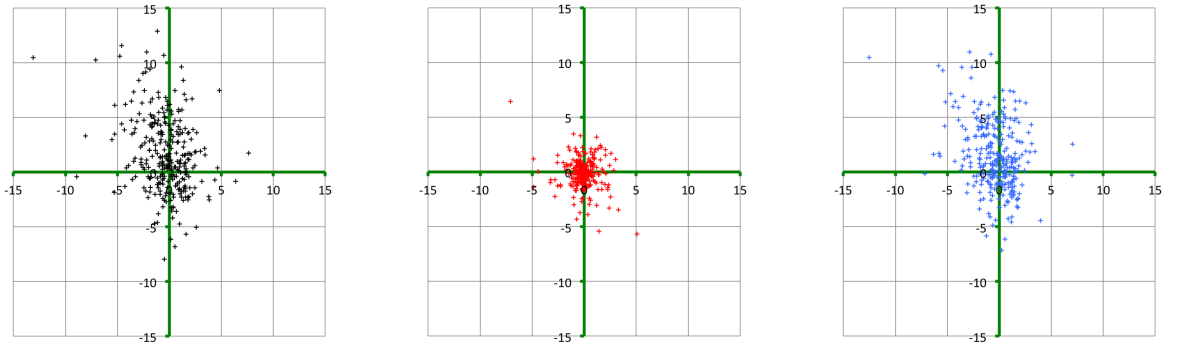
18 volunteers were asked to perform a sequence of positioning tasks on our test application (shown in Figure 11a). The test application shows 16 circular targets which are placed within the camera's FoV. The circles were 5 mm in radius and the centers of adjacent circles were separated by 15 mm on both the X- and Y-axes. For each circle, the participants were instructed to point at the center of the circle with a hovering fingertip for 5 seconds. During the experiment, two small dots were shown: a red dot, which corresponds to the location of the hovering finger as determined by the built-in Air View feature, and a blue dot, which corresponds to the location of the finger based on SymmetriSense. Participants were told to center the red dot in the circle. The process continued to show 16 circles, all of which are located within the camera's FoV. One might think the presence of the dots may influence a participant's targeting behaviors, e.g., trying to center the blue dot instead. Our metric is invariant in spite of such a participant; we consider the Air View's locations as the ground truth, and report the offset distances between SymmetriSense-computed locations and the ground truth. If a participant does not follow the instruction and purposely moves a finger elsewhere, AirView will displace the red dot to track the finger. It is impossible for the participant to manipulate the distance in between.

Detailed setup for TV experiment. A phone was affixed atop a 60-inch LCD TV in an office, and connected via an HDMI output. Figure 1 shows the setup, where the phone's rear camera is angled toward the TV surface. The mounting angle Θ_{HM} was 63.8 degrees, but SymmetriSense allows different mounting angles. Users were asked to stand in front of the TV and repeat an experiment similar to the phone experiment. Due to differences in surface areas, camera mounting positions, and camera vantage points, the TV experiment had a different number and placement of circles. A total of 22 circles were shown, each of which had a 25 mm radius and was separated by 75 mm on the X and Y axes. As there is no other technology available for ground truth, we report the offset distances between each circle center and the corresponding SymmetriSense-computed fingertip locations. No dots were displayed to avoid potential influences on the participants' targeting behaviors.

We additionally performed a single participant study to measure the Z-height accuracy. We mounted a ruler below the pointing finger and instructed the participant to hold the fingertip at several predetermined discrete heights. We visually inspected the ground truth heights from the recorded pictures and compared them with the Z-height measured by SymmetriSense. Note that Air View does not provide Z-heights.

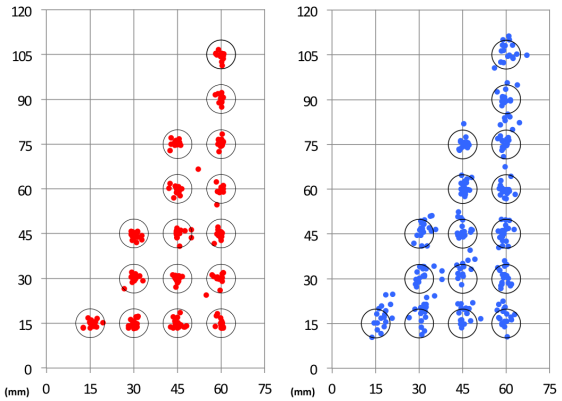
Evaluation Results

Localization accuracies on a smartphone. The raw data of fingertip localization over all users and all circles is shown in Figure 12. Figures 12a and 12b show the locations of the



