Reflector: Distance-Independent, Private Pointing on a Reflective Screen

Jong-In Lee[†1], **Sunjun Kim**[†1], **Masaaki Fukumoto**[2], **and Byungjoo Lee**[*1,2]
[1]Graduate School of Culture Technology, KAIST, [2]Microsoft Research Asia
{yi-jong-in,inornate,byungjoo.lee}@kaist.ac.kr, fukumoto@microsoft.com

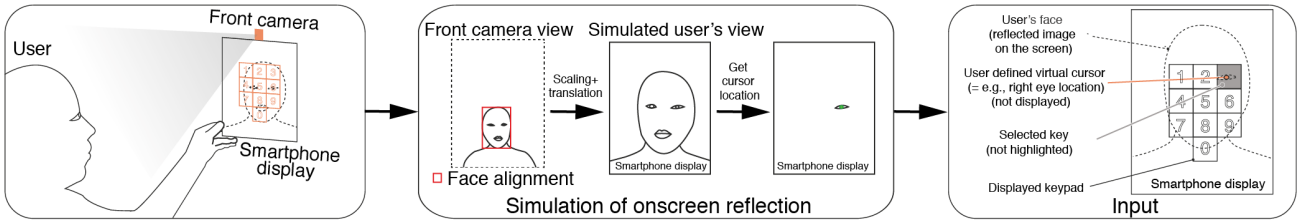

Figure 1. Reflector is a novel pointing method that utilizes hidden design space in reflective screens. By aligning a part of the user's onscreen reflection (e.g., the dominant eye) with objects rendered on the screen, Reflector enables (1) *distance-independent* and (2) *private* pointing on commodity screens. Distance independence means that changes in distance between the user and screen do not affect the user's input movement. This property enables more robust pointing performances to distance changes compared to methods such as eye-trackers or mid-air pointing. Reflector allows private pointing because bystanders cannot observe the same image reflected on the screen that the user sees.

ABSTRACT
Reflector is a novel direct pointing method that utilizes hidden design space on reflective screens. By aligning a part of the user's onscreen reflection with objects rendered on the screen, Reflector enables (1) distance-independent and (2) private pointing on commodity screens. Reflector can be implemented easily in both desktop and mobile conditions through a single camera installed at the edge of the screen. Reflector's pointing performance was compared to today's major direct input devices: eye trackers and touchscreens. We demonstrate that Reflector allows the user to point more reliably, regardless of distance from the screen, compared to an eye tracker. Further, due to the private nature of an onscreen reflection, Reflector shows a shoulder surfing success rate 20 times lower than that of touchscreens for the task of entering a 4-digit PIN.

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

Author Keywords
Direct input; onscreen reflection; distance independence; private pointing; shoulder surfing; touch screen; eye tracking.

UIST 2017, October 22–25, 2017, Quebec City, QC, Canada

ISBN 978-1-4503-4981-9/17/10…$15.00
DOI: https://doi.org/10.1145/3126594.3126665

INTRODUCTION
A visual display, or screen, is the most important output device in our everyday computer interactions. Modern displays are capable of rendering millions of pixels with millions of colors, which are optimized to convey a huge amount of concurrent information to users. Since the emergence of touchscreens, the function of the screen has extended to input as well. As input and output share the same space, current designs have highlighted direct manipulation paradigms. In direct manipulation, no cursor is required, unlike in traditional computing. Touch [1, 15, 26, 70], gaze [63], and mid-air pointing [3, 27, 37, 54, 68] are the most common examples of the input method following the paradigm.

Direct manipulation, or cursor-less interaction, has several advantages over cursor-mediated interaction. First, designers need not have to worry about the transfer function that maps the user's movement to the cursor's movement (i.e., the control-display gain function). Consequently, direct manipulation reduces the time needed for users to adapt to the system transfer function [46]. This feature is an important advantage, because designing a good transfer function in an indirect system is complex and difficult [43, 42, 41]. Moreover, when multiple users interact with one screen [34], presenting multiple cursors is distracting [36] and direct input (e.g., touch) is preferred.

However, despite the advantages of direct manipulation, there are two side effects due to the inherent properties of screens. First, a screen is *publicly visible*. Thus, direct input on a screen may also be visible to others. This property has raised privacy issues with interactive screens, such as shoulder surfing

†The first two authors contributed equally to this work.
*Corresponding author.

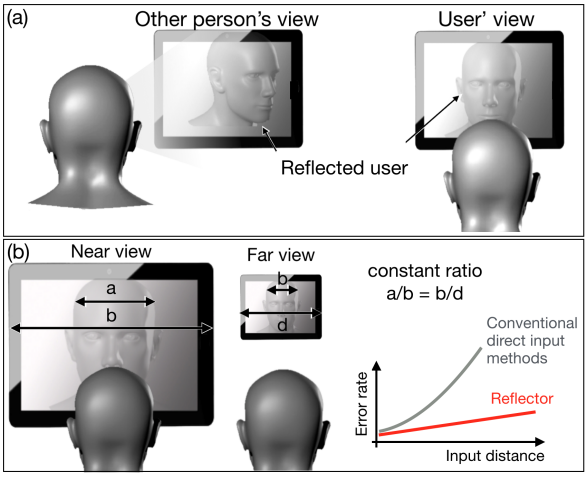

Figure 2. (a) The image reflected on the screen is private because it looks different to others. (b) The ratio of the size of the reflected landscape to the size of the screen is constant regardless of the user's distance from the screen; thus, the amount of user movement contributing to the input is not affected by the input distance between the user and screen. This dramatically reduces error due to distance variation compared to conventional direct input methods based on the user's angular movement (i.e., eye trackers and mid-air pointing).

on touchscreens [12, 58] or limiting activity on large public displays [67, 30].

Secondly, in a direct input system, further distances often make it difficult for users to interact with the screen [71]. A touch input system is, of course, not usable at a distant because the input space is limited to within the user's reach. For more distant setups, such as eye- and motion-tracking systems, distance adversely affects both the system and user's precision: In the *system's perspective*, with such computer vision-based input methods, the captured user image becomes smaller with distance; fewer pixels (that is, less information) contribute to the recognition of a user's movement due to the limited camera resolution. From the *user's perspective*, the screen gets smaller with increasing distance, which reduces the possible amount of user movement for the input. For example, a distant user must control their eye or finger more subtly to obtain the same level of control. However, since there is a limit to the precision of human motion, the smaller motion amplifies the error on the screen. Because these two sources of degradation occur in concert, the pointing performance of conventional systems generally decreases in proportion to the *square of the distance* ($=O(d^2)$).

To overcome these issues, this study examines *onscreen reflection* as a hidden input space for screen-based interactions that can overcome the aforementioned limitations of existing direct input systems. Compared to other readily available systems, onscreen reflection is (1) physically visible only to the user and (2) the geometric proportion of the user's reflected image and the screen remains constant regardless of distance (see Figure 2b). Therefore, this method is inherently private, and the amount of input movement can remain the same at any distance. The only remaining source of performance degradation is limited camera resolution; therefore, the precision of this system would *linearly decrease* in proportion to the distance ($=O(d)$).

To effectively show the characteristics of onscreen reflection as a novel screen input space, we implement a direct pointing method called Reflector. Reflector utilizes an onscreen reflection to select a target displayed on the screen. It is applicable to commodity reflective screens. More specifically, the user employs the reflected image of a body part (e.g., an eye or a fingertip) on the screen as a pointer (see Figure 1). To implement Reflector, the system captures the user's position in real time through a camera attached to the screen; based on the geometric relationship between the user, camera, and screen, it can infer the reflected image on the screen from the user's point of view.

The main contribution of this study is to propose a novel input technique exploiting the unique characteristics of a reflected image and evaluate its performance through experiments. To the best of our knowledge, this is the first attempt to utilize the physical properties of onscreen reflection for a pointing technique.

RELATED WORK

Reflective Screens Currently on the Market

Modern display panels predominantly use a glass substrate, which is inherently glossy. An antiglare coating may be applied for a matte screen, which generally consists of the rough surface with a fine grain-like texture. The texture diffuses incident light to prevent reflection, but it produces a hazy image; thus, glossy screens are more appealing to customers [6] and exhibit better saturation and contrast reproducibility [7]. This is a trade-off, and users select between gloss and matte screens depending on their requirements, operating environment, and preference.

