Mixed Reality Tabletop (MRT): A Low-Cost Teleconferencing Framework for Mixed-Reality Applications

Daniel Bekins, Scott Yost, Matthew Garrett, Jonathan Deutsch,
Win Mar Htay, Dongyan Xu[+], and Daniel Aliaga[*]

Department of Computer Science at Purdue University

ABSTRACT

Today's technology enables a rich set of virtual and mixed-reality applications and provides a degree of connectivity and interactivity beyond that of traditional teleconferencing scenarios. In this paper, we present the Mixed-Reality Tabletop (MRT), an example of a teleconferencing framework for networked mixed reality in which real and virtual worlds coexist and users can focus on the task instead of computer interaction. For example, students could use real-world objects to participate in physical simulations such as orbital motion, collisions, and fluid flow in a common virtual environment. Our framework isolates the low-level system details from the developer and provides a simple programming interface for developing novel applications in as little as a few minutes, using low-cost hardware. We discuss our implementation complete with methods of hand and object tracking, user interface, and example applications focused on remote teaching and learning.

Keywords: D.2.6.b Graphical environments; H.4.3.b Computer conferencing, teleconferencing, and videoconferencing; H.5.1.b Artificial, augmented, and virtual realities; I.3.2.a Distributed/network graphics.

1. Introduction

The proliferation of high-performance audio, video, and networking technologies enables distant parties to interact in rich collaborative environments. Simultaneously, virtual and mixed reality technology supports more direct and natural interactions with computer systems than is possible with standard input devices. Together, these technologies enable a more immersive collaboration than is possible in traditional teleconferencing.

In an effort to make mixed-reality available to a widespread audience and to enable rapid development of applications, we have developed the Mixed-Reality Tabletop (MRT) to demonstrate the viability of a low-cost, networked mixed-reality system (Fig. 1). The MRT provides a common plane for immersive demonstration and perception in which the real and virtual worlds coexist. As opposed to traditional video conferencing in which two areas of attention are required (computer and real-world), the MRT merges the two to allow more natural and direct interaction with the task at hand, free of traditional devices like the monitor, keyboard, and mouse. A MRT station comprises a physical tabletop, a PC, camera, and projector. The projector displays virtual output such as video, text, interface, and other imagery onto the tabletop. The camera tracks real-world objects and user hand movements to allow natural, device-free interaction.

* = aliaga@cs.purdue.edu
+ = dxu@cs.purdue.edu

There is a wide spectrum of application scenarios made possible by the MRT environment. Given its low-cost infrastructure, schools can install the system to enable children at several different stations to work collaboratively on a virtual puzzle, a painting, or learn a skill like origami from a remotely stationed teacher. The MRT keeps children focused on the learning task instead of computer interaction. Students could also use real-world objects to participate in physical simulations such as orbital motion, collisions, and fluid flow in a common virtual environment. Because there are no physical input devices involved, several students can participate on the same station without restriction.

Our general framework provides the essential functionality common to networked mixed-reality environments wrapped in a flexible application programmer interface (API). We have chosen a demonstrative group of features for the MRT that will showcase the viability of such a networked mixed-reality system. We discuss our implementation complete with methods of hand and object tracking, user interface, and sample applications.

A rich history of research in virtual, augmented, and mixed reality systems provide a foundation for our work. Application development platforms for mixed-reality applications have been developed such as MR platform [1], Tinmith-evo5 [2], VITA [3], and Teleport [4]. These platforms target 3-D mixed-reality applications that require head-mounted displays and complicated infrastructure making widespread deployment difficult. On the other hand, MRT targets 2-D collaborative scenarios that enrich traditional conferencing applications and can be installed at significantly less cost. As opposed to using a touch screen (e.g. a Tablet PC), we use a large desktop and a more natural interface.

