A System for Hybrid Vision- and Sound-Based Interaction with Distal and Proximal Targets on Wall-Sized, High-Resolution Tiled Displays

Daniel Stødle, John Markus Bjørndalen, and Otto J. Anshus

Dept. of Computer Science, University of Tromsø, N-9037 Tromsø, Norway
{daniels,jmb,otto}@cs.uit.no

Abstract. When interacting with wall-sized, high-resolution tiled displays, users typically stand or move in front of it rather than sit at fixed locations. Using a mouse to interact can be inconvenient in this context, as it must be carried around and often requires a surface to be used. Even for devices that work in mid-air, accuracy when trying to hit small or distal targets becomes an issue. Ideally, the user should not need devices to interact with applications on the display wall. We have developed a hybrid vision- and sound-based system for device-free interaction with software running on a 7x4 tile 220-inch display wall. The system comprises three components that together enable interaction with both distal and proximal targets: (i) A camera determines the direction in which a user is pointing, allowing distal targets to be selected. The direction is determined using edge detection followed by applying the Hough transform. (ii) Using four microphones, a user double-snapping his fingers is detected and located, before the selected target is moved to the location of the snap. This is implemented using correlation and multilateration. (iii) 16 cameras detect objects (fingers, hands) in front of the display wall. The 1D positions of detected objects are then used to triangulate object positions, enabling touch-free multi-point interaction with proximal content. The system is used on the display wall in three contexts to (i) move and interact with windows from a traditional desktop interface, (ii) interact with a whiteboard-style application, and (iii) play two games.

Keywords: Vision- and sound-based interaction, large displays.

1 Introduction

Wall-sized, high-resolution displays are typically built by tiling displays or projectors in a grid. The displays or projectors are driven by a cluster of computers cooperating to produce a combined image covering the display wall, with existing display walls ranging in resolution from 10 to 100 Megapixels [8,16]. Our display wall combines 7x4 projectors back-projecting onto a non-rigid canvas to create a 220-inch display with a resolution of 7168x3072 pixels.

Display walls invite users to stand and move in front of them. To use input devices like mice and keyboards in this context, users must carry them around.

M. Lew et al. (Eds.): HCI 2007, LNCS 4796, pp. 59–68, 2007.

Mice often require flat surfaces to be used, and even for "gyro-mice" and similar devices that don't have this requirement, the accuracy when trying to hit small or distal targets becomes an issue. We believe that ideally, users should not need devices to interact with applications running on the display wall.

We have developed a hybrid vision- and sound-based system that enables device-free interaction with software running on our display wall. The system analyzes and combines input from three main components to enable both distal and proximal interaction. The three components perform the following tasks, respectively: (i) Determine the direction in which a user is pointing his arm, (ii) determine the location at which a user snaps his fingers, and (iii) determine the location of objects (usually fingers or hands) in front of the wall. Figure 1 shows the system in use for playing two games.

Fig. 1. The system in use for playing Quake 3 Arena and Homeworld

Each component contributes to a different part of the device-free workflow on the display wall. The first component lets a user select a distal target by pointing his arm towards it. A camera situated in the ceiling behind the user captures video. Each frame of the video is run through an edge detector, before the Hough transform is applied to determine the angle and position of the prevalent line segments visible in the video. Combining this with knowledge about what is currently visible on the display wall lets the component determine which target the user is attempting to select.

The second component lets the user bring the selected, distal target closer by double-snapping his fingers. Four microphones arranged in a rectangular fashion near the display wall stream audio to a computer. The computer correlates the audio samples with a template audio clip. When the user's snap is detected from at least three microphones, the snap's position can be determined using multilateration.

The third component lets the user interact with proximal targets. 16 cameras arranged in a row on the floor below the display wall's canvas stream data to 8 computers. When an object is visible from at least three cameras as it intersects a plane parallel to the display wall's canvas, its position can be determined using triangulation. Multiple objects can be detected and positioned simultaneously, enabling touch-free multi-point and multi-user interaction. We refer to it as touch-free as the user does not actually have to touch the display wall in order to interact with it. This is important in our context, as the display wall's canvas is flexible, and thus prone to perturbations when users touch it.

