Ghost Fingers: A Hybrid Approach to the Interaction with Remote Displays

Seung Wook Kim
Hewlett-Packard
950 West Maude Ave.
Sunnyvale, CA 94085 USA
seungwook.k@gmail.com

Stefan Marti
Hewlett-Packard
950 West Maude Ave.
Sunnyvale, CA 94085 USA
stefan.marti@hp.com

CHI'12, May 5–10, 2012, Austin, Texas, USA.
ACM 978-1-4503-1016-1/12/05.

Abstract
In this paper, we describe a novel interaction method called Ghost Fingers, which enables efficient and intuitive switching between keyboard and multi-touch input on systems where the display is out of arm's reach. In addition, Ghost Fingers provides a translucent real-time visualization of the fingers and hands on the remote display, creating a closed interaction loop that enables direct manipulation even on remote displays. Our solution includes a wireless keyboard with attached imaging sensor that is used to both determine the position of the user's hand and fingers, and to provide a real-time translucent overlay of hand and fingers over the remote UI.

Author Keywords
Multi-touch interaction; remote finger and hand visualization; remote multi-touch; Ghost Fingers

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces - Graphical User Interfaces (GUI);

General Terms
Design, Human Factors

Introduction

More and more digital content is being consumed at home with devices that traditionally had not been considered computers. Integrated with digital network technologies, televisions are particularly attractive to users who intend to search the Internet and download media for instant watching on increasingly large screens, while still sitting on their couches in the most comfortable posture possible. However, such a "lean-back" mode of interaction naturally places the display at a distance from its user, where multi-touch interaction methods—such as pinch to zoom and two-finger rotate to rotate objects—cannot be used. Existing solutions to this problem utilize alternative natural input methods such as gestures or voice, but they do not support direct manipulation of display content.

This paper describes a novel interaction method that combines two distinct input modalities: text input (using a common physical keyboard that allows touch typing) and remote multi-touch input, with minimal effort for switching between the two. In addition, our system enables a convenient lean-back solution by visually providing users with the location of their fingers with regards to the UI on the display. This method delivers convenient multi-touch input for systems with a physical keyboard where the display is out of arm's reach, such as smart TVs. It is also useful for a multi-touch computing device with a keyboard physically separated from the touch display, such as advanced desktops, laptops, and netbooks, as well as tablets with slider keyboards. Conventional personal computers that do not support multi-touch input can also benefit from this method. Ultimately, our work offers a method that enables "virtual" multi-touch input without requiring the user to touch the display surface directly.

Related Work

Although touch-screen-like interactions with a TV remote control like device have been studied (e.g., [1]), our focus is specifically to enable multi-touch interaction on displays out of arm's reach. Several commercial solutions for this problem are available, among them: wireless keyboards with dedicated touch pad areas (many manufacturers); dedicated handheld touch pads (e.g., ZRRO TeleTouch devices [8]); remote gesture interfaces (e.g., Microsoft Kinect [2]).

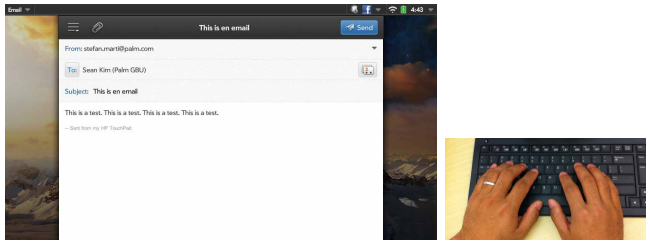
Our system improves on existing solutions such as wireless keyboards with built-in touch pads in two ways: first, these solutions keep keyboard and touch area separate, which requires the user to shift hand position when switching between text input and touch input mode. With our system, the user does not have to move her hands at all when switching between text input mode and multi-touch mode, meaning that zero hand travel time required. Second, these systems generally visualize the position of only one finger on the touch pad, most often in the shape of a mouse pointer. With our system, instead, the user has a clear indication of where all the fingers are with regards to the remote multi-touch display due to the Ghost Finger visualization. This visualization creates a closed–loop interaction system for multi-touch which enables the advantages of multi-touch interfaces on displays out of arm's reach.

Compared to dedicated handheld touchpads (e.g., ZRRO TeleTouch devices [8]), our system gives a full preview (outline) of hand and fingers, not just the hover and touch points. In addition, our system integrates multi-touch interaction directly on a physical keyboard, allowing the user to switch easily between

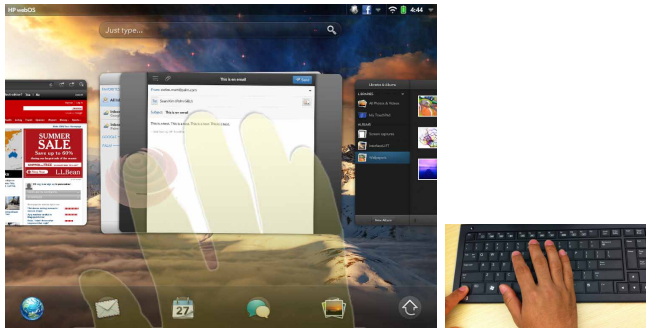
touch typing on the keyboard and multi-touch input on the remote display.