Closely related to the MRT are several flavors of custom mixed-reality workbenches [5][6][7][8], systems that support a variety of display surfaces (e.g., Emancipated Pixels [9], IBM's Everywhere Displays [10], TouchLight [11], and Build-It [12]), and Hi-Space Enhanced [13] (a powerful test-bed and API for implementing, measuring and analyzing direct interactions of multiple users standing around a single tabletop display surface).

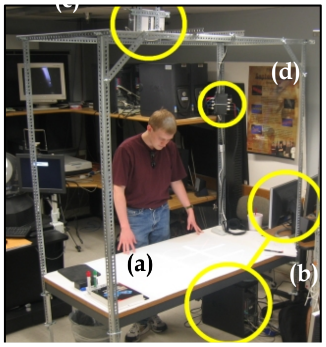

Figure 1. MRT setup. A MRT station consists of (a) a physical tabletop, (b) a PC, and (c) an overhead camera-projector tandem, and (d) synchronization device.

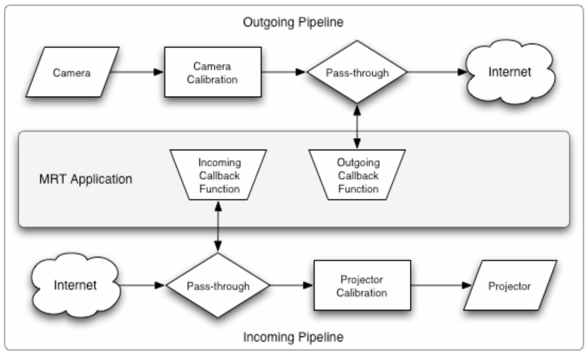

Figure 2. MRT pipeline. A conceptual diagram of the software pipeline of a MRT station.

These systems provide a very useful basis for developing a single MRT station, though none currently create both a networked mixed-reality environment and support simultaneous real-world and virtual display.

Three mixed-reality systems that do take advantage of networking capabilities are the Tele-Graffiti [14] project, blue-c project [15], and Augmented Surfaces [16]. Tele-Graffiti uses networked camera-projector setups to perform remote sketching, but does not offer a method of object tracking and focuses on one application. The blue-c project is developing a system to enable real-time reconstruction of the participants and their full immersion in a virtual world. Their approach, which uses a custom 3D stereo projection theatre, distributed computing architecture and dedicated highspeed networks, is not readily deployable to a large user-base. The Augmented Surfaces project extends the desktop of portable computers to tables and to displays using a custom infrastucture and visually-tagged objects and does not readily expose a programming interface.

2. SYSTEM OVERVIEW

The MRT configuration consists of a PC, camera, projector, and tabletop. Multiple stations are connected via a network to create a common, interactive mixed-reality environment. The cost for converting an existing PC into an MRT station is about $100-$500 for a camera and $1000 for an entry-level projector.

We have created a framework that handles all of the low-level details of a MRT application. Figure 2 provides a conceptual diagram of the software pipeline of a single MRT station. During the incoming pipeline, the station receives and processes video from remote stations before displaying output to the projector. During the outgoing pipeline, it receives and processes local video before sending it to the network. Remote video is formatted and displayed during the incoming pipeline, while object and hand tracking takes place during the outgoing pipeline. The application accesses the video data via callbacks.

3. SYNCHRONIZATION AND CALIBRATION

One of the most challenging problems in composing real and virtual objects is simultaneously displaying output to and accepting input from the same location. A single overhead camera sees both the real-world objects as well as the projected virtual image, so a system is needed for differentiating between the two.

Our approach is to control the shuttering of the camera and projector so that the camera only captures the real-world objects. To accomplish this, we synchronize the projector with the camera, making sure the projector is showing black while the camera is taking a snapshot. A "black box" electronic device receives a sync signal from the PC and sends out sync signals to the camera and

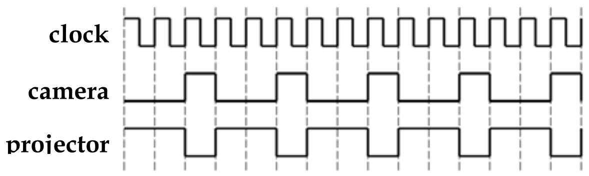

Figure 3. Synchronization. The camera becomes active during every third frame.

to the projector. We use a Point Grey Firewire camera that accepts external trigger and send sync signals to the project via a standard RGBHV connection. Our projector runs at 60 Hz, and we take a snapshot of the table every third frame, or at 20 Hz (Fig. 3). While this effectively solves the problem, it does result in a noticeable flicker. We discuss possible improvements to this method in the future work section.