The main contribution of this paper is a hybrid vision- and sound-based system for interacting with both distal and proximal targets. We detail the system's implementation and show its use in three different contexts on a display wall:

(i) A traditional desktop interface, (ii) an experimental whiteboard-style application, and (iii) two games.

2 Related Work

Interacting with distal targets on very large displays is an important problem in HCI, and much previous work exists. In [18], the authors present a system that allows distant freehand pointing for interacting with wall-sized displays. Our system does not support pointing from afar, but lets users select distal targets while standing close to the display wall. Our system detects the angle at which a user points his arm, while the authors of [18] use a commercial system that requires the user to wear markers. There are numerous other techniques for reaching distal targets described in the literature; they include drag-and-pop [1], the Vacuum [2] and the Frisbee [7]. These techniques generally work by temporarily bringing distal targets closer. Our approach is complementary in this regard, as distal targets are moved semi-permanently (that is, until the user decides to move them again). The "tablecloth" is an entirely different approach that lets the user scroll the desktop much as he would scroll a window [13].

In [14], untagged audio is used for interacting with a 3D interface connected to a media player. The system recognizes loud sounds above a dynamic threshold, such as snapping fingers, and in turn uses this to determine the position of the sound. Their work focuses on creating 3D interfaces, by creating and placing virtual "buttons" in space. Rather than creating buttons with fixed locations, we utilize the user's actual position to bring distal targets closer to the user. In [6], the authors propose using continuous vocal sounds for controlling an interface, rather than interpreting spoken commands. They do not attempt to determine the user's physical location, while one key capability of the snap-detecting component of our system is that it allows applications to leverage this information. Using vocal sounds to control the position of a cursor is proposed in [9]. This work could conceivably be used to act on distal targets, and indeed one aspect of moving the cursor often includes moving windows. However, no attempt is made at determining the user's location and using this information to improve interaction.

Sound source localization is much used in the field of robotic navigation. In [17], a system is described whereby 8 microphones are used to reliably locate sounds in a 3D environment using cross-correlation and time-delay. The technique used by the snap-detecting component is similar in that it estimates the time-delay between incoming snaps, but differs in that it uses the located sound for interacting with a user interface, rather than interacting with a robot. Our component only attempts to detect snaps, whereas the system in [17] does not discriminate between different sounds. Rather, it is used to locate all "interesting" sound sources, in order to focus the robot's attention towards the sound source.

Much research has been done on systems for supporting multi-touch and multi-point interaction. The Diamondtouch [3] tabletop is one approach, where

the position of touches is detected using electric capacitance. Other technologies include [5], where infrared light is projected into a canvas and internally reflected. The light escapes the canvas at points where the user is touching, which can be detected using a camera. Our system is based on detecting the presence of objects directly using cameras, and does not require the user to actually touch the display wall's canvas. In [10], the author presents a camera-based solution to detecting and positioning objects in front of a whiteboard. They use custom cameras with on-chip processing to perform object recognition, while we take a parallel and commodity-based approach with 16 cheap cameras connected to 8 computers.

3 Design

The hybrid vision- and sound-based system is comprised of three components and a custom event delivery system. The three components are (i) arm-angle, (ii) snap-detect and (iii) object-locator. Each component is responsible for detecting different user actions, with the event system providing communication between the components and end-user applications[1]. Figure 2 illustrates the overall design.

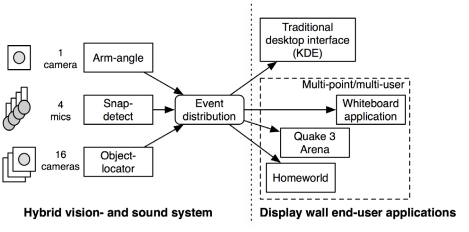

Fig. 2. The system design with three components and event distribution

The purpose of the arm-angle and snap-detect components is to let the user select and access distal targets, while the object-locator enables the user to interact with proximal targets using single- or multi-point interaction. Figure 4 illustrates an example scenario where each component is used in turn to select a distal target, move it closer to the user and then interact with it.