Other previously suggested solutions include remote gesture systems that detect hands and other body parts remotely (e.g., Kinect based systems [2]). These systems, however, are potentially tiring during extended periods of use since they require holding hands or arms in the air for the interaction. With our system, instead, the user's fingers can be kept in a resting position on the keyboard at all times. Also, such systems suffer from parallax issues, because the user's fingers/hands are at a distance from the display. In our system, the outline of the user's fingers is on the same optical plane as the UI content, which enables precise multi-touch manipulation on remote displays.

As for visualizing the user's hand and fingers directly on a UI, work by Patrick Baudisch et al. on behind-device interaction (e.g., LucidTouch [7]) uses visualizations of the hands and fingers, but is neither combining a physical keyboard with it, nor using it for remote display interaction. *TactaPad* [4] also presents a filtered video image of the user's hands on the interaction area. Unlike our system, however, the TactaPad is not a direct-touch device, and does not combine the interaction with a physical keyboard. In general, visualizing a user's hands (and arms and more) on the UI with real-time video is a known theme in CSCW (e.g., VideoDraw [6] and VideoArms [5] use video representations of users' hands), although intended to represent a remote user, not the user's own hands.



(a) Text input mode (writing email)



(b) Multi-touch input mode (manipulating card GUI elements)

Figure 1: The illustration (b) shows that the user can track her own hand and fingers (right) with high fidelity on the GUI (left), and therefore easily perform very specific multi-touch actions such as selecting a middle card from a stack. The user sees her hands/fingers continuously, even without (and before) manipulating actual UI content (e.g., pressing a button).

System Overview

Our prototype system works as follows: A standard QWERTY physical keyboard is equipped with an image sensor (e.g., webcam), which is able to detect the position of one or two hands over the keys. In text-input mode (Figure 1a), the keyboard works like a normal keyboard. However, when the user presses a designated key (e.g., CTRL key), the system switches

to "multi-touch mode" (Figure 1b), which allows gestures such as pinch to zoom and two-finger rotation. In this mode, the sensor detects the user's fingers on the keyboard. At the same time (in real time), a highly transparent image of the user's hands (including fingers) is displayed on the remote display (e.g., overlaid over the UI), resulting in "ghost fingers" or hand outlines. These ghost fingers or hand outlines allow the user to easily manipulate the UI on the display, e.g., pressing an icon on the screen. The physical keyboard becomes the proxy input surface for the multi-touch display, and the physical keys on the keyboard are used to detect touch events. Swiping gestures can also be detected (e.g., flicking UI content), by the user gliding over keys, slightly depressing them.

Figure 2. System overview (left) and image sensor (right)

Implementation

As shown in Figure 2, we built a low-profile overhead image sensor by modifying a typical webcam and attaching it on a Bluetooth keyboard. We added a fish-eye lens to the sensor in order to achieve a wide field of view that can cover the entire layout of keyboard. As a tradeoff, the sensor's output image data suffered from radial distortion in addition to perspective distortion. Therefore, we first calibrated the raw image data with homography and undistortion matrices using OpenCV library [2]. Figure 3 depicts both row image with significant distortion and the processed image after calibration.

Figure 3: Raw image data from the sensor showing significant distortion (left); undistorted and rectified hand image after calibration process

The calibrated image data is subject to additional vision processing, as it still includes unnecessary image elements in the background. In order to simplify this process, the system stores the original keyboard image (calibrated, without the user's hand) when it is initiated, and subtracts it from each calibrated image frame to achieve a clear image of hand outline. The system superimposes this final output image on the graphical user interfaces and their content. We implemented this rendering process with OpenGL in order to acquire a translucent effect on the hand image (ghost fingers). Figure 4 illustrates the process and the final output.

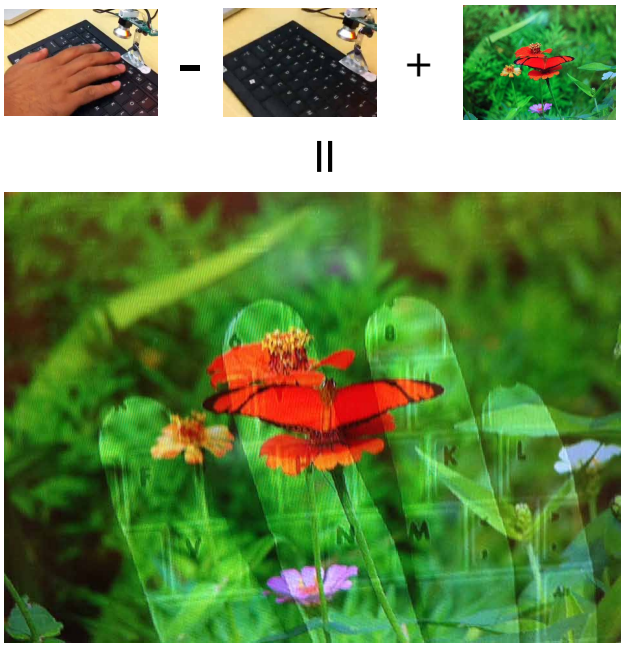


Figure 4. [Top] Input sources = (calibrated hand image – calibrated background image) * transparency + GUI content; [Bottom] Final output image (Ghost Fingers) rendered by the prototype system; note that the "bleeding" of the keyboard through the fingers is an artifact of the initial prototype, and will be eliminated in later iterations