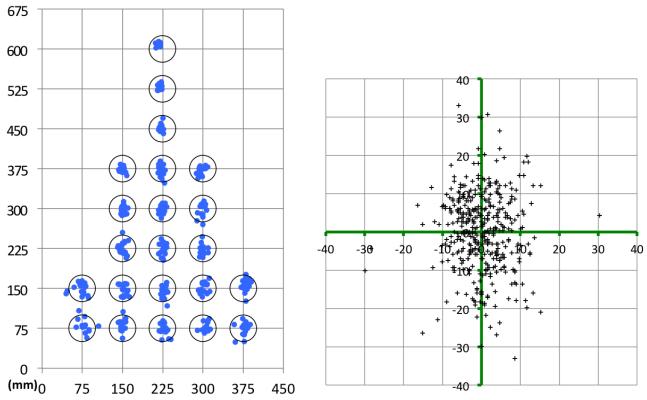
(a) SymmetriSense offset to Air View. (b) Air View offset to circle centers. (c) SymmetriSense offset to circle centers.

Figure 13: Offset, in millimeters, for Air View and SymmetriSense on the phone.



(a) Samsung Air View. (b) SymmetriSense.

Figure 12: Location estimates on the phone for all users.



(a) Location estimates. (b) Offset, in mm, to circle centers.

Figure 14: Location estimates on the TV surface and offsets.

fingertip hovering above each target, retrieved from Air View and SymmetriSense, respectively. The gridlines and circles corresponding to the actual screen content are also shown for reference. Our main evaluation metric, i.e., the offset distance between SymmetriSense-computed locations and Air View's ground truth locations, is shown in Figure 13a. For information purposes, Figure 13b and 13c present scatter plots of the locations from Air View and SymmetriSense respectively relative to each circle's center. As we discussed earlier, the results in Figures 13b and 13c may contain human error.

The amount of error is relatively small; the average offset distance between SymmetriSense-computed locations and the corresponding Air View's ground truth locations is 3.61 mm (std=2.55). As shown in Figure 13a, the errors on the X-axis are generally lower than the errors on the Y-axis (1.53 mm vs 2.63 mm average). Furthermore, the largest errors in SymmetriSense come from the furthest points from the camera: the furthest row has an average distance of 4.38 mm from their Air View counterparts, while the closer points have an average error of 2.55 mm. These limitations are addressed in the Discussion section.

Finally, we examine how SymmetriSense performs if we apply a *snap-to-grid* feature, i.e., snapping the estimated fingertip position to the closest circle center. We post-process the data to see if any estimate snaps to a wrong circle. The Air View estimates are 100% correct. SymmetriSense misclassifies only 2.78% (8 out of 288 samples). All of these errors occur on the two rows furthest from the camera.

Localization accuracies on a 60-inch TV. Figure 14 shows the X, Y accuracy results from the TV experiment. Figure 14a shows SymmetriSense's estimated locations and Figure 14b shows the offset of the estimated locations from the circle centers. As discussed earlier, part of the error from the circle centers is introduced by the users themselves. Regardless, the relative error is small: the average distance from the circle centers to our estimate is only 10.04 mm (std=6.20). Even though these errors are larger than the phone experiment, the error relative to the screen size is much smaller in this experiment. None of the snap-to-grid points are misclassified. We see similar trends as the phone experiment, with the average error on the Y-axis (7.89 mm) being higher than the average error on the X-axis (4.70 mm).