Because of misalignment between the camera and projector it is necessary to calibrate each device to obtain a consistent standard coordinate system. Calibration is performed during installation by establishing correspondences between points in the camera image, projector image and physical table. During each frame, the camera image is postwarped into canonical space before making it available to the application and the output image is prewarped before projection onto the tabletop.

4. APPLICATION PROGRAMMING INTERFACE

Our application programming interface provides access to events related to incoming and outgoing table images, network data, object tracking, and hand tracking. Because the hand tracking system is similar to a standard point-and-click interface, MRT application code is nearly identical to standard GUI application code. The API also provides controls such as buttons, panels, and labels, as well as an easy way to create custom controls.

4.1 Programmability

The API, developed in C++, takes advantage of class inheritance and virtual functions. In order to create a fully functional MRT application, the programmer simply creates a class derived from the MRTApp base class and provides function overrides for the desired events (Fig. 4). The framework includes a networking layer to send both video and data at 30 frames per second. Separate sockets are established for each and are accessible via the API. Point-and-click events are exposed in their most essential form. However, the application still has access to the tracked objects, hands, and raw local and remote video images.

There are several MRT configurations that make sense in a networked environment, and not all of them require the same set of features. The application can disable unneeded features in order

```
class SampleApp : public MRTApp {
public:
   SampleApp() { }
   void Initialize() {
       btnQuit = new MRTButton("Quit", 0.5, 0.5, 0.25, 0.25);
       AddControl(btnQuit); }
   void OnRender(); // OpenGL per-frame rendering code
   void OnVideoIncoming(uchar *video); // process in images
   void OnVideoOutgoing(GLubyte *video); // process out images
   void OnPointerDown(int pointerId, float x, float y);
   void OnPointerMove(int pointerId, float x, float y);
protected:
   MRTButton *btnQuit;
   // add application data here
};
```

Figure 4: Example application. A simple application derived from MRTApp base class that creates a clickable quit button.

Application	Hand Tracking	Object Tracking	Networked Video
Tic-tac-toe	Yes	No	None
Interactive Classroom	Yes	Yes	One way streaming multicast
Interactive Physics	Yes	Yes	Two way snapshots
Interactive Origami	Yes	No	Two way streaming

Table 1. Application features. The framework is fully customizable to allow a variety of application configurations.

to gain performance. For example, a simple tic-tac-toe application might use hand tracking and virtual graphics, but not video. The API makes all of the possible scenarios simple to realize and optimize for performance. Our sample applications demonstrate a variety of possible MRT configurations (Section 5). The features implemented for each application are summarized in Table 1.

4.2 Object and Hand Tracking

To provide the user with a list of objects on the table and to support point-and-click (e.g., a virtual mouse), we implemented a real-time method for separating foreground and background pixels and subsequently grouping pixels into objects using a single camera. While a near-infrared camera could be used to assist with tracking, it would require an additional camera per station to capture color images for mixed-reality rendering. Instead, during each frame a snapshot of the empty tabletop is compared with the current snapshot. Foreground pixel regions are segmented by tracing the outline of each connected component in the image and creating a polygonal outline. By tracing the outline we ignore potential “holes” in the interior of an object. Regions touching the frame edge are classified as pointers (e.g., user’s arm and hand reaching to an object), while purely interior regions are counted as table objects (objects touching the frame edge at startup are also considered table objects). The API makes available to the programmer several object features such as object border, object center, and object size in pixels (Fig. 5a).