Fig. 3. 16 cameras along the floor enable object detection in front of the display wall

The arm-angle component processes images streamed from a single camera, with the purpose of identifying the angle at which a user is pointing. The camera is mounted in the ceiling, looking at the backs of users interacting with the display wall. The component delivers events to end-user applications, which interpret them according to their needs and current state.

The snap-detect component performs signal processing on audio from four microphones. The microphones are arranged in a rectangle in front of the display wall, with two microphones mounted near the ceiling at op-

[1] The event system is outside the scope of this paper.

posite ends of the canvas, and the other two near the floor. The component's goal is to detect a user snapping his fingers or clapping his hands in at least three of the four audio streams, allowing the signal's origin in 2D space to be determined using multilateration.

Multilateration uses the difference between the time at which snaps are detected by the different microphones to determine possible locations where the user snapped his finger. With a fixed sample rate, it is possible to determine the approximate, relative distance the sound has travelled from the time it was detected by a given microphone until it was detected by another. The resulting points lie on a hyperbolic curve with focus point at the given microphone's position. For many microphones, the intersections of the resulting curves can be used to determine the location of the user in 2D. For each positioned snap, an event containing the snap's 2D location and strength is delivered to end-user applications.

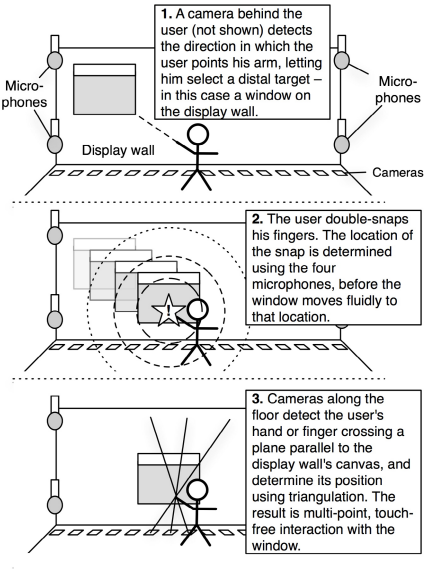

Fig. 4. The user selects a window, double-snaps to bring it closer, and interacts using the touch-free interface

The object-locator component uses 16 cameras to detect objects intersecting a plane parallel to the display wall's canvas, with typical objects being a user's fingers or hands. The cameras are connected pairwise to 8 computers, and mounted in a row along the floor looking at the ceiling (Figure 3). When three or more cameras see the same object, its position can be determined using triangulation. The component can detect several objects simultaneously. If the end-user application supports it, both multi-user and multi-point interaction is possible.

4 Implementation

The arm-angle component, implemented in C, uses a Canon VC-C4R pan-tilt-zoom camera connected to a framegrabber-card on a computer running Linux. The component captures frames at 8-9 frames per second (FPS). Each frame has a resolution of 720x540 pixels and is scaled down to half-size and converted from color to grayscale. Then the Sobel edge detector is used to locate edges in the image, before the equations of lines appearing in the image are determined using the Hough transform [4]. End-user applications receive events from the arm-angle component, which they use to determine the targets that the user is trying to select. For the traditional desktop interface, this is done by determining

the bounding rectangles of currently visible windows, and then intersect each window's bounds with the line indicating the user's pointing direction. Since the Hough-transform typically yields several potential lines for the arm and other straight edges, each line votes for the closest window it intersects with. A red square over the selected window highlights it to the user. The traditional desktop interface on the display wall is created using Xvnc [12,11], with each tile of the display wall showing its respective part of the entire, high-resolution desktop.

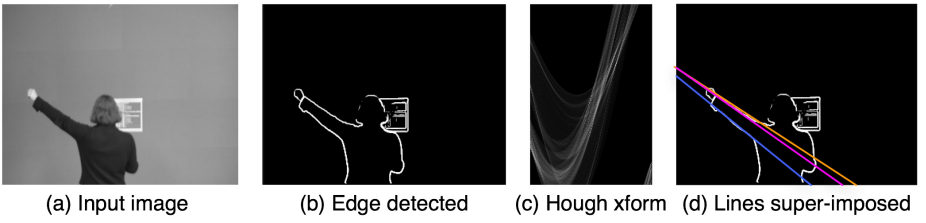

Fig. 5. The arm-angle image processing steps. (a) The input image, (b) the image after edge detection, (c) output from the Hough-transform, (d) the edge-detected image with lines extracted from the Hough-transform super-imposed.