As Reflector is designed for a glossy screen, we surveyed how many screens on market have sufficiently reflective display panels. We collected information from commodity displays, including 141 monitors, 56 TVs, 49 all-in-one PCs, and 167 laptops from five leading manufacturers (LG, Samsung, Apple, Dell, and HP)[1]. We investigated the specifications of all monitor products presented in each manufacturer's online store of to see if they were anti-glare treated. If there was no detailed explanation of screen glossiness, we visually checked the glossiness by watching a review video of the product.

Overall, 53.3% (220/413) displays were glossy (Figure 3). If we exclude monitors designed for desktop computers, which are commonly used with a keyboard and mouse, glossy screens were the dominant screen type (78.3%, 213/272). Considering that suppliers such as Apple (which produces glossy screens) produce only a few product models, but its actual sales volume is higher than that of other suppliers, the percentage of glossy screens purchased by consumers may be higher than the percentage of products in the market. Also, some devices, such as TVs, are not accompanied by an efficient pointing device, and the majority use glossy screens. Our proposed method could be used immediately for these devices.

[1] Retrieval date: January 10, 2017

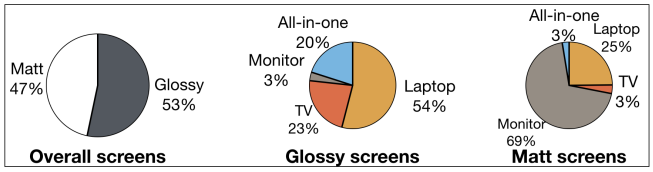

Figure 3. About half of the screens sold on the market are intentionally produced to be reflective, because glossy screens are more appealing to customers and exhibit better saturation and contrast reproducibility. In this study, we present onscreen reflections as a new alternative for direct input.

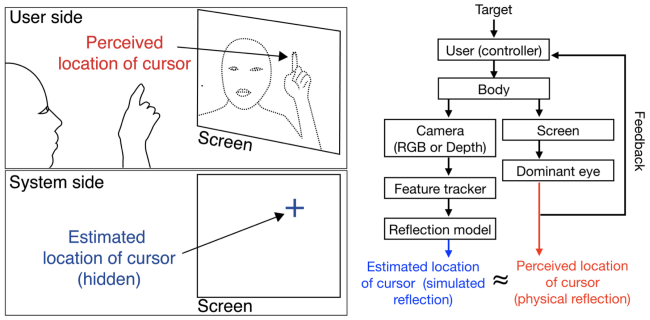

Figure 4. The Reflector system runs in the background when the user is pointing through his or her reflection. The system tracks the user's face and body through the front camera installed at the edge of the screen and reconstructs the reflection that the user sees on the screen.

Direct Input Methods on Screens

Of the diverse direct manipulation methods, touch is the most common and widespread. Researchers have extensively explored and suggested intuitive and efficient touch gestures, such as pin-and-crossing [47], consecutive tapping [29, 44], shearing [28], and dragging [4, 25] to enrich touch input. To enhance touch input, additional sensors such as pressure and motion sensors [22, 32], or finger identification [23] have been proposed. However, as we mentioned in the introduction, touch input is often exposed to others and is inadequate for a distant user.

The privacy issues of touchscreens have been addressed in previous research by utilizing back-of-device input [14] or communicating through a hidden tactile communication channel [13], which require separate hardware. A recent study [39] devised a private input that combines gaze and touch using only the front-facing camera of a mobile phone; however, the eye tracking used combined with touch to form an indirect input, which was complex and unnatural. In addition, some studies [69, 56] have attempted to increase input complexity. This change makes it difficult for hackers to steal input, but the input process also becomes more difficult for the user, causing usability losses.

For direct manipulation at a distance, mid-air pointing [27, 54, 68] and gaze input methods [17, 33, 49, 52] have been proposed. Despite extensive research, user pointing error increases with distance when using these methods because they often require subdegree angular body movement [10]. Even if this drawback is accounted for, users with finger impairments cannot even use the mid-air input method. A recent study [40] implemented accurate gaze input on multiple screens, but there is a limit to users having to wear a head-mounted eye tracker.

Interaction Techniques Utilizing Onscreen Reflection

Reflections have been noted for a long time by researchers because of their distinctive characteristics, and utilized as an interaction medium in numerous works. However, most studies have focused only on their visual aspects, rather than utilizing the inherent optical properties of reflection. Some examples used reflection to superimpose a virtual image over the reflected user image [2, 65] or to blend virtual images with reflected image of real space [21, 45, 50, 57]. Still others have used reflections on a half-silvered mirror to augment virtual images that interact with the user's hands or physical objects [53, 31, 59, 24]. Although these methods employed reflection for user interaction, they tended to use mirrors simply to redirect light, and did not fully utilize the unique optical properties of reflection: (1) distance-independence and (2) the private nature of the image. Moreover, they tend to require complex hardware configurations and a large space for system implementation. In this paper, we focus on utilizing the unique characteristics of onscreen reflections as a novel pointing method. The method solves the problems of the direct input method described above, and uses only simple off-the-shelf instruments.

Estimating Onscreen Reflection

To use reflection as an interaction method, a proper reflection estimation method is needed to transform the camera image to the onscreen reflection seen from the user's viewpoint. A number of works have used single or multiple image capture devices, such as RGB or depth cameras, to extract positional information from a human body to estimate the precise viewpoint and simulate a mirror image [61, 60, 45, 62]. These approaches generally requires rigorous hardware setups and considerable computing power for real-time tracking, though Francois and Kang devised a relatively simple reflection model with a single camera to achieve this result [20, 19]. In this study, we show that Francois and Kang's model can be further simplified when the location of a feature point on the user's face can be tracked in real time. Furthermore, we present a method that employs a depth camera to resolve their model when one of its important assumptions is broken.

IMPLEMENTATION

Figure 4 shows an overview of the Reflector system. First, the user aligns the reflected image of their body part (e.g., a fingertip in Figure 4) with a target on a screen. At the same time, the system captures the user's face and body using a front-facing camera fixed at the edge of the screen. Finally, the system infers the desired location by estimating the transformation from the frame captured by the camera to the onscreen reflection seen from the user's viewpoint.

Reflector requires one RGB camera when pointing through the feature points on the face, and one depth-sensing camera when pointing with body parts that are not located on the face.

We first describe the input through the RGB camera and then discuss how to extend pointing-capable body parts to the hands or feet.

Estimation of a Reflected Image

An earlier study [20] combined a camera and screen to simulate a virtual mirror. In that study, three important assumptions were made to approximate the mirror experience provided to the user: (1) the camera is assumed to be a pinhole model, and (2) the world—including the user's face—is assumed to be a flat plane parallel to the camera's imaging plane (a *flat-world assumption*), and finally, (3) the imaging plane of the camera is assumed to coincide with the screen surface (see Figure 5). Given these assumptions, researchers have devised an equation to relate the user's view of the screen to the camera image:

$$x_p = \frac{d(f+d)x'_p}{f(f+d+t_z)} + \frac{dt_x}{(f+d+t_z)}$$
$$y_p = \frac{d(f+d)y'_p}{f(f+d+t_z)} + \frac{dt_y}{(f+d+t_z)} \quad (1)$$

Here, (x_p, y_p) is the position where an arbitrary point P on the flat-world is reflected on the screen when viewed from the user's point of view, and (x'_p, y'_p) is the position at which the image of the same point P is captured on the camera's imaging plane (see Figure 5). Note that (x_p, y_p) is the coordinate value defined relative to the user's mirrored viewpoint. Also, (x'_p, y'_p) is a coordinate value defined relative to the center of the camera. The values of (t_x, t_y, t_z) represent the translation distance between the user's viewpoint and the camera center in each coordinate axis. f is the focal length of the camera and d is the distance between the user's viewpoint and the mirror's imaging plane. Note that focal length and depth of field are different concepts, and for pinhole cameras, focal length is finite and depth of field is nearly infinite.