Each object on the table is assigned a numeric label to allow the programmer to track its movement. During each frame, the objects in the current frame are compared to those of the previous frame in order to reassign labels. The size, shape of the polygonal outline, and location of each object is used to determine correspondence between frames. Because tabletop objects are unlikely to move or change shape very quickly, this simple heuristic is sufficient and fast.

It is also desirable to provide a mechanism of point-and-click that will be familiar to MRT users and therefore make the system more intuitive. Such a system also has the advantage of making MRT applications nearly identical to standard GUI applications, enabling a seamless transition from “PC-only” mode. We have found that it is very simple and natural to use fingers as both “pointers” and “clickers” by opening and closing the hand. For example, the index finger can be used to point, while the thumb can be used to click. Alternatively, a scissor motion can be made with the index and middle finger, or a grabbing motion can be performed with the entire hand. It is interesting to note that the MRT allows several pointer objects at once, as opposed to the traditional mouse. A single user can use both hands to perform tasks, and several users can perform point-and-click actions on the same table at once.

Hand tracking is performed in software by observing the pixel-thinned skeleton of the hand and arm region (Fig. 5b-c). A graph is constructed describing the pixel connectivity, then inspected for certain features. A closed hand (mouse down) contains no sharp edges and its graph will therefore consist of a single edge oriented according to the forearm. An open hand (mouse up) will consist of a forearm edge as well as several edges corresponding to fingers. In each case, we take the farthest point from the table border as the current pointer position. Hand tracking is then a simple matter of 1) how many nodes are in the graph (mouse state), 2) where the farthest graph node from the edge is (mouse location).

In practice, variations in shadows and lighting may cause artifacts in the form of superficial graph edges, which affects hand tracking. To address this, it suffices to use length metrics for the graph edges to determine if they represent fingers or noise. It is unlikely that edges due to noise will be as long as a finger. An especially noisy graph can simply be removed from consideration.

5. Sample Applications

We have implemented three sample applications using the MRT API for use in a classroom setting. The API is exposed as a C++ library of classes and methods. Each application demonstrates a different configuration and use of MRT. Two of the three applications were developed by programmers with no knowledge of the underlying system details.

The *Interactive Classroom* application is designed as an aid to an actual classroom lecture held over the Internet (Fig. 6a). Video of real-world objects placed on the instructor’s tabletop is sent to the student tables, where they can identify certain parts of the object, ask questions about the object, or be quizzed interactively by the instructor. The instructor could examine an artifact, perform a dissection, or disassemble a mechanical part.

The *Interactive Physics* application allows students to experiment with the physics behind gravitational motion (Fig. 6b). Students at each MRT station supply a real-world object to act as a satellite for the object at the teacher table. The students set the mass and initial velocity of their objects, and then view a simulation showing the orbital path of their object. Since the real-world object cannot be animated, the projected background is animated instead. The background for each table includes images of the other tables’ objects, showing their relative motion and rotation. Thus, there is a common global coordinate system and a local coordinate system centered at and aligned with its real-world object. At any time the user can adjust the position of their satellite and view the newly calculated motion.

The *Interactive Origami* application allows the teacher to enhance the view of the real object (origami paper) with virtual objects (illustration) to conduct a more effective remote teaching

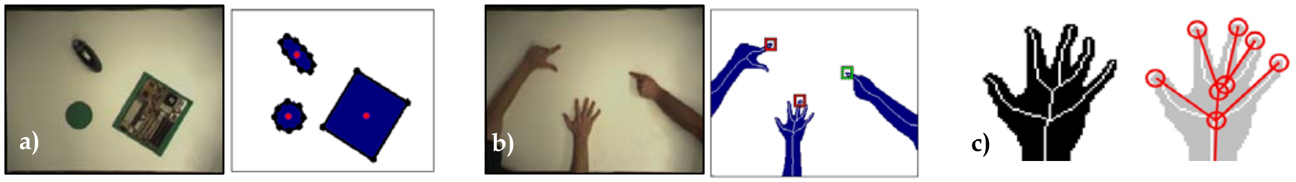

Figure 5. Tracking. (a) The system approximates the center and polygonal border of each object. (b-c) The pixel skeletons of hand pointer regions are used to classify them as open or closed.