The snap-detect component uses four microphones connected to a mixer. The mixer feeds the audio to a PowerMac G5, which uses a Hammerfall HDSP 9652 sound card to capture 4-channel audio samples at a rate of 48000 Hz. Samples from each channel are correlated with a template sound of a user snapping his fingers. When the correlation for a given channel exceeds an experimentally determined threshold, a snap is detected in that channel. The snap-detect component records a timestamp (essentially a sample counter) for each channel the snap is detected in. When a snap is detected in at least three channels, the resulting timestamps can be used to determine the user's location using the difference in time between when the snap is detected by the first microphone, and when it is detected by the remaining two or three microphones (multilateration). The location is determined by the intersection of conics created from different pairs of microphones, illustrated in Figure 7. For the desktop interface on the display wall, the selected window is moved to the location of the snap over a period of half

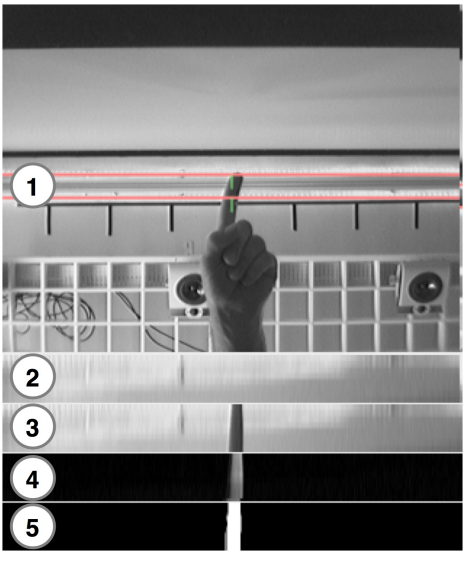

Fig. 6. (1) Image from camera. (2) Current background image. (3) Area of interest from current camera image. (4) The result by subtracting (3) from (2). (5) Result after thresholding the image.

a second. Changes to existing applications are not necessary to allow window movement by snapping fingers.

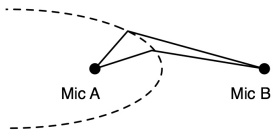

Fig. 7. The possible positions given the time difference between mic A and B are all located on the conic section.

The object-locator component is implemented in Objective-C, and consists of two modules: An image processing module, and a triangulation module. The image processing module runs in parallel on 8 Mac minis. Each Mac mini captures streaming video from two Unibrain Fire-i FireWire cameras in 640x480 grayscale at 30 FPS. Figure 6 illustrates the image processing done for a single camera. When the image processing module on a Mac mini starts up, it sets the first grabbed image to be the background image; the background is subsequently updated dynamically to deal with changing light conditions. For each new image captured, a narrow region of interest is isolated, before the background is subtracted from it. The result is then thresholded, yielding one-dimensional object positions and radii wherever pixel values exceed the dynamic threshold (the radius is set to the number of successive pixels above the threshold divided by two). The one-dimensional positions and radii are gathered by the triangulation module. For each camera, the triangulation module creates line segments that start at the camera's position, and pass through the center of each object detected by that camera. These lines are then intersected with the lines generated for the two cameras to the immediate left and right of the current camera. The resulting intersections are examined, and if two or more intersection points from three different cameras lie sufficiently close, an object is detected, as illustrated in Figure 8. The traditional desktop interface uses the 2D positions reported by the object-locator to control the cursor, and uses the radius to determine if the user wants to click.