In this case, values such as (t_x, t_y, t_z) may be difficult to measure because they are real-world distance values. However, the value of t_z can easily be expressed as $(d-f)$ because of the third assumption that the image plane of the camera coincides with the screen. We can further simplify the equation by assuming that the camera can track one feature point (or reference point) on the face in real time. When the distance from the center point on the camera image to the reference point of the face is (x'_r, y'_r), t_x and t_y can be expressed from the proportionality of two triangles ($\triangle ABC, \triangle ARD$) as follows:

$$t_x = \frac{(f+d)}{f}x'_r + a_x \quad \text{and} \quad t_y = \frac{(f+d)}{f}y'_r + a_y \quad (2)$$

Here, a_x and a_y represent the translational distance between the user's viewpoint and the reference point on the user's face. Finally, Equation 3 can be expressed without any transitional terms (t_x, t_y, t_z) as follows:

$$x_p = \frac{(f+d)x'_p + (f+d)x'_r + a_x}{2f}$$
$$y_p = \frac{(f+d)y'_p + (f+d)y'_r + a_y}{2f} \quad (3)$$

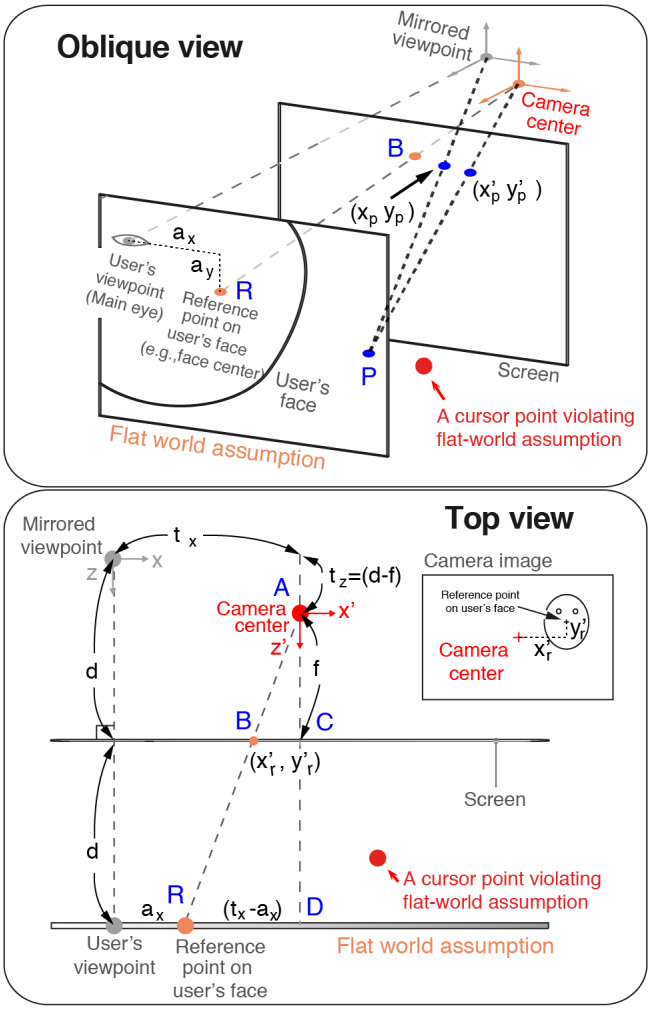

Figure 5. The Reflector system can reconstruct a reflection onscreen at the user's point of view from three assumptions: (1) the camera is assumed to be a pinhole model, and (2) the world including the user's face is assumed to be a flat plane parallel to the camera's image plane (a *flat-world assumption*), and finally, (3) the image plane of the camera is assumed to coincide with the screen surface ($t_z = (d-f)$).

If we continue, these equations take on a simpler form consisting of coefficients A and B, as shown below:

$$x_p = Ax'_p + (Ax'_r + B_x) \text{ and } y_p = Ay'_p + (Ay'_r + B_y)$$
$$\text{where, } A = \frac{(f+d)}{2f}, \text{ and } B_x = \frac{a_x}{2f}, \quad B_y = \frac{a_y}{2f} \quad (4)$$

This final equation shows that the transformation from the camera image to the onscreen reflection results in a simple translation $(Ax'_r + B_x, Ay'_r + B_y)$ after uniform scaling A. The translation amount changes depending on the coordinate value (x'_r, y'_r) from the center of the camera image to the reference point (R in Figure 5). However, it is important that B_x and B_y in Equation 4 are constants, as the distance from the reference point of the face to the user's view point (a_x, a_y) is always constant. Note that here, we apply the same scaling A for x and y coordinates of the camera image, which assumes the

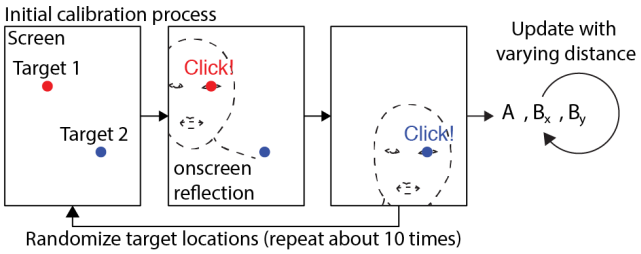

Figure 6. Reflector must be initially calibrated once for each user. This process is similar to that of an eye tracker, and the image transformation obtained through calibration is updated in real time as the distance between the user and screen changes.

same aspect ratio between the original camera image and the world. If the aspect ratio of the camera image is different from the aspect ratio of the real world, we should set A_x and A_y separately.

In summary, the transformations shown in Equation 4 allows Reflector system to overlap the camera image with the reflected image that the user sees on the screen. However, Reflector should be able to obtain both translation $(Ax'_r + B_x, Ay'_r + B_y)$ and scaling A in real time to estimate the user's perceived cursor location. As per the definition of A, B_x, and B_y, the transformation changes with a variable: distance d between the user's viewpoint and the screen.

Tracking On-body Cursor Point of a User

To realize the transformation defined in Equation 4, Reflector system should be able to detect and track the location of the user's face and the body cursor within the camera image frame. If the cursor is defined as a part of the user's face, then both the user's viewpoint and the cursor can be tracked by a single face alignment algorithm [38]. The algorithm is optimized to find the locations of important facial features—such as the eyes and mouth—more robustly. However, its detection speed is rather slow and works for only one face at a time. To address this issue, we use a simpler tracking module with a GPU-powered `OpenCV` cascade classifier that can track a user's face more quickly.

If the cursor point is not defined as a feature on the user's face, then we need to use other techniques that can track that feature. Commercial depth-sensing cameras, such as the Microsoft Kinect, can track the user's skeleton. These features provide sufficient functionality in most cases for pointing through the hands or feet.

Initial Calibration

After the system begins tracking feature points of the user's face and body, Reflector requires an initial calibration done by the user without varying his or her distance from the screen. This calibration process is similar to the 9-point calibration scheme of a typical eye-tracker and is done only once for each user. During calibration, one of the user's eyes reflected on the screen is employed as a cursor to select the targets displayed on the screen (see Figure 6). Note that in this calibration step, the cursor point is restricted to the user's face, to not violate the flat-world assumption.

At each target selection, the system compares the amplitude of movement in the camera frame to the amplitude of movement in the screen frame to obtain scaling factor A. Then, by substituting the obtained A value into Equation 4, the B_x value and B_y values can also be obtained. This process is repeated about 10 times, and all the values of A, B_x, B_y are averaged to remove possible high-frequency noise from the user movement and tracking module. The total time required for this calibration is approximately 3 minutes on average. The exact calibration algorithm is shown in Algorithm 1.