The rendered hand image on the screen provides the user with direct and immediate reference of the actual hand position relative to the screen. We expect the user to perceive Ghost Fingers as being stretched from his/her body for manipulating content on the screen. In order to complete this interaction loop, our system maps the layout of keys of the physical keyboard onto the screen coordinates (without actually rendering a real of virtual keyboard). When the user presses one of the keys, the system takes a subset of pixels (in the hand image) overlapping with the screen sector mapped from the pressed key (Figure 5). Then the system runs a set of image processes to a) detect the blob and b) determine the position of the fingertip, all within the taken subset of pixels (Figure 6). With this method, the system can detect finger presses with higher resolution than just key sizes, and is also able to detect touch targets in between keys. Together with the key press events, the detected fingertip coordinates create a single or multi-touch event.

Our system works best with common touch UIs, similar to phone and tablet UIs, where touch targets are larger than on systems based on mouse cursor interaction, so the thickness of the fingers is not an issue.

Since the surface area of the keyboard may not be equivalent to the size and shape of the display, the keyboard area is stretched to the display area.

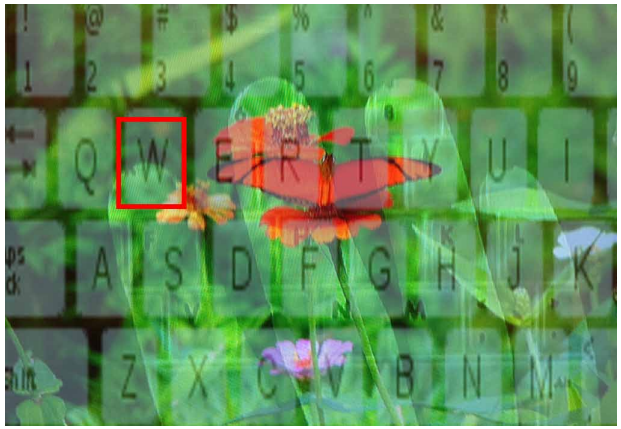


Figure 5. Mapping keyboard layout into screen coordinates (in this image, the keyboard is visible to illustrate the process; it is not visible to the user normally)

Figure 6. Blob detection and fingertip determination from the hand image within the pressed key region

Future Work

In addition to the detection of touch events like described above, a swipe gesture over a physical keyboard can be identified in several ways. The simplest method is to detect multiple key presses that are both spatially and temporally consecutive. This approach may benefit from the hardware design of the keyboard that would allow a key press only with a subtle amount of force from the user's finger(s), such as "chiclet" type keyboards with rounded keys, low-profile, and low-travel. An informal evaluation shows that these types of keyboards indeed allow for a natural and convenient finger swiping experience. A more hardware intensive solution is to integrate a capacitive sensor with the actual keyboard. Another method would be to measure the optical field flow within and around the pressed key in the finger tip image.

Although a physical keyboard is still considered the most reliable and efficient method of text input, it is possible to replace it with other types of sensors to improve the multi-touch modality of the Ghost Fingers interaction method. This may include capacitive, electronic-field, ultrasound, or IR-based sensors, which may be capable of generating a hand and finger image (e.g., a "heat map") from the raw data. Additional image processing steps (e.g., noise filtering, edge detection) would be needed to generate a clean visualization of the hand.

Variations in the graphical visualization of Ghost Fingers will enable novel interaction scenarios. For example, by replacing the typical pointing cursor with actual finger tips, our solution would enable a highly intuitive interaction with GUI elements even on conventional window-based platforms.

References

[1] Choi, S., Han, J., Lee, G., Lee, N., Lee, W. (2011). RemoteTouch: Touch-Screen-like interaction in the TV viewing environment. CHI 2011, pp. 393–401.

[2] Kinect, http://www.xbox.com/en-US/kinect

[3] OpenCV, http://opencv.willowgarage.com/

[4] TactaPad http://www.tactiva.com/tactapad.html

[5] Tang, A., Neustaedter, C., Greenberg, S. (2004). VideoArms: Supporting Remote Embodiment in Groupware. Video Proceedings of CSCW '04.

[6] Tang, J. C. and Minneman, S. L. (1990). VideoDraw: a video interface for collaborative drawing. CHI '90, pp. 313-320.

[7] Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., Shen, C. (2007). LucidTouch: A See-Through Mobile Device. UIST 2007, pp. 269–278.

[8] ZRRO, http://www.zrro.com/