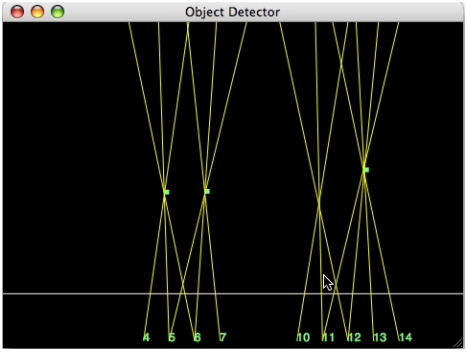

Fig. 8. The object-locator identifying three of four objects and their position

5 Early Deployment

The hybrid vision- and sound-based interface has been deployed in three different contexts: (i) A traditional desktop interface running on the display wall, (ii) a prototype whiteboard-style application supporting multiple users, and (iii) two previously commercial, but now open-source games (Quake 3 Arena and Homeworld). When used with a traditional desktop interface, the system enables

one user to select distal windows on the display wall, and move them closer by double-snapping his fingers. When the window is within reach, the user can interact with it using the touch-free interface. The cursor tracks the position of the user's hand or finger, and a click or drag can be invoked by tilting the hand (making it flat), as illustrated in Figure 9. Its use in the whiteboard-application is similar, where the distal objects drawn on screen can be brought closer by pointing at them and then double-snapping. In addition, the whiteboard brings up a tool palette at the user's location when a user single-snaps, allowing new objects to be added. The touch-free interface is used to support multi-user and multi-point interaction, for instance allowing users to resize objects by varying the distance between their hands.

The final context in which the system has been used is to play two games, Quake 3 Arena and Homeworld, further detailed in [15]. In this context, we only utilize the object-locator component, and to some extent the snap-detect component (people not playing the game can make the players fire their weapons in Quake 3 Arena by snapping their fingers, for instance). The touch-free interface provided by the object-locator enables users to play the two games using gestures only, rather than using traditional mouse/keyboard-based input.

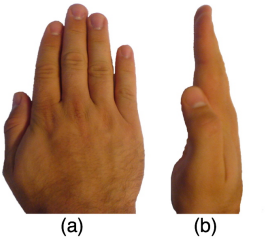

Fig. 9. (a) A flat and (b) vertical hand

6 Discussion

One principle employed throughout the design of the system is the following: Rather than identify what objects are, it is more important to identify where the objects are. The system at present does not distinguish between an arm pointing to give a direction, or a stick being used to do the same; nor does the system distinguish a snap from a clap or snap-like sounds made by mechanical clickers. This principle is also used to realize the touch-free interface - users can interact using their fingers, hands, arms, pens or even by moving their head through the (invisible) plane in front of the display wall.

This principle is not without issues, however, as false positives can occur for all of the components used in the system. For the arm-angle component, one issue is that the content currently on the display wall interferes with the edge-detection and the following Hough-transform to produce results that do not reflect the user's intentions. Another issue is that the resolution offered by the arm-angle component is too coarse to be used for selecting very small targets, such as icons. Techniques like drag-and-pop or the vacuum may be better suited for picking out small targets [1,2]. For the object-locator, the fact that it does not recognize what objects are means that it has no way of distinguishing several users from a single user interacting with several fingers or both hands. Although our ideal is to provide device-free interaction with the display wall for multiple, simultaneous users, the only component of the system that currently supports

multiple simultaneous users is the object-locator. We are currently working on ways to support multiple users with the arm-angle and snap-detect components as well. We are also investigating the accuracy of the three components, to better characterize their performance.

7 Conclusion

This paper has presented a system combining techniques from computer vision and signal processing to support device-free interaction with applications running on high-resolution, wall-sized displays. The system consists of three components utilizing in total 17 cameras and 4 microphones. It enables a user to select distal targets by pointing at them, bring the targets closer by double-snapping his fingers and finally interact with them through the use of a touch-free, multi-point interface. The system does not require the user to wear gloves or use other external devices. Instead, the system has been designed so as not to care exactly *what* a detected object is, but rather *where* the object – whatever it may be – is located. This in turn means that the system does not attempt to tell different users apart, for either of the three components. The interface has been deployed in three different contexts on the display wall: (i) One user at a time can interact with a traditional desktop interface, (ii) several users can interact simultaneously with a custom whiteboard-style application, with the caveat that the distal target selector only works for the user positioned near the center of the display wall, and (iii) up to three persons may play the games Quake 3 Arena and Homeworld simultaneously.

