# Interactive Walls: Addressing the challenges of large-scale interactive surfaces

Jay Summet, Ramswaroop Somani, Gregory D. Abowd, and Jim Rehg

College of Computing and GVU Center
Georgia Institute of Technology
{summetj,somani,abowd,rehg}@cc.gatech.edu

**Abstract.** We present a prototype large-scale interactive electronic whiteboard wall. Various input, output and vision technologies are used to create a surface that can capture digital ink as well as support pen-based interaction with displayed information on subregions of the wall. A simple automated capture application is demonstrated on our prototype surface and research challenges for developing more complex applications with this interactive technology are discussed.

**Keywords:** large-scale interactive surfaces, automated capture, input and output technologies, software engineering

## 1 Introduction

While electronic whiteboards are becoming more popular, in practice they are not large enough for a variety of applications. In a typical 50-seat classroom, for example, a traditional whiteboard is at least three times the size of the largest available commercial electronic whiteboard system. In previous work in the classroom[1], the interactive electronic whiteboard was extended by two additional non-interactive projected surfaces to facilitate showing the history of a lecture presentation. Office-related environments, such as Stanford's iRoom[8], Berkeley's Designer's Outpost[9], or IBM's Everywhere Display[11] show the importance of creating interactive surfaces which ultimately cover entire walls.

The work reported here is inspired by the need for large-scale interactive surfaces that are an alternative to a chain of embedded back-projected displays in the walls of a room. We present an example large-scale interactive surface in the form of instrumented whiteboard walls. Our two current prototype walls measure 17 and 19.5 feet across and both are 8 feet high (5.2 and 5.9 by 2.4 meters). With additional hardware the system is scalable to even larger sizes. Each wall presents a single input surface, and we have added the unique ability to point multiple, front-projected displays at arbitrary subregions on the walls. This creates a surface that can be used simultaneously to capture handwriting as digital ink, as well as provide pen-based interactive capabilities on the displayed subregions. In Section 2, we will give an overview of the input and output technologies that are used to create this flexible interactive wall surface as well as demonstrate a simple automated capture application that exploits the interactive wall's capabilities.

Moving the interaction space from the desktop to the wall provides several research challenges, discussed in Section 3. There are obvious engineering challenges confronting the seamless integration of a variety of input, output and vision technologies so as to create highly responsive interactive walls. In programming applications to exploit these walls, we break several assumptions about the relationship between input and output that have ruled desktop-based development. These broken assumptions become open research challenges relating to non-homogeneous coverage and resolution of input and output technologies. There are also some software engineering research challenges concerning the correct programming abstractions for application developers.

## 2 Technology Overview

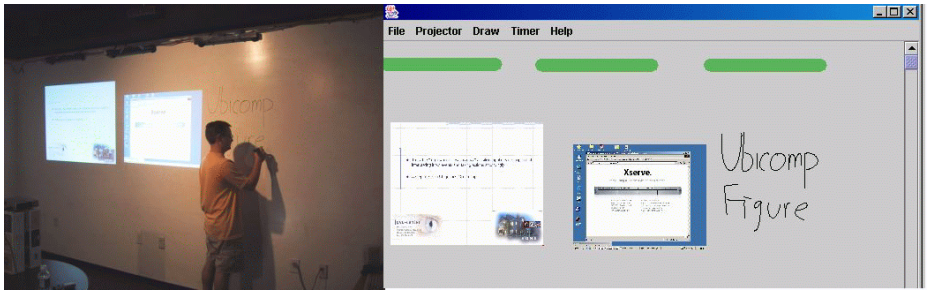


**Fig. 1.** The left image shows a user at the smaller of the two interactive walls in the basement media room. The scale of this half of the input surface (with two output sub-areas displayed) is evident. The right image shows a screenshot of an access application for captured pen strokes and computer output.

When working with our prototype, users interact with two adjacent walls which are covered with dry erase material. Using six mimios[6] (See section 2.1) we are able to track pen input over 192 $ft^2$ (17.8 $m^2$). Because the interactive walls are covered with a traditional dry erase surface and users hold standard whiteboard pens housed in mimio pen-holders, the affordances of using a very large whiteboard are maintained, although we lose the direct augmentation ability of systems such as Flatland[10] that make use of fully electronic ink.

All strokes on the wall are captured by the system for future access. While using the system, any user selectable portion of the wall may be used to display a traditional computer desktop via projectors. Users can give a PowerPoint presentation, display a web page, video conference, or use any other desktop application. Pen events (taps & drags) inside of these projection areas[1] are interpreted by the system as mouse events

[1] Each projection area must be manually calibrated when created or moved by touching diagonal corners. We plan on using computer vision to auto-calibrate these areas allowing even easier creation and movement.