Updating the Scaling Factor with Varying Distance

After the initial calibration, the system is ready to simulate an onscreen reflection when the distance between the screen and user does not change from that of the initial calibration step. However, as shown in Equation 4, the value of A is a function of current distance D (= $(f+d)$), which is the usual variation during direct interaction with screens. To compensate for this effect, Reflector simply updates the current value of A in real time using the equation below:

$$A = A_0 \frac{(f+d)}{(f+d_0)} = A_0 \frac{D}{D_0} \tag{5}$$

Here $(f+d_0)$ represents the distance of the user from the camera at the calibration step (D_0 in Algorithm 1). When the system is implemented with a depth-sensing camera, $(f+d)$ can easily be obtained by looking at the depth at the user's viewpoint. However, when the system is only implemented with an RGB camera, the absolute measurement of $(f+d)$ is usually not available. In those cases, the system should utilize the camera-view frustum to estimate the relative distance between the user and camera. The size of a photographed object decreases in inverse proportion to the distance from the camera. If D_{eye0} represents the pixel distance between the left and right eyes during the calibration step that serves as a surrogate for absolute distance D_0, then the value of A can be

Algorithm 1 Calibrating Reflector

procedure INITIAL CALIBRATION
  $A_N, B_{xN}, B_{yN}, D_N, N \leftarrow 0$
top:
  randomize $Target_1(x_1, y_1), Target_2(x_2, y_2)$
  $Input_1(x'_1, y'_1) \leftarrow$ *User input aiming at* $Target_1$
  $Input_2(x'_2, y'_2) \leftarrow$ *User input aiming at* $Target_2$
  $A_N \leftarrow A_N + \frac{(x_1 - x_2)}{2(x'_1 - x'_2)}$ and $D_N \leftarrow D_N +$ *user's current distance*
  $B_{xN} \leftarrow B_{xN} + (x_1 - 2A_N/N \cdot x'_1)$
  $B_{yN} \leftarrow B_{yN} + (y_1 - 2A_N/N \cdot y'_1)$
  $N \leftarrow N + 1$
  if $N < N_{threshold}$ **then**
    goto *top*
  else
    goto *finish*
finish:
  $A_0 \leftarrow A_N/N, B_{x0} \leftarrow B_{xN}/N, B_{y0} \leftarrow B_{yN}/N, D_0 \leftarrow D_N/N$
  return A_0, B_{x0}, B_{y0}, D_0

updated using the current eye distance D_{eye}, as shown below:

$$D \propto \frac{1}{D_{eye}} \Rightarrow A = A_0 \frac{D}{D_0} = A_0 \frac{D_{eye0}}{D_{eye}} \quad (6)$$

After applying this real-time update to A, the system's main pointing module is implemented and the camera frame is overlapped on the onscreen reflection seen from the user's viewpoint, regardless of the distance between the user and screen.

Relaxation of the Flat-world Assumption

So far, we have presented an implementation of Reflector that only works with a cursor located on the same plane as the user's face (see Figure 5). If the feature point on the face—such as the user's eyes or nose—is used as a cursor, the implementation so far is sufficient. In the case of interaction with a mobile device, in which the user holds the screen in his or her hand, feature points on the face are more likely to be used as cursors. However, when interacting with a desktop or large screen, other parts of the body—such as the user's hands or feet—are often used as cursors, which means that we have to worry about implementing a cursor that violates our flat-world assumption (see the red dot in Figure 5).

First, for Reflector system to use a body part that is not on the flat-world as a cursor, it must be able to track the distance of that body part from the screen (or the center of the camera, approximately), which is made possible using a commercial depth-sensing camera such as the Kinect.

Next, the Reflector system must compensate for two kinds of errors that arise from the incorrect assumption that the body part is on the flat-world: (1) errors from the camera view frustum and (2) errors from the view frustum of the user. We explain these two error compensation below.

Let z be the distance from the cursor point to the screen as shown in Figure 7. The plane of the flat-world passing through the user's face is a distance d away from the screen. The location of the tracked cursor relative to the center point of the camera's RGB frame x_1 can be projected to x_2 on the flat-world due to the characteristics of the camera view frustum:

$$x_2' = (d/z) \times x_1' \quad (7)$$

After correcting the error caused by the camera view frustum, we correct the error from the reflection model that occurs when a cursor located in the flat-world moves closer to the screen by $(d-z)$. Now, the location of the reflection of the corresponding point on the screen changes from x_3 to x_4 with respect to the user's viewpoint:

$$x_4 = (2 - 2z/(d+z))x_3 \quad (8)$$

In the above equations, if z equals d, it satisfies the flat-world assumption, and we can see that $x_2' = x_1'$ and $x_4 = x_3$.

Input Triggering

Reflector, like eye tracking, is merely a pointing system, and needs a separate trigger for acquiring a target. Various signals can be considered as input triggering candidates in Reflector system. If a screen is accessible to the user, a touch event can be a trigger (as in Study 2, described later in this paper).

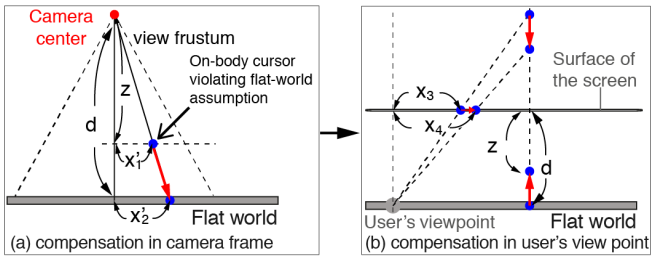

Figure 7. A cursor on the user's body that does not satisfy the flat-world assumption can be used as a cursor through two compensations: (a) and (b). These processes recompensate for errors that occur when a cursor point that is not located on the flat-world is assumed to be on the flat-world. Reflector system uses a commercial depth sensing camera because it must know the physical distance from the screen to the cursor z and the user's viewpoint d.

With a distant screen, a separate controller (e.g., an IR remote controller) may be used. Without such a device, the dwell-time method is common for eye tracking or mid-air pointing. In recent years, smooth-pursuit tracking [17, 66], which compensates for the shortcomings of the dwell-time method, has been widely studied and can also be used in Reflector system. Finally, eye blinking [11] may be used, accompanied with face detection.

STUDY 1: DESKTOP POINTING

We conducted a user study to evaluate Reflector 's performance and observed the effect of screen–user distance. We set an eye-tracker-based pointing system as the baseline, because it has standardized performance compared to mid-air pointing and is available on the market. In this study, we compared the performance of the two systems at different distances without updating the initial calibration settings obtained at the closest distance (0.5 m). We confirmed that Reflector worked well at the longest distance (2.5 m), at which an eye-tracker does not work.

Method

Participant: We recruited 12 participants from a local university. The average age was 27 years old ($\sigma = 6.27$); 7 were male and 5 were female. Five participants wore glasses and the remainder participated with naked eyes. The experiment took about 2 hours per participant, and participants were compensated 20,000 KRW for their time.

Design: The experiment used a 2×2 within-subject design with two independent variables: the *input distance* between the user and screen {0.5 m, 0.9 m} and the *input method* {REFLECTOR, EYE TRACKER}. Each participant tested all four conditions, and we counterbalanced the presentation order of conditions across participants using Latin-square design. We measured the *trial completion time*, *error rate*, and *throughput* as dependent variables.

Apparatus: All apparatuses except a depth camera were the same in both *input methods*. The application for the experiment was implemented in C++ and shown on a 23-inch glossy monitor with 1920 (51 cm)×1080 (29 cm) resolution (LG

23ET63). The operating system used was Microsoft Windows 8.1, and the hardware specification was 3.4 GHz Intel Core i7, 16 GB RAM, and a GTX 650 graphic card. A monitor arm (Camel deskmount MA-2) was used to place the monitor perpendicular to the ground. A height-adjustable chair was used to set the proper height based on the screen for each participant.

In the EYE TRACKER condition, participants gazed at the screen for pointing. We used the Tobii Eyetracker 4C, which has an operating range from 0.5 m to 0.95 m. In the REFLECTOR condition, participants pointed using the fingers of the dominant hand as the cursor. We used the Microsoft Kinect for Windows v2 to track the user's face and fingers. Instead of inferring the distance from the user to the screen indirectly through the eye distance (see Equation 6), we updated the calibration through the absolute distance from the depth-sensing camera to the participant's point of view (see Equation 5). We used the Face Tracking SDK[2] to detect a face. The pointing finger was tracked by finding the closest point to the screen in the depth map.

To reduce the jitter of the output from the EYE TRACKER condition, we applied a 1€filter [9] to the signal. As per recommendations from a recent study [18], each parameter in the filter was selected as follows: for the x-coordinate of the cursor, f_{cmin} was 0.24, β was 0.08; for the y-coordinate of the cursor, f_{cmin} was 0.46, and β was 0.07. The same filter was applied with different parameter values for locating the user's face in the REFLECTOR condition. We decided the REFLECTOR values to reduce lag and minimize jitter: f_{cmin} was 0.01 and β was 0.007 for both the x- and y-coordinates.