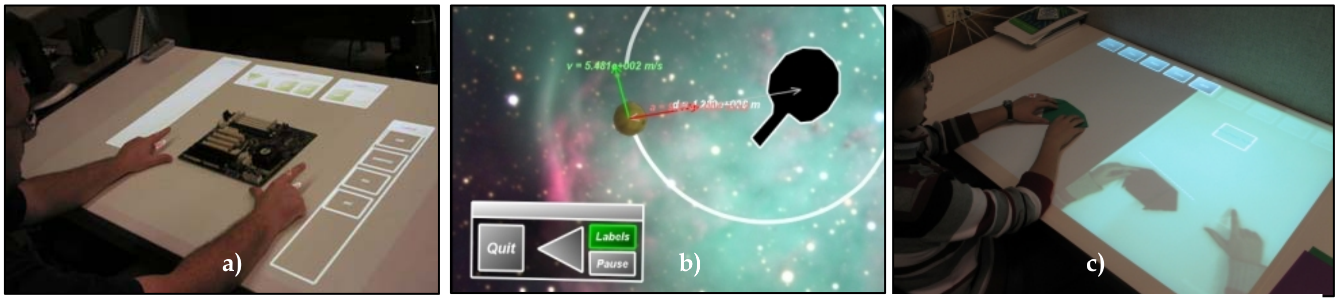

Figure 6. Sample applications. (a) An instructor teaches computer circuitry to a set of remote students. (b) Students control a networked physics simulation. (c) A teacher demonstrates origami using a set of virtual illustration tools.

session (Fig. 6c). Each table is split vertically into a local workspace and video window displaying the remote tabletop. This allows the teacher and the student to examine each other's origami fold as if they are sitting next to each other. The application provides virtual drawing tools that allow the teacher to draw various lines and symbols corresponding to specific origami folds.

6. Discussion and Future Work

Our MRT framework uses a synchronization-based method to compose real and virtual objects. This allows us to simultaneously display output and accept input from the same location at the expense of a perceivable flickering effect. As future work, we would like to investigate several methods to ameliorate this flickering effect. Possibilities include using higher frame rate cameras and projectors and displaying an alternate color during the "projector-off" frame. In addition, we look at imperceptible structured light research as a viable way of addressing this problem [17]. For improved hand-tracking, we are looking to real-time implementations of more accurate batch processing methods [18, 19]. We would also like to extend our API to include collaborative tools such as object synchronization and locking.

Another avenue of future work is to liberate users from static tables and to provide them with mobile augmented-reality perception and interaction. Using Tablet PCs [20] as portable windows into a mixed-reality environment or IBM Everywhere Displays technology are both possibilities.

7. Conclusion

We have presented a low-cost framework to create mixed-reality applications in educational scenarios. Our approach, the Mixed-Reality Tabletop, allows networked immersive interactive learning that combines virtual and real imagery. The developer is given the essential functionality common to mixed-reality environments and is able to focus immediately on the application itself without having to worry about low-level details.

We look forward to deploying our lightweight system and framework to local schools and campuses. The quick installation and rapid prototyping made possible by our system make it ideal for non-graphics experts to develop applications and setup the system in classrooms or labs.

Acknowledgments

This research was made possible by funding provided in part by Microsoft Research Conference XP Team and by Oliver Colic for assembling the camera-projector boxes.

References

[1] S. Uchiyama, K. Takemoto, K. Satoh, and H. Yamamoto, "MR Platform: A Basic Body on Which Mixed Reality Applications Are Built", *Proceedings of IEEE ISMAR*, 2002.