(as opposed to pen strokes) and routed to the appropriate computer over the local area network. The same client that accepts pen/mouse events over the network forwards periodic screenshots back to the server, to be saved along with the pen strokes for later review. In the access application, these screenshots are placed where they were originally projected relative to the captured pen strokes. (See figure 1.)

As front-projection is subject to occlusion by users, we have begun work on using multiple overlapped projectors for virtual back-projection capabilities (Section 2.2). A client-server architecture is used to coordinate the computers involved in presenting multiple desktop environments on the wall and capturing events (Section 2.3).

### 2.1 Input: Tiled mimios

User input actions, in the form of pen strokes and taps, are captured via a chain of off-the-shelf mimio devices. Because the mimio devices are designed for stand-alone use, we developed custom software which tiles their sensor output together across the entire surface. This software interpolates between the reference frame of each individual mimio and the global coordinate frame of the display wall, providing a single input coordinate space per wall. In addition, it compensates for a non-linear warping which occurs in the mimio outputs.

### 2.2 Output: Virtual Back-Projection

In standard front-projection, as our current system uses for output, a single projector is used to provide output capabilities on a display surface. However, if a user approaches the surface (as commonly occurs with interactive surfaces) they risk occluding the projection, ruining the output capabilities of the surface. This problem can be solved with back-projection, although this solution has a high cost, both in terms of installation effort and space requirements. Additionally it is difficult, costly, and sometimes impossible to retrofit existing walls with back-projection capability. For example, the wall of the basement media room that our research is conducted in is lower than the local water table, and adjacent to a city street.

We are developing virtual back-projection to solve the occlusion problem (See figure 2), allowing users to approach our interactive walls without obscuring their work and without being illuminated with blinding light from the projectors. By using multiple overlapped projectors, and dynamically varying their output depending upon which projection paths are currently occluded, we will be able to simulate a back-projected surface. Proofs of concept for dynamically eliminating shadows[13,7] and eliminating projector illumination on the user[14,3] have been demonstrated. We are currently working to enhance and combine both techniques into a production system which can display dynamic content, and integrate this output subsystem with the rest of the interactive surfaces infrastructure.

### 2.3 Client-Server Infrastructure

Our current prototype uses a client-server model implemented in JAVA. Figure 3 presents an architectural diagram of the system. Clients sit on remote machines which project

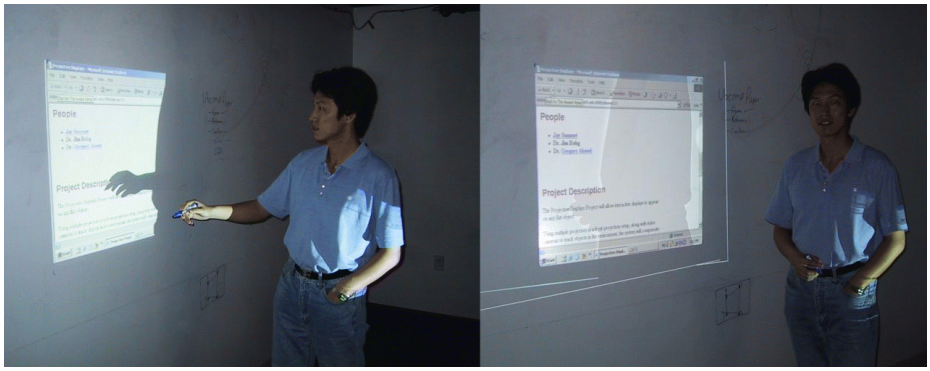

**Fig. 2.** In the left image, the user is occluding a portion of a desktop projected using a single projector. On the right, two projectors are used to dynamically eliminate the shadow. The current system does not incorporate pre-emptive shadows, so the body of the user is still illuminated by the occluded table-top projector.

their desktop and any applications the user may wish to use onto the wall. The server receives pen events from the tiled mimio input hardware and, if they fall within a projection area, routes them via TCP/IP to the appropriate client, which interprets them as mouse events for the local computer. Pen events which do not fall within a client-controlled (projection) area are interpreted as pen strokes, producing digital ink which is saved for future access, along with periodic screenshots provided by the clients.

As we integrate virtual back-projection, the client computers will no longer directly drive projectors, and output will be sent to dedicated projection control computers, abstracting the output much as the system currently abstracts the input. We will also support native applications that can react to the unique properties of the interactive walls via our own API (see Section 3.2). Such applications would not be trapped in a desktop window, instead being free to move independently around the interactive surface.

## 3 Research Challenges

As with most research platforms, pure engineering problems must be overcome before applications can be prototyped and the system can be evaluated. Although requiring significant system engineering work, these issues are solvable, as evidenced by work such as the Princeton Display Wall[4] and the Stanford Interactive Mural[5], two projects which tile back-projectors into large displays. More interesting from a ubiquitous computing research perspective are the problems of resolution and coverage mismatch between input and output technologies (Section 3.1), and software engineering and API issues for native applications (Section 3.2).