Device calibration: We calibrated instruments once for each individual participant 0.5 m from the screen, for the best performance in each *input method*. For REFLECTOR, each participant was instructed to perform the calibration as described in Figure 6. On average, the calibrated REFLECTOR parameters were 1.92 ($\sigma = 0.3$), 1071 ($\sigma = 139.2$), 666.5 ($\sigma = 99.1$) for A, B_x, and B_y, respectively (see Equation 4). For the EYE TRACKER, each participant performed 7-point calibration according to manufacturer recommendations. The experimenter asked the participant to perform this as accurately as possible. At the end of the calibration, the experimenter made the cursor visible on the screen and confirmed that the cursor matched the participant's intended pointing position. In the case of the REFLECTOR condition, the cursor should be located at the position of the right or left eye from the participant's viewpoint after calibration; in the case of the EYE TRACKER condition, the cursor should be located at the gaze point.

Task: Participants performed a standardized point-and-select task following ISO 9241-9 standards [16]. They pointed to a target using one of the *input methods* and acquired the target by dwelling on it for 0.75 s. In each trial, two targets were presented as red (start) and blue (end) circles on a black background. Participants were asked to acquire the red and blue targets sequentially. With successful target acquisition, the red target changed to cyan and the blue target changed to yellow. We used two target widths—17.5 mm and 36 mm—and three target distances—100 mm, 142.5 mm, and 187.5 mm. For each width–distance pair, we tested 10 angles of approach (10×36° rotation). Therefore, one block consists of 2 widths × 3 distances × 10 angles of approaches = 60 trials. The order of pairs was randomized within a block.

During a trial, each target appeared for only 7 seconds. After the time limit, the trial was treated as a failure. This is sufficient time to conduct actual pointing movements with both *input methods*. This limit also prevents the experiment time from becoming extremely long if input is impossible due to severe errors in the system.

Note 1) The 0.75 s dwell time is based on the fact that it takes 0.2 s to 0.6 s for humans to look at and recognize an object [35]. Using a shorter dwell time makes it difficult to tell whether the user simply starts watching the target or actually intends to select it [48].

Note 2) The target widths were determined not to be too difficult for the eye tracker, based on a recently published paper [18]; a commercial eye tracker has a success rate of 90% with 39 mm targets.

Procedure: After a participant's arrival, we briefly informed him or her about the experimental procedure. They sat on a chair throughout the experiment. We adjusted the chair height to align the participant's eye level with the middle of the screen. First, the participant calibrated the forthcoming *input method*. This process was performed only once at a distance of 0.5 m from the screen for each *input method*. After this, the experimenter explained the pointing tasks. The participant practiced the pointing and target acquisition methods for 20 trials. During practice, a cursor attached to the tip of the participant's dominant hand is set to be visible. The cursor was hidden during actual trials. We asked participants to perform the task as quickly and accurately as possible.

One block consisted of 60 trials; two blocks were performed for each *input method*×*input distance* condition to test for a learning effect. We allowed participants to rest freely between blocks. After finishing a trial condition, the participant rated the condition using the NASA Task Load Index (NASA-TLX) questionnaire [3]. Before testing the next condition, the participant practiced again to adapt to the condition.

In total, each participant performed 2 (*input distances*) × 2 (*input conditions*) × 2 (*blocks*) × 60 (trials) = 480 trials. After all conditions were completed, the participant was asked to measure one more block using REFLECTOR at a distance of 2.5 m. This is an additional step to observe how Reflector's error rate changes over longer distances. Trial completion time was measured from the acquisition of the red target to the acquisition of blue target. Note that a constant dwell time of 0.75 s was included in this definition. The error rate was calculated as $\frac{\text{\# of failed trials}}{\text{\# of total trials}}$. Throughput was calculated as $\frac{TrialCompletionTime}{ID_e}$, where ID_e was the effective index of difficulty in bits (see [16]).

[2] http://msdn.microsoft.com/en-us/library/jj130970.aspx

[3] http://www.keithv.com/software/nasatlx/nasatlx.html

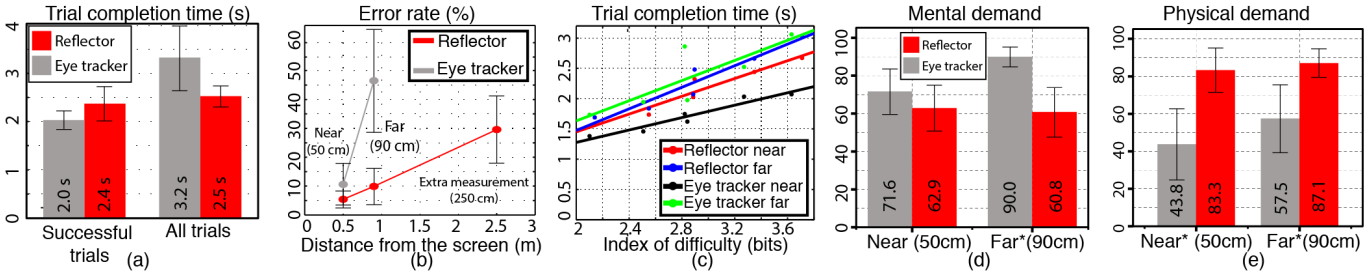

Figure 8. (a) The difference in trial completion time between the two was not significant when only including successful trials. (b) Overall, the error rate of REFLECTOR was 73% lower than that of the EYE TRACKER. As the distance increased, the error rate of EYE TRACKER increased sharply, but REFLECTOR's error rate was much more robust against changes in distance. (c) The throughput of REFLECTOR was lower than the throughput of the EYE TRACKER because it requires large upper limb movements. When we measured the workload of the two input methods, REFLECTOR was better than the EYE TRACKER in terms of (d) mental demand, and the EYE TRACKER was better than REFLECTOR in terms of (e) physical demand.

Results and Discussion

Learning Effect

We first briefly looked at the learning effect by observing the effect of *block* on *trial completion time*. We performed a three-way repeated-measure ANOVA (RM-ANOVA) on *trial completion time* with three independent variables: *input distance*, *input method*, and *block*. We found that *block* exhibited a significant main effect on *trial completion time* ($F_{1,11} = 6.13, p = 0.025$) where no interaction between other variables were observed. The *trial completion time* in the second block ($\mu = 2.2$ s, $\sigma = 0.69$ s) was significantly shorter than in the first block ($\mu = 2.4$ s, $\sigma = 0.82$ s). Therefore, we excluded data from the first block in all subsequent analysis.

Trial Completion Time

For the successful trials, we performed a two-way RM-ANOVA with two independent variables: *input distance* and *input method*. *Input method* showed a marginal main effect ($F_{1,11} = 3.4, p = 0.091$) and *input distance* showed a significant main effect ($F_{1,11} = 7.7, p = 0.018$) on *trial completion time*. There was no interaction between them ($F_{1,11} = 0.1, p = 0.76$). As REFLECTOR required larger body movements than the EYE TRACKER (see Figure 8a), the *trial completion time* for REFLECTOR ($\mu = 2.4$ s, $\sigma = 0.84$ s) increased marginally over that of the EYE TRACKER condition ($\mu = 2.0$ s, $\sigma = 0.46$ s). It is also possible to determine that applying different filter values to the two input methods leads to different delays, which increases the input time of REFLECTOR. However, we believe that the delay from filtering would not have a significant effect on the user's movement since the cursor was not visible. As expected, the mean *trial completion time* at 0.9 m ($\mu = 2.4$ s, $\sigma = 0.86$ s) was significantly increased over that at 0.5 m ($\mu = 2.0$ s, $\sigma = 0.36$ s).

However, when we included failed trials (the *trial completion time* for failed trials was set to 7 seconds), the difference between the two conditions became significant ($F_{1,11} = 9.9, p = 0.01$) and reversed. The *trial completion time* measured in the EYE TRACKER conditions ($\mu = 3.2$ s, $\sigma = 1.6$ s) becomes much slower than the REFLECTOR conditions ($\mu = 2.5$ s, $\sigma = 0.52$ s), because of the higher error rate of the EYE TRACKER at longer distances (see Figure 8b).

Error Rates

Following an identical process as the time analysis, we performed a two-way RM-ANOVA on *error rates*. *Input method* ($F_{1,11} = 19.6, p = 0.001$) and *input distance* ($F_{1,11} = 22.0, p = 0.001$) both showed significant main effects on *error rates*, and a significant interaction between them was observed ($F_{1,11} = 9.7, p = 0.01$). The *error rate* at 0.9 m ($\mu = 28.2\%$, $\sigma = 27.9\%$) was higher than that of pointing at 0.5 m ($\mu = 8\%$, $\sigma = 8.9\%$), and the *error rate* of REFLECTOR ($\mu = 7.6\%$, $\sigma = 7.9\%$) was lower than that of the EYE TRACKER ($\mu = 28.5\%$, $\sigma = 27.8\%$).

