MRT: Mixed-Reality Tabletop

Students: Dan Bekins, Jonathan Deutsch, Matthew Garrett, Scott Yost

PIs: Daniel Aliaga, Dongyan Xu

August 2004

Department of Computer Sciences
Purdue University

Goals

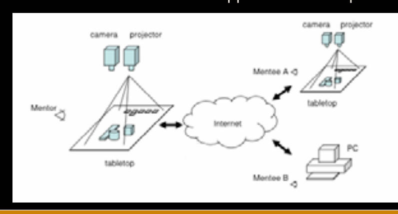
- Create a *common locus* for virtual interaction without having to shift attention between input and display devices
- Compose and synchronize *mixed-reality* video and audio for local and distant participants
- Create a *low-cost* scalable system that integrates multiple data streams over a uniform distributed platform

Motivation

- Immersive learning in Year 2020
  - "There is a power in virtual interaction" – Rita R. Colwell
- Going beyond current-generation whiteboard
  - Provide a natural focus of attention: lab table, desk, counter…
  - Support rich and intuitive interactions among distributed users
- Adding virtual and real objects to the equation
  - Mix real and virtual objects in the same focus of attention
  - Create virtual venue and context for interactions
- Wider deployment than full-fledged VR systems
  - Lower cost
  - Less infrastructural requirement
  - Easier to develop, install, and operate

Mixed-Reality Tabletop (MRT)

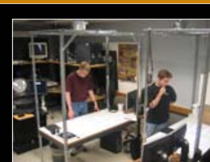
- Create *stations* containing a tabletop, camera, and projector to provide intuitive, device-free interaction
- Support both virtual and real objects on same tabletop
- Connect stations by transporting multimedia data over the network for composition and display on remote stations
- Provide a software toolkit for fast application development



Related Work

- Whiteboards
- HMD-based VR systems (UNC-CH, Feiner at Columbia)
- The Workbench (Barco, 3rd Tech)
- Tangible user interfaces (MIT, UVA)
- Emancipated Pixels (SIGGRAPH '99)
- Shader Lamps (Raskar at MERL)
- Everywhere Displays (IBM)

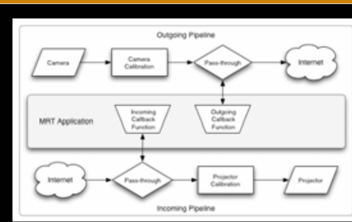
Example MRT Applications



Presentation

- Introduction
- ➔ System Overview
  - MRT Station
  - Station Pipeline
- Key Components
  - Synchronization
  - Calibration
  - User Interface
- Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
- Conclusions

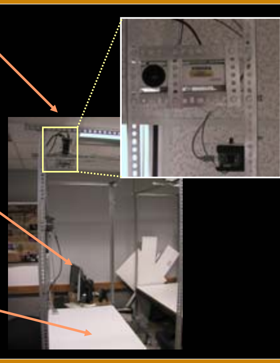
MRT Station Pipeline



- The stations are interconnected by a programmable pipeline for "composing" real and virtual imagery over a network

MRT Station

- Projector and camera
- PC workstation
- Tabletop



Presentation

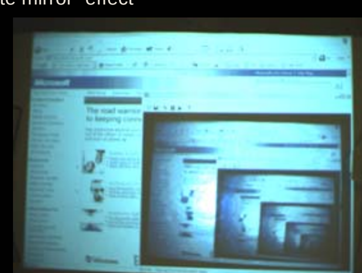
- Introduction
- System Overview
  - MRT Station
  - Station Pipeline
- ➔ Key Components
  - Synchronization
  - Calibration
  - User Interface
- Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
- Conclusions

MRT Software-only Station

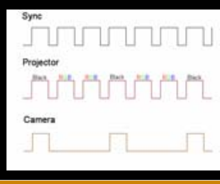
- PC only
  - Mouse movements are mapped into MRT environment

Camera-Projector Synchronization

- Synchronize the camera and projector to prevent an "infinite mirror" effect



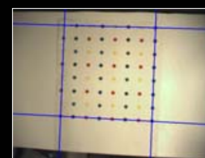
Camera-Projector Synchronization

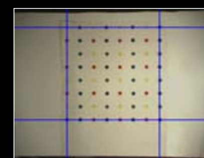


- Frame 1
  - Camera triggered
  - Black image projected
- Frame 2
  - RGB image projected
- Frame 3
  - RGB image projected
- and so on…

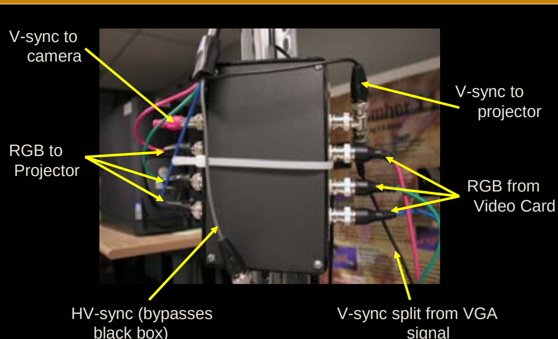
Calibration: Camera

- A snapshot is taken of a rectilinear grid on the tabletop
- Known points on the grid are corresponded to their pixel locations in the snapshot
- The point correspondences are used to approximate the camera warp [Tsai87]