Figure 15 shows the the Z-height test results. The user hovered his finger 10, 30 and 50 mm above the screen and the Z-height estimate was captured. For each distance, 22 data

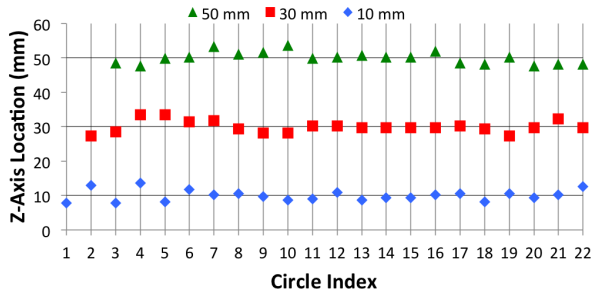
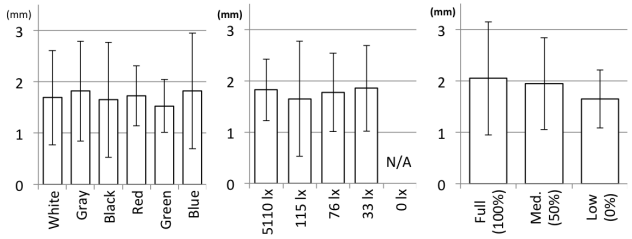


Figure 15: Z-axis estimate for 10, 30, and 50 mm.



(a) Errors over background colors. (b) Errors over lighting conditions. (c) Errors over screen brightness.

Figure 16: Environmental benchmark results.

points were captured, one for each circle center. We index the circles from 1-22, from left to right, and then top to bottom. For example, the top circle in Figure 14a is Circle 1, the 2nd-to-top is Circle 2, the bottom-left is Circle 18, and the bottom-right is Circle 22. Note that we are only able to capture the Z-heights that lie within the camera's vertical FoV. This means that for 30 mm height, we cannot capture Circle 1 (which is closest to the camera), and for 50 mm height, we cannot capture Circles 1 and 2. Overall, however, SymmetriSense does an effective job tracking the Z-height of a user's finger. The average error at the 10 mm height is 1.26 mm (std=0.99), at the 30 mm height is 1.24 mm (std=1.15) and at the 50 mm height is 1.35 mm (std=1.06). These results indicate that SymmetriSense can effectively estimate a user's 3-D fingertip location, which is a powerful primitive with only a single commodity camera.

Environmental benchmarks. The first microbenchmark is changing the color of the screen. The surface color in our earlier tests was black. To study the effectiveness of SymmetriSense on a variety of surface colors, a single user repeated the previous phone test on the surface colors of white, gray, black, red, green, and blue. Figure 16a shows the X, Y accuracies for each color in terms of the distance between AirView's and SymmetriSense's locations with one standard deviation error bars. SymmetriSense stays reasonably consistent across surface colors ($F_{5,90} = 0.240$, $p > 0.943$), indicating that the technology is fairly robust to surface colors.

Figure 16b shows the average error over varying ambient lighting conditions. Our earlier experiments on the phone and the TV were done under 115 lux and 150 lux, respectively, which are typical office illuminance levels [23]. To diversify the illuminance conditions, a single participant conducted tests on five conditions, i.e., in an office lighting (115 lux), in full sun in mid-afternoon (5100 lux), in a small conference room with a full set of lights on (76 lux), the same room with only the dimmer lights on (33 lux), and in complete darkness. Figure 16b shows similar errors across the conditions ($F_{3,60} = 0.180$, $p > 0.909$), usually ranging from 1-3 mm, indicating SymmetriSense can effectively deal with a range of ambient lighting conditions. Our scheme does not work in complete darkness.

Last, Figure 16c shows the average error for SymmetriSense over varying screen brightness settings. The same user repeated the experiments at 0%, 50% and 100% screen brightness. The screen background color was set to white. We see that screen brightness makes little difference ($F_{2,45} = 0.909$, $p > 0.410$), with low brightness giving slightly larger errors. Regardless, the results indicate that SymmetriSense can deal with varying screen brightness.

DISCUSSION

Limitations

Significant CPU utilization. In the camera resolution of 320×240, SymmetriSense shows a steady CPU utilization around 25%. It is the full CPU utilization for a single-threaded application on a quad-core CPU, which is the case of our Galaxy S5. This can be alleviated by increasing smartphones' computing power, exploiting parallelism, or computational offload to GPGPU or external devices [10]. As discussed in the Implementation section, the camera resolution affects several real-time performance metrics, including: (1) localization errors along the Y-axis and (2) the real-time localization lag resulted from the moving average filter length. We analyze the effects on (1) in detail below.