Acknowledgements

The authors thank Espen S. Johnsen and Tor-Magne Stien Hagen for their discussions, as well as the technical staff at the CS department at the University of Tromsø. This work has been supported by the Norwegian Research Council, projects No. 159936/V30, SHARE - A Distributed Shared Virtual Desktop for Simple, Scalable and Robust Resource Sharing across Computer, Storage and Display Devices, and No. 155550/420 - Display Wall with Compute Cluster.

References

1. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., Zierlinger, A.: Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems. In: Proceedings of Interact 2003, pp. 57–64 (2003)
2. Bezerianos, A., Balakrishnan, R.: The vacuum: facilitating the manipulation of distant objects. In: CHI 2005. Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 361–370. ACM Press, New York (2005)
3. Dietz, P., Leigh, D.: DiamondTouch: a multi-user touch technology. In: UIST 2001. Proceedings of the 14th annual ACM symposium on User interface software and technology, pp. 219–226. ACM Press, New York (2001)

4. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Commun. ACM 15(1), 11–15 (1972)
5. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: UIST 2005. Proceedings of the 18th annual ACM symposium on User interface software and technology, pp. 115–118. ACM Press, New York (2005)
6. Igarashi, T., Hughes, J.F.: Voice as sound: using non-verbal voice input for interactive control. In: UIST 2001. Proceedings of the 14th annual ACM symposium on User interface software and technology, pp. 155–156. ACM Press, New York (2001)
7. Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N., Kurtenbach, G.: A remote control interface for large displays. In: UIST 2004. Proceedings of the 17th annual ACM symposium on User interface software and technology, pp. 127–136. ACM Press, New York (2004)
8. Li, K., Chen, H., Chen, Y., Clark, D.W., Cook, P., Damianakis, S., Essl, G., Finkelstein, A., Funkhouser, T., Housel, T., Klein, A., Liu, Z., Praun, E., Samanta, R., Shedd, B., Singh, J.P., Tzanetakis, G., Zheng, J.: Building and Using A Scalable Display Wall System. IEEE Comput. Graph. Appl. 20(4), 29–37 (2000)
9. Mihara, Y., Shibayama, E., Takahashi, S.: The migratory cursor: accurate speech-based cursor movement by moving multiple ghost cursors using non-verbal vocalizations. In: Assets 2005. Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility, pp. 76–83. ACM Press, New York (2005)
10. Gerald, D.: A camera-based input device for large interactive displays. IEEE Computer Graphics and Applications 25(4), 52–57 (2005)
11. RealVNC, Ltd. VNC for Unix 4.0. `http://www.realvnc.com/`
12. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual Network Computing. IEEE Internet Computing 2(1), 33–38 (1998)
13. Robertson, G., Czerwinski, M., Baudisch, P., Meyers, B., Robbins, D., Smith, G., Tan, D.: The large-display user experience. IEEE Comput. Graph. Appl. 25(4), 44–51 (2005)
14. Scott, J., Dragovic, B.: Audio Location: Accurate Low-Cost Location Sensing. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 1–18. Springer, Heidelberg (2005)
15. Stødle, D., Hagen, T.-M.S., Bjørndalen, J.M., Anshus, O.J.: Gesture-based, touch-free multi-user gaming on wall-sized, high-resolution tiled displays. In: Proceedings of the 4th Intl. Symposium on Pervasive Gaming Applications, PerGames 2007, pp. 75–83 (June 2007)
16. Stolk, B., Wielinga, P.: Building a 100 Mpixel graphics device for the OptIPuter. Future Gener. Comput. Syst. 22(8), 972–975 (2006)
17. Valin, J.-M., Michaud, F., Rouat, J., Letourneau, D.: Robust sound source localization using a microphone array on a mobile robot. In: Proceedings of Interation Conference on Intelligent Robots and Systems (IROS), vol. 2, pp. 1228–1233 (October 2003)
18. Vogel, D., Balakrishnan, R.: Distant freehand pointing and clicking on very large, high resolution displays. In: UIST 2005. Proceedings of the 18th annual ACM symposium on User interface software and technology, pp. 33–42. ACM Press, New York (2005)