[2] W. Piekarski and B. Thomas, "An Object-Oriented Software Architecture for 3D Mixed Reality Apps", *Proceedings of ISMAR*, 2003.

[3] H. Benko, E. Ishak, and S. Feiner, "Collaborative Mixed Reality Visualization of an Archaeological Excavation", *Proceedings of IEEE ISMAR*, 2004.

[4] C. Breiteneder, S. Gibbs, C. Arapis, "Teleport – An Augmented Reality Teleconferencing Environment", *Proc. of Eurographics Workshop on Virtual Env. and Scientific Vis.*, pp. 41-49, 1996.

[5] Leibe, B., T. Starner, W. Ribarsky, Z. Wartell, D. Krum, B. Singletary, and L. Hodges, "The Perceptive Workbench: Towards Spontaneous and Natural Interaction in Semi-Immersive Virtual Environments," *IEEE Virtual Reality 2000 Conference (VR'2000),* New Brunswick, NJ, March 2000, pp. 13-20.

[6] Ullmer, B., Ishii, H., "The metaDESK: Models and Prototypes for Tangible User Interfaces," *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1997, 223-232.

[7] Kobayashi, M. and H. Koike, "EnhancedDesk: integrating paper documents and digital documents", *Proceedings of 3rd Asia Pacific Computer Human Interaction*, pp. 57-62 (1998).

[8] H. Ishii, J. Underkoffler, D. Chak, B. Piper, E. Ben-Joseph, L. Yeung, Z. Kanji, "Augmented Urban Planning Workbench: Overlaying Drawings, Physical Models and Digital Simulation", *Proceedings of IEEE ISMAR*, 2002.

[9] J. Underkoffler, B. Ullmer, and H. Ishii, "Emancipated Pixels: Real-World Graphics in the Luminous Room," *Proc. Siggraph 99*, ACM Press, New York, 1999, pp. 385-392.

[10] C. Pinhanez, "The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces," *Proc. UbiComp 2001: Ubiquitous Computing*, LLCS 2201, Berlin, 2001, pp. 315-331.

[11] A. Wilson, "Touchlight: An Imaging Touch Screen and Display for Gesture-based Interaction", *Proceedings of Int'l Conference on Multimodal Interfaces*, 2004.

[12] Rauterberg, M.; Fjeld, M.; Krueger, H.; Bichsel, M.; Leonhardt, U.; Meier, M.: "BUILD-IT: a computer vision-based interaction technique for a planning tool", Proceedings of HCI, pp. 303-314, 1997.

[13] R. May, "Toward Directly Mediated Interaction in Computer Supported Environments", *Ph.D. Dissertation*, University of Washington, Seattle, CA, 2004.

[14] N. Takao, J. Shi, and S. Baker. "Telegraffiti: A camera-projector based remote sketching system with hand-based user interface and automatic session summarization," *International Journal of Computer Vision*, 53(2):115-133, July 2003.

[15] M. Gross, S. Wurmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. van Gool, S. Lang, K. Strehkle, A. Wande Moere, O. Staadt, "blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence", *Proceedings of ACM SIGGRAPH*, 2003.

[16] J. Rekimoto , M. Saitoh, "Augmented Surfaces: A Spatially Continous Work Space for Hybrid Collaborative Environments", *Proc. of ACM CHI*, 1999.

[17] D. Cotting, M. Naef, M. Gross, and H. Fuchs, "Embedding Imperceptible Patterns into Projected Images for Simultaneous Acquisition and Display", *Proceedings of IEEE ISMAR*, 2004.

[18] Xiao R., Zhu L., Zhang H., "Boosting Chain Learning for Object Detection", *Proceedings of ICCV*, 2003.

[19] Stenger B., Thayananthan A., Torr P., Cipolla R., "Filtering using a Tree-based Estimator", *Proceedings of ICCV*, 2003.

[20] G. Klein and T. Drummond, "Sensor Fusion and Occlusion Refinement for Tablet-Based AR", *Proceedings of IEEE ISMAR*, 2004.