Larger localization errors along the Y-axis. In the evaluations we observed larger localization errors along the Y-axis than the X-axis, and growing Y-axis errors near the surface's distal edge from the camera. It is not surprising; the tangent function in Equation (1) takes the camera's mounting angle Θ_{HM}, in addition to the variable θ_{MY}. Because the tangent function diverges near $\pi/2$, the existence of Θ_{HM} makes the tangent function very sensitive to a small change of θ_{MY}. This sensitivity increases around the surface's distal edge where θ_{MY} is greater. Still, Θ_{HM} should be large enough to ensure a sufficient view of the surface from the camera.

To alleviate this sensitivity, processing the input images at a higher pixel-density would help. For example, in our setting with a 60-inch LCD television and camera resolution of 320×240 pixels, a single pixel width is converted into 25 mm of Y-axis distance at the farthest edge (60 cm away from the camera in our settings). In a future implementation with additional CPU power available, we can quadruple the pixel density (640×480) so that the converted distance per pixel is improved to 12 mm, giving smaller localization errors.

Limited FoV. SymmetriSense introduces near-surface interactivity only for part of the surface within the camera's FoV;

on a Galaxy S5 phone, the FoVs are 79.69° for the front camera and 62° for the back camera at maximum. Accordingly, designing an interface on top of SymmetriSense should follow a guideline such as placing interactive regions within this area. We may further extend the interactive area with a fish-eye lens and image deformation techniques [17].

Largely dynamic on-surface content. Our implementation leverages moving object tracking techniques to extract the fingertip from the scene. We have observed SymmetriSense performs well when applied on a static surface or a display with modest content changes. Some dynamic content is tolerated because screen content viewed from the edge-mounted camera often appears severely blurred and much dimmer. Still, applying SymmetriSense on a display with large dynamic contents might interfere with the finger extraction. A potential resolution is to evaluate the symmetry of a moving object and discard non-symmetric parts because content changes only appear on the screen side. We may also apply advanced image processing techniques such as optical feature-based image tracking on digital surfaces [4], or separating an object's reflection from a single image [35].

Battery consumption. The smartphone's battery consumption has not been a major consideration in our initial development. We acknowledge the smartphone running heavy vision tasks requires nontrivial energy. In our implementation, the battery drop is roughly 20% per hour if the phone is unplugged. Given this energy requirement, desirable application scenarios would be an instant, short-term interaction with a user's primary phone, or a permanent instrumentation with a plugged, possibly recycled, phone.

Design guidelines and considerations. While our results show SymmetriSense can accurately localize a fingertip, a few limitations were also brought to light. The goal of our accuracy studies is to provide developers with a series of best practices when using our system. For example, certain touch- or hover-sensitive regions should be separated by a safe distance. Alternatively, coarse-grained interactions, such as scrolling or swiping, can be implemented over areas with lower pin-point accuracy. For larger surfaces like TVs or mirrors, this may not be a big burden. This work aims to provide a first-order evaluation of SymmetriSense with the hope that future algorithms, uses and techniques can be refined.

Open Issues and Future Works

Deriving form-factor constants. SymmetriSense relies on a few constants like the camera's FoV, mounted height from the surface, and mounted angle. The camera's FoV angles are obtainable at runtime, e.g., via `Camera.Parameters` APIs of Android. The other constants can be obtained through a one-time calibration sequence after mounting the camera device, such as instructing the user to point to a few predetermined positions. In some cases, SymmetriSense may not need to convert the fingertip's position into real-world distance metrics, making a calibration sequence unnecessary. For example, an application letting the user define her own interactive regions on her dressing table mirror that respond to her finger's proximity to a region to execute various tasks. The design of a runtime calibration sequence is future work.

Incorporating a touch feature. SymmetriSense's touch support depends on how strictly we define a touch. We observed that SymmetriSense often does not clearly distinguish a finger physically contacting the surface from one floating very close, e.g., 1-2 mm above the surface. The reasons are (1) low camera resolution makes such a tiny gap unclear, and (2) a fat fingertip often obstructs the line of sight from the gap to the camera, making the camera see the fingertip's original image and reflected image slightly connected. If we accept a touch only if the fingertip physically contacts the surface (i.e., only when Z-height is 0.0 mm), SymmetriSense may generate non-trivial false positives. If we accept a relaxed definition of touch (e.g., if the gap is under 2 mm), we can say SymmetriSense supports touch. This may not be a radical assumption given that even capacitive touch screens sometimes misclassify a touch upon a very closely floating fingertip. Due to the delicate issue of touch definitions, we chose not to justify our redefinition of a touch in this study, and focused on pinpointing a user's fingertip within a near-surface space. Still, incorporating physical touches and near-surface interaction by a single uniform technology makes SymmetriSense more attractive, and a rigorous extension towards touch support and/or an equivalent touch definition is our future work.