### 3.1 Input/Output Mismatch Problems

With a desktop computer, input is possible over the entire output region and input resolution is much finer than output resolution, easily allowing users to select individual

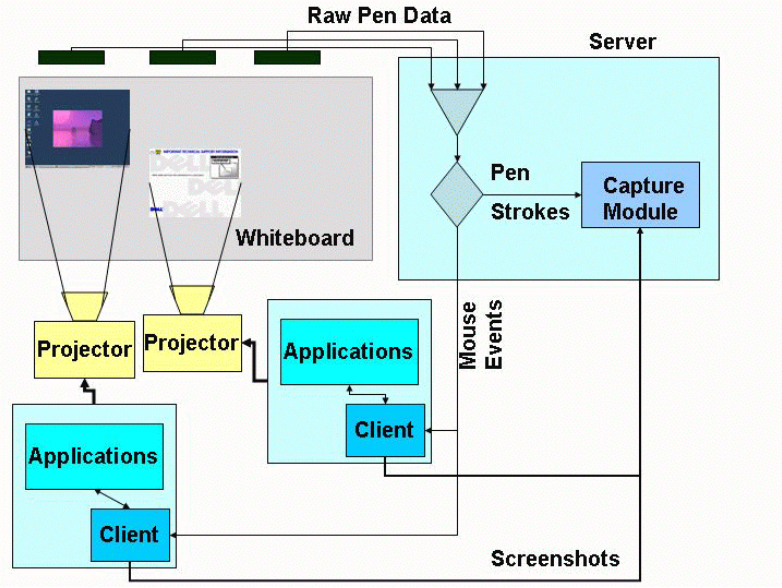


**Fig. 3.** A diagram outlining data flow within the client-server architecture.

pixels. Additionally, input and output resolutions remain constant over the entire interaction area. None of these assumptions necessarily apply to large interactive surfaces, like the one we have prototyped, leading to the following two general classes of problems.

**The Non-Homogeneous Coverage Problem.** A unique feature of our prototype is that we currently support input (pen event capture) on the entire wall, of which the output (projection) capable areas are a subset. It is possible to add enough projectors, or use multiple re-taskable projectors like the Everywhere Display[11] to provide virtual coverage of the entire wall. However, having projected regions as a proper subset of the input surface is an interesting design point that presents challenges a general infrastructure should handle gracefully.

The reverse problem (input region a proper subset of the output region) can also be imagined, where a short range sensor mounted on the floor is only able to capture user input on the bottom two-thirds of a wall. This problem does not have to be the result of immature hardware. Even with perfect input coverage, the scenario perfectly describes the case of a short user working with a tall interactive wall.

One solution to the situation of having more output surfaces than input surfaces is hyper-dragging on Augmented Surfaces[12], where users use a small input area (their laptop's pointing device) to manipulate objects displayed on a larger output environment. This solution assumes that each user carries an input device with them, while we assume input capability is located on surfaces in the environment, provided by unspecified sensors with varying resolutions, of which our mimio network is an initial

prototype. Because these input areas are not necessarily mobile, they may not be conveniently located near the user. Nevertheless, applications must be able to provide services in these dynamic environments, being able to make use of input areas that are not output capable, and *vice versa*.

**The Multi-Resolution Problem.** Using a variety of input and output devices over an interactive surface can lead to varying input and output resolutions. As an example, an interactive wall could be output enabled at a low resolution (e.g. via a long throw projector) on its entire surface, while providing one or more high-resolution patches (e.g. via a zoomed projector). Such a multi-resolution, focus plus context display has been shown to enhance task completion speed and reduce error rates on appropriate tasks[2].

The same wall could incorporate a touch sensitive SmartBoard in the center, while input outside the SmartBoard is provided by a low-resolution optical hand tracker. Applications must react to such situations by making appropriate use of the different capabilities of the environment in which they find themselves. Proper support from the software infrastructure will be required to free application developers from most of the burdens involved with keeping track of where it is appropriate to place UI elements or display a high-resolution image.

### 3.2 Software Engineering Challenges

We face many software engineering challenges which can be broken into two subcategories, System Infrastructure and Support for the Application Developer. Many of the challenges in the first area result from the distributed nature of the problem, while most of the challenges in the second area result directly from the input/output mismatch problems discussed above.

**System Infrastructure.** The system must solve many challenges hidden from the application developer, such as being able to interface with multiple input and output devices, and work reliably in the face of hardware failure. Additionally, it must provide support for mediation between multiple applications, and a method of application management, much like a window manager on the desktop.