In a pairwise *t*-test with Bonferroni correction, the difference between two *input methods* was not significant at a distance of 0.5 m ($p = 0.188$), and was significant at a distance of 0.9 m ($p = 0.002$). At a 0.5 m *input distance*, the *error rate* of REFLECTOR and the EYE TRACKER were 5.42% ($\sigma = 4.72\%$) and 10.6% ($\sigma = 11.3\%$), respectively, but at a 0.9 m *input distance*; REFLECTOR was 9.9% ($\sigma = 9.9\%$) and the EYE TRACKER was 46.5% ($\sigma = 28.2\%$).

As we predicted earlier, the *error rate* of the EYE TRACKER condition steeply increased with distance even within the claimed available range (0.5 m to 0.95 m) of the eye tracker. REFLECTOR, however, was relatively stable; particularly, pointing at 2.5 m, which the additional trial that the participant performed with REFLECTOR, showed an *error rate* of 30.0% ($\sigma = 18.3\%$). If we plot the *error rates* over *input distances*, we can confirm that REFLECTOR exhibits a linear function (see Figure 8b). In contrast, the EYE TRACKER becomes unusable at over 0.9 m. Even if a new calibration is made at each distance (although this is not an ecologically valid assumption), the *error rate* of the eye tracker still increases with the square of the perturbed distance ($O(\Delta d^2)$), and the *error rate* of Reflector increases linearly with the perturbed distance ($O(\Delta d)$).

Throughput

We calculated *throughputs* only for successful trials and plotted them in Figure 8c. With a linear regression, the *throughput* of REFLECTOR was 1.36 bits/s ($R^2 = 0.93$) at 0.5 m and 1.12 bits/s ($R^2 = 0.92$) at 0.9 m. The *throughput* of the EYE TRACKER condition was 1.95 bits/s ($R^2 = 0.92$) at 0.5 m and 1.2 bits/s ($R^2 = 0.68$) at 0.9 m.

Workload

The average workload scores measured in NASA-TLX are (lower is better): EYE TRACKER at 50 cm = 63.8 ($\sigma = 17.9$), EYE TRACKER at 90 cm = 81.9 ($\sigma = 9.4$), REFLECTOR at 50cm = 67.4 ($\sigma = 12.8$), and REFLECTOR at 90 cm = 68.722 ($\sigma = 13.38$). We performed a two-way ANOVA on the scores. *Input distance* exhibited a significant main effect on workload ($F_{1.11} = 10.4, p = 0.008$), but *input method* did not. A significant interaction between *input distance* and *input method* was observed ($F_{1,11} = 7.8, p = 0.017$). In post-hoc tests with Bonferroni correction, there was no difference in workload at 0.5 m ($p = 0.57$), but the workload of REFLECTOR was significantly lower than that of EYE TRACKER at 0.9 m ($p = 0.013$). For component-wise comparison, the EYE TRACKER required higher mental demand than REFLECTOR at 0.9 m (Figure 8d), but the EYE TRACKER consistently required less physical demand than REFLECTOR (Figure 8d). These are convincing results, because REFLECTOR obviously requires more physical movement than the EYE TRACKER , but the high error rate of the EYE TRACKER at 0.9 m results in more mental effort and a greater workload.

In summary, workload increased and performance decreased with distance, and the difference between the two *input methods* is more noticeable at greater distances. The EYE TRACKER and REFLECTOR exhibited similar performances and workload at close distances, but REFLECTOR overwhelmed the EYE TRACKER in error rate and workload at long distances. From these results, we conclude that our proposed interaction, Reflector, is not only as efficient as eye tracking but also robust at long distance.

STUDY 2: MOBILE PIN AUTHENTICATION

The onscreen reflection is unique for each viewer, so Reflector enables naturally private interaction. In particular, touch input on touchscreens is known to be vulnerable to a visual attack known as *shoulder surfing*. In the second study, we measured the robustness of Reflector against a shoulder surfing attack compared to a touchscreen in a 4-digit PIN input task.

Method

Participant: Ten participants were recruited from a local university. The participants ages ranged from 22 years to 35 years ($\mu = 27.4$ years; $\sigma = 4.76$ years). Two participants wore glasses or contact lenses and all participants had normal or corrected-to-normal vision.

Experimental Design: The experiment used a within-subject design with a single independent variable: *authentication method*{REFLECTOR, TOUCH}. In the REFLECTOR condition, the experimenter entered PINs with Reflector implemented on a commodity smartphone equipped with a front camera. In the TOUCH condition, a regular touch input was used on the same device. Two dependent variables were used to evaluate the performance of each authentication methods: *success rate* and *input duration*. We assumed that input duration can be used as a measure of confidence in the shoulder surfer's input.

Task and Materials: The task was a simple input alternation between a true user and a hacker. Throughout the experiment, the experimenter played the role of the true user, and the participants played the role of a hacker who steals the 4-digit PIN that the experimenter enters. First, the experimenter (the true user) memorized a 4-digit PIN and input it as quickly and accurately as possible using the given *authentication method*. After watching the input process, the participant (hacker) guessed the password entered by the experimenter and tried the guessed password *via touch*.

The participants and experimenter used the same visual interface to input the password. The visual interface conformed to a general smartphone format. The background of the interface was black, and the outline of the visual interface was in white (see Figure 9). With each input, the lower four squares were sequentially filled with white to indicate input progress. The experimenter used his right eye as a cursor and the initial calibration of the experimenter for the REFLECTOR condition was maintained throughout the experiment. Participants touched the screen in the way that they used their usual smartphone. Square buttons with an 80-pixel width (12.4 mm) were used.

Procedure: Before beginning of the experiment, participants were instructed to understand the underlying principle of our system fully by watching several demonstrations conducted by the experimenter. The experimenter was seated at a desk while the participant freely utilized the entire workspace to accommodate various shoulder surfing attacks. The laptop that informed the experimenter of a random password for each trial was located behind the participant, and they were not allowed to see the laptop screen. When the experimenter entered one password, the participant took control of the device and entered the guessed password via touch. The experimenter tried to input as the passwords as quickly and accurately as possible for both authentication methods, but participants' input time was not limited. After 20 attack trials with the first *authentication method*, another 20 trials were conducted with the other method. The order of the *authentication method* was counterbalanced across the participants. Experiments were carried out in a typical office lighting situation.

Apparatus: The smartphone used in the experiment was an Apple iPhone 5. To realize the REFLECTOR condition in real time (30 fps), the front camera image was scaled down to 120×150 pixels. The password input system and Reflector algorithm were coded in C++ with the face detection algorithm bundled in the OpenCV library [5]. The laptop that generated the random password was Apple's MacBook Pro (late 2012, 15-inch) and connected to the smartphone through a UDP connection. It also logged the success or failure of the PIN input and the duration of the password input on the smartphone. In the REFLECTOR condition, input was triggered by touching anywhere on the screen.

Results and Discussion

First, the success rate of the input performed by the experimenter was similarly high for both methods (REFLECTOR = 95.5 % and TOUCH = 97.5 %). However, for the input time, REFLECTOR showed a longer duration by 4.64 seconds ($\mu = 7.23$ s, $\sigma = 1.4$ s) than the TOUCH condition ($\mu = 2.59$ s, $\sigma = 0.53$ s). However, as these values include the time taken

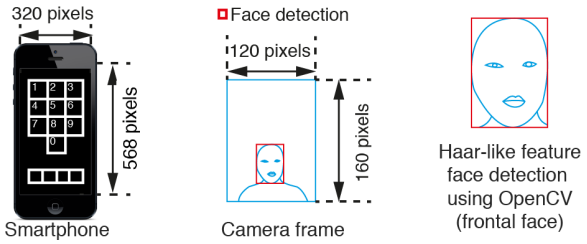

Figure 9. In Study 2, the onscreen reflection was simulated in real time through the front camera of a smartphone and the face detection algorithm from the `OpenCV` library.

to check the current password shown on the laptop, the actual input time is somewhat shorter than the measured value.