Alternate input mechanisms. While most of our evaluation focused on locating a fingertip, we believe SymmetriSense will work with other input objects. We informally tested pens, styluses and gloves with our approach and the performance was promising. Our localization scheme relies on motion and convex hull detection, which allows for a wide variety of input objects. We leave detailed analysis to future work.

CONCLUSION

In this paper, we presented SymmetriSense, a technology that enables near-surface 3-D fingertip localization above arbitrary glossy surfaces, using a single commodity camera such as one from a smartphone. Unlike the state-of-the-art, SymmetriSense achieves localization by using only a commodity device which is ubiquitously available. To address the challenge using a single camera, we presented a novel technique utilizing the fingertip's natural reflection and the principle of reflection symmetry. We reported the implementation details and built a working prototype. Our performance evaluation shows sub-centimeter 3-D localization accuracy in most cases, even as environmental conditions change. We believe that SymmetriSense can serve as a platform that can instrument various everyday glossy surfaces with near-surface interactivity to enable new ubiquitous computing applications.

REFERENCES

1. Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. Medusa: a proximity-aware multi-touch tabletop. In *Proc. UIST 2011*. ACM, 337–346.
2. Arduino. Retrieved September 24, 2015 from https://www.arduino.cc.

3. Aras Balali Moghaddam, Jeremy Svendsen, Melanie Tory, and Alexandra Branzan Albu. Integrating touch and near touch interactions for information visualizations. In *Proc. CHI 2011 Extended Abstracts*. ACM, 2347–2352.
4. Dominikus Baur, Sebastian Boring, and Steven Feiner. Virtual projection: exploring optical projection as a metaphor for multi-device interaction. In *Proc. CHI 2012*. ACM, 1693–1702.
5. SMART Board. Retrieved December 28, 2015 from http://education.smarttech.com/.
6. Camera calibration with OpenCV. Retrieved December 26, 2015 from http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html.
7. Ke-Yu Chen, Gabe A Cohn, Sidhant Gupta, and Shwetak N Patel. uTouch: sensing touch gestures on unmodified LCDs. In *Proc. CHI 2013*. ACM, 2581–2584.
8. Victor Cheung, Jens Heydekorn, Stacey Scott, and Raimund Dachselt. Revisiting hovering: Interaction guides for interactive surfaces. In *Proc. ITS 2012*. ACM, 355–358.
9. Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25, 10 (2003), 1337–1342.
10. Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. MAUI: making smartphones last longer with code offload. In *Proc. MobiSys 2010 (MobiSys '10)*. ACM, New York, NY, USA, 49–62. DOI: http://dx.doi.org/10.1145/1814433.1814441
11. Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine vision and applications* 13, 1 (2001), 14–24.
12. Florian Echtler, Andreas Dippon, Marcus Tönnis, and Gudrun Klinker. Inverted FTIR: easy multitouch sensing for flatscreens. In *Proc. ITS 2009*. ACM, 29–32.
13. Mayank Goel, Brendan Lee, Md Tanvir Islam Aumi, Shwetak Patel, Gaetano Borriello, Stacie Hibino, and Bo Begole. SurfaceLink: using inertial and acoustic sensing to enable multi-device interaction on a surface. In *Proc. CHI 2014*. ACM, 1387–1396.
14. Chris Harrison, Hrvoje Benko, and Andrew D Wilson. OmniTouch: wearable multitouch interaction everywhere. In *Proc. UIST 2011*. ACM, 441–450.
15. Chris Harrison and Scott E Hudson. Scratch input: creating large, inexpensive, unpowered and mobile finger input surfaces. In *Proc. UIST 2008*. ACM, 205–208.
16. Otmar Hilliges, Shahram Izadi, Andrew D Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. Interactions in the air: adding further depth to interactive tabletops. In *Proc. UIST 2009*. ACM, 139–148.
17. Pan Hu, Guobin Shen, Liqun Li, and Donghuan Lu. ViRi: view it right. In *Proc. MobiSys 2013*. ACM, 277–290.
18. David Kim, Shahram Izadi, Jakub Dostal, Christoph Rhemann, Cem Keskin, Christopher Zach, Jamie Shotton, Timothy Large, Steven Bathiche, Matthias Nießner, and others. RetroDepth: 3D silhouette sensing for high-precision input on and above physical surfaces. In *Proc. CHI 2014*. ACM, 1377–1386.
19. Sven Kratz, Patrick Chiu, and Maribeth Back. Pointpose: finger pose estimation for touch input on mobile devices using a depth sensor. In *Proc. ITS 2013*. ACM, 223–230.
20. Shenwei Liu and François Guimbretière. FlexAura: a flexible near-surface range sensor. In *Proc. UIST 2012*. ACM, 327–330.
21. Nicolai Marquardt, Ricardo Jota, Saul Greenberg, and Joaquim A Jorge. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. In *Proc. INTERACT 2011*. Springer, 461–476.
22. Davide A Migliore, Matteo Matteucci, and Matteo Naccari. A revaluation of frame difference in fast and robust motion detection. In *Proc. 4th ACM international workshop on Video surveillance and sensor networks, 2006*. ACM, 215–218.
23. Evan Mills and Nils Borg. Trends in recommended illuminance levels: an international comparison. *Journal of the Illuminating Engineering Society* 28, 1 (1999), 155–163.
24. Leap Motion. Retrieved September 24, 2015 from https://www.leapmotion.com.
25. Anna Ostberg and Nada Matic. Hover cursor: Improving touchscreen acquisition of small targets with hover-enabled pre-selection. In *Proc. CHI 2015 Extended Abstracts*. ACM, 1723–1728.
26. Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai. A novel skin color model in ycbcr color space and its application to human face detection. In *Proc. International Conference on Image Processing, 2002*, Vol. 1. IEEE, I–289.
27. Rasberry Pi. Retrieved September 24, 2015 from https://www.raspberrypi.org.
28. Jun Rekimoto. SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In *Proc. CHI 2002*. ACM, 113–120.
29. Munehiko Sato, Ivan Poupyrev, and Chris Harrison. Touché: enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proc. CHI 2012*. ACM, 483–492.

30. Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim Christoph Rhemann Ido Leichter, Alon Vinnikov Yichen Wei, Daniel Freedman Pushmeet Kohli Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proc. CHI 2015*. ACM, 3633–3642.
31. Jeongho Shin, Sangjin Kim, Sangkyu Kang, Seong-Won Lee, Joonki Paik, Besma Abidi, and Mongi Abidi. Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging* 11, 3 (2005), 204–218.
32. Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. In-air gestures around unmodified mobile devices. In *Proc. UIST 2014*. ACM, 319–329.
33. Martin Spindler, Marcel Martsch, and Raimund Dachselt. Going beyond the surface: studying multi-layer interaction above the tabletop. In *Proc. CHI 2012*. ACM, 1277–1286.
34. Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *Proc. ITS 2010*. ACM, 91–94.
35. Robby T Tan and Katsushi Ikeuchi. Separating reflection components of textured surfaces using a single image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27, 2 (2005), 178–193.
36. Sony Floating Touch. Retrieved September 24, 2015 from http://developer.sonymobile.com/knowledge-base/technologies/floating-touch.
37. Samsung Air View. Retrieved September 24, 2015 from http://www.samsung.com/us/support/answer/ANS00044011/997407174/.
38. Chat Wacharamanotham, Kashyap Todi, Marty Pye, and Jan Borchers. Understanding finger input above desktop devices. In *Proc. CHI 2014*. ACM, 1083–1092.
39. Andrew D Wilson. PlayAnywhere: a compact interactive tabletop projection-vision system. In *Proc. UIST 2005*. ACM, 83–92.
40. Robert Xiao, Chris Harrison, and Scott E Hudson. WorldKit: rapid and easy creation of ad-hoc interactive applications on everyday surfaces. In *Proc. CHI 2013*. ACM, 879–888.
41. Robert Xiao, Greg Lew, James Marsanico, Divya Hariharan, Scott Hudson, and Chris Harrison. Toffee: enabling ad hoc, around-device interaction with acoustic time-of-arrival correlation. In *Proc. MobileHCI 2014*. ACM, 67–76.
42. Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)* 38, 4 (2006), 13.