**Support for the Application Developer.** Ubiquitous applications need to be aware of, and have the ability to adapt to, the changing attributes of a world where input technologies may shift from wall to wall, output resolutions may vary from surface to surface, and the user expects his applications to follow as he moves.

However, this awareness, if provided only in the form of a low level API, which forces the programmer to determine exact information about the state of the environment and make specific requests for resources[2], will make application development even more difficult than it already is. Low level API's can be valuable, but should be

[2] "Iterate high-res output areas. Find an unused one. Move my image there."

augmented with high-level API's and framework-based support. For example, a high-level interface could allow the application to specify a "requested" output resolution for each UI element (e.g. a button could appear at 10dpi while an image would request 72dpi) and the system would attempt to layout the application to meet these requests. Indirect, or framework-based support, would be transparent to the application developer, possibly even allowing legacy, non-native applications to function effectively on the interactive surface.

A legacy application using an existing UI framework (e.g. Java AWT or Swing) that has been modified to support the interactive wall would layout UI elements such as buttons and image panels as appropriate for the environment. Such a framework could implement a partial solution to the "short user" problem introduced above by automatically placing tool-button palettes near the user.

## 4 Conclusion

In this paper we have introduced our prototype wall-scale input surface which has sub-areas that are output enabled. The unique aspects of our prototype are the extremely large input area, non-homogeneous areas of output, and the enabling of fluid transitions between whiteboard marking and computer control. We used our prototype's input and output technologies as examples to motivate the general problems of non-homogeneous coverage and multiple resolutions of input and output on interactive surfaces.

While our prototype is focusing specifically on the design meeting and classroom areas in the short term, our research results should be general enough to apply to other uses of interactive surfaces. The prototype is driving our future work in developing a software infrastructure to support applications natively developed for interactive surfaces, as well as allowing current desktop applications to be more easily used on, or ported to, such an environment. In the end, the utility of interactive surfaces depends upon the applications available to the user, which in turn depend upon the software infrastructure in place to ease the creation of these applications.

## Acknowledgment

The authors thank Desney Tan for discussions about his work on pre-emptive shadows, Zhonghao Yang and Thomas O'Connell for their assistance with the figures, and members of the Future Computing Environments group for discussing drafts of the paper. This work is sponsored in part by the Aware Home Research Initiative and the National Science Foundation (grant 0121661).

## References

1. Gregory D. Abowd. Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, 38(4):508–530, October 1999.
2. Patrick Baudisch, Nathaniel Good, Victoria Bellotti, and Pamela Schraedley. Keeping things in context: A comparative evaluation of focus plus context screens, overviews, and zooming. In *CHI'02 Proceedings*, pages 259–266, 2002.

3. T.-J. Cham, R. Sukthankar, J. M. Rehg, and G. Sukthankar. Shadow elimination and occluder light suppression for multi-projector displays. In *Submitted to Proc. IEEE Visualization*, 2002.
4. Kai Li et. al. Early experiences and challenges in building and using a scalable display wall system. *IEEE Computer Graphics and Applications*, 20(4):671–680, 2000.
5. Greg Humphreys and Pat Hanrahan. A distributed graphics system for large tiled displays. In *Proceedings of IEEE Visualization*, 1999.
6. Virtual Ink Inc. *http://www.mimio.com*.
7. C. Jaynes, S. Webb, R. M. Steele, M. Brown, and W. B. Seales. Dynamic shadow removal from front projection displays. In *Proc. IEEE Visualization*, 2001.
8. Brad Johanson, Armando Fox, and Terry Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):To appear, 2002.
9. Scott R. Klemmer, Mark W. Newman, Ryan Farrell, Mark Bilezikjian, and James A. Landay. The designers' outpost: A tangible interface for collaborative web site design. In *Proc. ACM UIST'01 Symposium on User Interface Software and Technology*, pages 1–10, 2001.
10. Elizabeth D. Mynatt, Takeo Igarashi, W. Keith Edwards, and Anthony LaMarca. Flatland: New dimensions in office whiteboards. In *CHI'99 Proceedings*, pages 346–353, 1999.
11. Claudio Pinhanez. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In *Proceedings of Ubiquitous Computing (Ubicomp)*, pages 315–331, 2001.
12. Jun Rekimoto and Masanori Saitoh. Augmented Surfaces: A spatially continuous work space for hybrid computing environments. In *CHI'99 Proceedings*, pages 378–385, 1999.
13. R. Sukthankar, T.J. Cham, G. Sukthankar. Dynamic shadow elimination for multi-projector displays. In *Proceedings of CVPR*, 2001.
14. Desney S. Tan and Randy Pausch. Pre-emptive shadows: Eliminating the blinding light from projectors. In *CHI'02 extended abstracts*, pages 682–683, 2002.