Next, for the hacking trials, the *authentication method* showed a significant main effect on *success rates* ($F_{1,9} = 252.3, p < 0.001$). REFLECTOR showed much a lower hacking success rate ($\mu = 3.5\%$, $\sigma = 5.2\%$) than the TOUCH condition ($\mu = 74\%$, $\sigma = 12.8\%$) (see Figure 10). In addition, the *authentication method* showed a significant main effect on participants' *input duration* ($F_{1,9} = 20.0, p = 0.002$). When the experimenter used REFLECTOR, participants showed a prolonged input duration ($\mu = 3.03$ s, $\sigma = 0.88$ s) than the TOUCH condition ($\mu = 1.82$ s, $\sigma = 0.47$ s).

The number of correctly guessed digits in REFLECTOR was 1.03 digits and in the TOUCH condition was 3.54 digits per trial. Most of the participants agreed that it was much more difficult to catch passwords when entered with REFLECTOR. This tendency also appeared in the participants' *input duration*, showing a prolonged duration of 1.2 seconds for thinking about and guessing the password. By contrast, some participants said there are still some clues remaining in the REFLECTOR condition that helped to guess the password entered. Especially, in this experiment, the experimenter performed the input by rotating the smartphone rather than moving. Therefore, if the participant closely examined the rotation of the smartphone, some input information could be obtained. For this reason, participants in the REFLECTOR condition may have been able to guess at least 3.5% of the input trials.

Obviously, the performance of our system depends on light conditions. To guarantee robust face detection, a moderate amount of ambient light is required. However, as most open spaces that are vulnerable to shoulder surfing attacks generally have sufficient light, this problem will not be pronounced in practical use. In addition, in this experiment, Reflector input

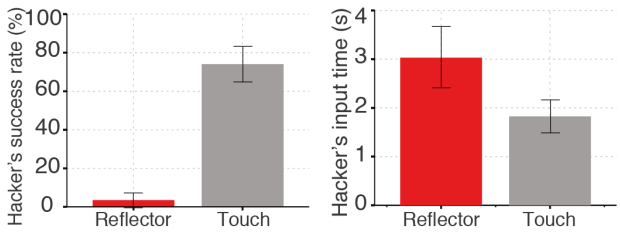

Figure 10. Due to the private nature of the reflection, the PIN input via REFLECTOR showed a hacking success rate 20 times lower than that of TOUCH: (a) success rate of malicious users, (b) input time of malicious users.

was performed by experts. Future study should explore how well the system works for the average user.

LIMITATIONS AND FUTURE WORK

Although Reflector allows precise pointing from a distance, its viability is highly influenced by some situational factors, such as lighting condition, screen reflectivity, and brightness of the image rendered onscreen. Our informal tests have shown that if the brightness of the onscreen image is less than 50% on a glossy monitor, the onscreen reflection was fully recognizable. In the future, it may be also possible to optimize content layout based on the user's visual perception model, to make it easier for the user to recognize the onscreen reflection [55, 51, 64].

In addition, there may be cases in which the three basic assumptions of the reflection model that we present are violated. For example, our model cannot handle cases in which the user is standing on the side of the screen. In addition, if multiple users need to use the system, additional usage scenarios and advanced system implementations are needed to handle all users' viewpoints.

The need for one calibration per user would not be a problem for spontaneous interaction in a typical situation where a single user is using a desktop PC. However, it is impractical to calibrate each user when there are many users interacting with a large public display. Currently, if the size of the targets used in the public display is sufficiently large, user-specific calibration can be roughly omitted by measuring absolute distance from the user's viewpoint to the screen using a depth-sensing camera. This technique allows the system to work with Equation 5 instead of Equation 6. However, in that case, the system must implement a function that can infer the user's eyedness.

In the performance evaluations in this study, we compared Reflector only with two direct input techniques: eye tracking and a touchscreen. However, several indirect pointing techniques (e.g., cursor-based interaction or gesture input through a depth camera[8]) are readily available and widespread for distant screens. A comparison between Reflector and such techniques remains future work.

CONCLUSION

Through this study, we have shown that the proposed system, Reflector, guarantees privacy of input and is robust to changes in distance between the user and screen. In Study 1, the performance of the eye tracker decreased dramatically as the distance between the user and screen increased. However, the performance reduction of the proposed system was slower. In Study 2, our proposed system enabled more private and safer input from shoulder surfing attacks than a touchscreen. Finally, we believe that our model can be implemented relatively simply on an all commodity glossy screens and can provide a high-fidelity direct-input user experience.

ACKNOWLEDGEMENTS

This research was supported the by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1C1B2002101).

REFERENCES

1. Pär-Anders Albinsson and Shumin Zhai. 2003. High precision touch screen interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 105–112.
2. Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2013. YouMove: enhancing movement training with an augmented reality mirror. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 311–320.
3. Amartya Banerjee, Jesse Burstyn, Audrey Girouard, and Roel Vertegaal. 2011. Pointable: an in-air pointing technique to manipulate out-of-reach targets on tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM, 11–20.
4. Florian Block, Daniel Wigdor, Brenda Caldwell Phillips, Michael S Horn, and Chia Shen. 2012. FlowBlocks: a multi-touch ui for crowd interaction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 497–508.
5. Gary Bradski and Adrian Kaehler. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".
6. Kjell Brunnström, Börje Andrén, Zacharias Konstantinides, and Lukas Nordström. 2007. Visual ergonomic aspects of computer displays: glossy screens and angular dependence. *Proc. of SPIE-IS&T Human Vision and Electronic Imaging XII* 6492 (2007).
7. Kjell Brunnström, Katarina Josefsson, and Börje Andrén. 2008. The effects of glossy screens on the acceptance of flat-panel displays. *Journal of the Society for Information Display* 16, 10 (2008), 1041–1049.
8. Marcus Carter, Joshua Newn, Eduardo Velloso, and Frank Vetere. 2015. Remote gaze and gesture tracking on the microsoft kinect: Investigating the role of feedback. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*. ACM, 167–176.
9. Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2527–2530.
10. Ishan Chatterjee, Robert Xiao, and Chris Harrison. 2015. Gaze+ Gesture: Expressive, Precise and Targeted Free-Space Interactions. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 131–138.
11. Michael Chau and Margrit Betke. 2005. Real time eye tracking and blink detection with usb cameras.
12. Alexander De Luca, Marian Harbach, Emanuel von Zezschwitz, Max-Emanuel Maurer, Bernhard Ewald Slawik, Heinrich Hussmann, and Matthew Smith. 2014. Now you see me, now you don't: protecting smartphone authentication from shoulder surfers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2937–2946.
13. Alexander De Luca, Emanuel Von Zezschwitz, and Heinrich Hußmann. 2009. Vibrapass: secure authentication based on shared lies. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 913–916.
14. Alexander De Luca, Emanuel Von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. 2013. Back-of-device authentication on smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2389–2398.
15. Paul Dietz and Darren Leigh. 2001. DiamondTouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. ACM, 219–226.
16. Sarah A Douglas, Arthur E Kirkpatrick, and I Scott MacKenzie. 1999. Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 215–222.
17. Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 457–466.
18. Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish S. Kulkarni, Shaun Kane, and Meredith Ringel Morris. 2017. Toward Everyday Gaze Input: Accuracy and Precision of Eye Tracking and Implications for Design. (May 2017).
19. Alexandre François, Elaine Kang, and Umberto Malesci. 2002. A handheld virtual mirror. In *ACM SIGGRAPH 2002 conference abstracts and applications*. ACM, 140–140.
20. Alexandre RJ François and E-YE Kang. 2003. A handheld mirror simulation. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, Vol. 2. IEEE, II–745.
21. Kaori Fujinami, Fahim Kawsar, and Tatsuo Nakajima. 2005. AwareMirror: a personalized display using a mirror. In *International Conference on Pervasive Computing*. Springer, 315–332.
22. Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. GripSense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 545–554.

23. Alix Goguey, Daniel Vogel, Fanny Chevalier, Thomas Pietrzak, Nicolas Roussel, and Géry Casiez. 2017. Leveraging finger identification to integrate multi-touch command selection and parameter manipulation. *International Journal of Human-Computer Studies* 99 (2017), 21–36.
24. Martin Hachet, Benoit Bossavit, Aurélie Cohé, and Jean-Baptiste de la Rivière. 2011. Toucheo: multitouch and stereo combined in a seamless workspace. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 587–592.
25. Jaehyun Han and Geehyuk Lee. 2015. Push-push: A drag-like operation overlapped with a page transition operation on touch interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. acm, 313–322.
26. Jefferson Y Han. 2005. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. ACM, 115–118.
27. Faizan Haque, Mathieu Nancel, and Daniel Vogel. 2015. Myopoint: Pointing and clicking using forearm mounted electromyography and inertial motion sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3653–3656.
28. Chris Harrison and Scott Hudson. 2012. Using shear as a supplemental two-dimensional input channel for rich touchscreen interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3149–3152.
29. Seongkook Heo, Jiseong Gu, and Geehyuk Lee. 2014. Expanding touch input vocabulary by using consecutive distant taps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2597–2606.
30. Luke Hespanhol and Martin Tomitsch. 2012. Designing for collective participation with media installations in public spaces. In *Proceedings of the 4th Media Architecture Biennale Conference: Participation*. ACM, 33–42.
31. Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss, and Andrew Wilson. 2012. HoloDesk: direct 3d interactions with a situated see-through display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2421–2430.
32. Ken Hinckley and Hyunyoung Song. 2011. Sensor synaesthesia: touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 801–810.
33. Thomas E Hutchinson, K Preston White, Worthy N Martin, Kelly C Reichert, and Lisa A Frey. 1989. Human-computer interaction using eye-gaze input. *IEEE Transactions on systems, man, and cybernetics* 19, 6 (1989), 1527–1534.
34. Petra Isenberg, Sheelegh Carpendale, Anastasia Bezerianos, Nathalie Henry, and Jean-Daniel Fekete. 2009. Coconuttrix: Collaborative retrofitting for information visualization. *IEEE Computer Graphics and Applications* 29, 5 (2009), 44–57.
35. Robert JK Jacob. 1995. Eye tracking in advanced interface design. *Virtual environments and advanced interface design* (1995), 258–288.
36. Mikkel R Jakobsen and Kasper Hornbæk. 2016. Negotiating for Space?: Collaborative Work Using a Wall Display with Mouse and Touch Input.. In *CHI*. 2050–2061.
37. Ricardo Jota, Miguel A Nacenta, Joaquim A Jorge, Sheelagh Carpendale, and Saul Greenberg. 2010. A comparison of ray pointing techniques for very large displays. In *Proceedings of Graphics Interface 2010*. Canadian Information Processing Society, 269–276.
38. Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1867–1874.
39. Mohamed Khamis, Florian Alt, Mariam Hassib, Emanuel von Zezschwitz, Regina Hasholzner, and Andreas Bulling. 2016. Gazetouchpass: Multimodal authentication using gaze and touch on mobile devices. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2156–2164.
40. Christian Lander, Sven Gehring, Antonio Krüger, Sebastian Boring, and Andreas Bulling. 2015. Gazeprojector: Accurate gaze estimation and seamless gaze interaction across multiple displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 395–404.
41. Byungjoo Lee and Hyunwoo Bang. 2013. A kinematic analysis of directional effects on mouse control. *Ergonomics* 56, 11 (2013), 1754–1765.
42. Byungjoo Lee and Hyunwoo Bang. 2015. A mouse with two optical sensors that eliminates coordinate disturbance during skilled strokes. *Human–Computer Interaction* 30, 2 (2015), 122–155.
43. Byungjoo Lee, Mathieu Nancel, and Antti Oulasvirta. 2016. AutoGain: Adapting Gain Functions by Optimizing Submovement Efficiency. *arXiv preprint arXiv:1611.08154* (2016).
44. Byungjoo Lee and Antti Oulasvirta. 2016. Modelling error rates in temporal pointing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1857–1868.
45. Wing Ho Andy Li and Hongbo Fu. 2012. Augmented reflection of reality. In *ACM SIGGRAPH 2012 Emerging Technologies*. ACM, 3.

46. Chengdong Lu and Douglas Frye. 1992. Mastering the machine: A comparison of the mouse and touch screen for children's use of computers. *Computer assisted learning* (1992), 417–427.
47. Yuexing Luo and Daniel Vogel. 2015. Pin-and-cross: A unimanual multitouch technique combining static touches with crossing selection. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 323–332.
48. Päivi Majaranta, I Scott MacKenzie, Anne Aula, and Kari-Jouko Räihä. 2006. Effects of feedback and dwell time on eye typing speed and accuracy. *Universal Access in the Information Society* 5, 2 (2006), 199–208.
49. Päivi Majaranta and Kari-Jouko Räihä. 2002. Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on Eye tracking research & applications*. ACM, 15–22.
50. Diego Martinez Plasencia, Florent Berthaut, Abhijit Karnik, and Sriram Subramanian. 2014. Through the combining glass. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 341–350.
51. Luana Micallef, Gregorio Palmas, Antti Oulasvirta, and Tino Weinkauf. 2017. Towards Perceptual Optimization of the Visual Design of Scatterplots. *IEEE Transactions on Visualization and Computer Graphics* 23, 6 (2017), 1588–1599.
52. Carlos H Morimoto and Marcio RM Mimica. 2005. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding* 98, 1 (2005), 4–24.
53. Jurriaan D Mulder and BR Boscker. 2004. A modular system for collaborative desktop vr/ar with a shared workspace. In *Virtual Reality, 2004. Proceedings. IEEE*. IEEE, 75–280.
54. Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015. Mid-air pointing on ultra-walls. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 5 (2015), 21.
55. Antti Oulasvirta. 2017. User interface design with combinatorial optimization. *Computer* 50, 1 (2017), 40–47.
56. Volker Roth, Kai Richter, and Rene Freidinger. 2004. A PIN-entry method resilient against shoulder surfing. In *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 236–245.
57. Hideaki Sato, Itaru Kitahara, and Yuichi Ohta. 2009. MR-mirror: a complex of real and virtual mirrors. In *International Conference on Virtual and Mixed Reality*. Springer, 482–491.
58. Florian Schaub, Ruben Deyhle, and Michael Weber. 2012. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proceedings of the 11th international conference on mobile and ubiquitous multimedia*. ACM, 13.
59. Christopher Schmandt. 1983. Spatial input/display correspondence in a stereoscopic computer graphic work station. In *ACM SIGGRAPH Computer Graphics*, Vol. 17. ACM, 253–261.
60. Ju Shen, SC S Cheung, and Jian Zhao. 2012. Virtual mirror by fusing multiple RGB-D cameras. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE, 1–9.
61. Ju Shen, Po-Chang Su, Sen-ching Samson Cheung, and Jian Zhao. 2013. Virtual mirror rendering with stationary rgb-d cameras and stored 3-d background. *IEEE Transactions on Image Processing* 22, 9 (2013), 3433–3448.
62. Matthias Straka, Stefan Hauswiesner, Matthias Rüther, and Horst Bischof. 2011. A free-viewpoint virtual mirror with marker-less user interaction. In *Scandinavian Conference on Image Analysis*. Springer, 635–645.
63. Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. 2013. Appearance-based gaze estimation using visual saliency. *IEEE transactions on pattern analysis and machine intelligence* 35, 2 (2013), 329–341.
64. Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and explore with a layout optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. ACM, 543–555.
65. Keita Ushida, Yu Tanaka, Takeshi Naemura, and Hiroshi Harashima. 2002. i-mirror: An Interaction/Information Environment Based on a Mirror Metaphor Aiming to Install into Our Life Space. In *Proceedings of the 12th International Conference on Artificial Reality and Telexistence (ICAT2002)*. 113–118.
66. Mélodie Vidal, Andreas Bulling, and Hans Gellersen. 2013. Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 439–448.
67. Daniel Vogel and Ravin Balakrishnan. 2004. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. ACM, 137–146.
68. Daniel Vogel and Ravin Balakrishnan. 2005. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. ACM, 33–42.
69. Emanuel Von Zezschwitz, Alexander De Luca, Bruno Brunkow, and Heinrich Hussmann. 2015. Swipin: Fast and secure pin-entry on smartphones. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1403–1406.

70. Andrew D Wilson. 2004. TouchLight: an imaging touch screen and display for gesture-based interaction. In *Proceedings of the 6th international conference on Multimodal interfaces*. ACM, 69–76.

71. Xuan Zhang and I Scott MacKenzie. 2007. Evaluating eye tracking with ISO 9241-part 9. In *International Conference on Human-Computer Interaction*. Springer, 779–788.