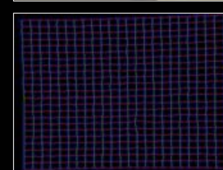


Camera-Projector Synchronization



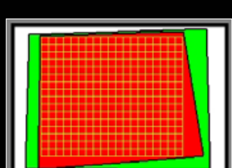
Calibration: Projector

- A rectilinear grid is projected onto the tabletop and recorded by the camera
- The recording is transformed by the camera warp
- Points on the grid are corresponded to their pixel locations in the warped camera image



Calibration

- Perspective and lens distortion cause the raw camera and projector images to be misaligned with the tabletop and each other
- Determine a mapping to warp from the camera's coordinate system to the tabletop's coordinate system



Tabletop overhead: the visible camera area (green) and projector area (red) are aligned with the tabletop (white) to form a rectilinear grid (yellow)

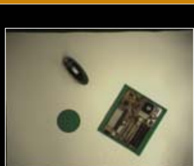
User Interface

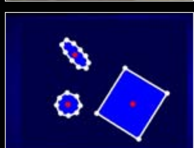
- Provide an intuitive graphical user interface with no interaction with keyboard or mouse
- Support object tracking and recognition
- Adopt same interface for PC-only mode

Tracking Objects

- Objects are distinguished from the white table background using an intensity threshold
- Foreground regions in the interior of the table are considered objects
- Foreground regions touching the edge of the table are considered hands or pointers
- Objects are tracked from frame to frame by considering attributes like pixel area and average position
- Mouse press events are simulated by the opening and closing of the hand

Tracking Objects





- The following attributes are determined for objects:
  - object center - average pixel location
  - object area - pixel count
  - object border - outline of pixel region
- The object border geometry is simplified to a small set of edges, based on an error threshold
- Moving objects are tracked based on attribute similarities between frames

Tracking Hands

- Hand regions are thinned to produce a single-pixel thick skeleton
- A graph is created to describe the skeleton's connectivity
- A hand's hotspot is placed at the farthest endpoint from the image border
- A skeleton with fingers is an open hand (mouse up)
- A skeleton with no fingers is a closed hand (mouse down)

Presentation

- Introduction
- System Overview
  - MRT Station
  - Station Pipeline
- Key Components
  - Synchronization
  - Calibration
  - User Interface
- ➔ Applications
  - API Framework
  - Interactive Classroom
  - Interactive Physics
- Conclusions

API Framework

- Provide basic controls like buttons, numeric selectors, and panels
  - Use C++ inheritance to create custom controls from a base control class
- Provide programmable event control
  - networking
  - mouse click, move, drag n' drop
  - object tracking
- Render graphics using DirectX/OpenGL

Application #1: Interactive Classroom

- Uses an Instructor / Student model
  - One instructor and multiple students
- Designed for use with students from grade 6 and up
- Instructor can use environment for:
  - Demonstrations and labs (e.g., biology dissections)
  - "Show and Tell" (e.g., describe parts of circuit board)

Instructor and Student Environments

- Instructor environment includes:
  - Programmable labels
  - Extendable list of students
  - Composable multiple-choice quizzes
  - Movable button panels
- Student environment includes:
  - Movable labels
  - Ask-question and submit-response buttons
  - Viewable user list
  - Movable button panels

Question Button

- Allows the student to notify the instructor that they have a question (e.g., raising your hand)
- Once pressed, the question button is colored brown
- This button will be colored green when the student table is considered live
  - after instructor recognizes the students question, or
  - if instructor “calls on” this student
- When the table is live, the student is now allowed to move labels
- The question button returns to original color when instructor deselects the student

Programmable Labels

- Label text loaded at run-time
- Instructor freely moves labels
- Instructor “calls” a specific student to move a label
- Instructor may correct student and move label to proper location

Quizzes

- Instructor:
  - Instructor presses the Quiz button
  - Presses up and down to select how many questions required
  - Move the quiz labels to proper location
  - The students selection will appear beside their user button after they have pressed ‘Submit’
  - Clear and ready for another quiz
- Student:
  - Make selection by clicking on appropriate quiz label
  - Press ‘Submit’
  - The student cannot move the quiz labels -- they can only select them and submit answers to instructor

User Lists

- Students added to list at runtime
- Student buttons are colored yellow when the student has pressed their question button
- Both instructor and students view the user list
  - Instructor list is interactive. A student is called upon by pressing their button. Their button will then be colored green
  - Student list is non-interactive. A student can only view the list

Interactive Classroom

Application #2: Interactive Physics

- Allow students to interactively experiment with physics concepts in mixed reality
- Allow remote tables to interact in a common physical simulation environment
- Take advantage of object tracking to model real physical characteristics
- Display interactive labels such as vector arrows

More Physics Tutorials

- Projectile Motion
  - Students attempt to hit targets on other tables by solving projectile motion equations
- Rotational Motion
  - Students experiment with the effects of applying force to various points on a real object. The system simulates the 2D center of mass and rotational inertia
- Collisions
  - Objects from various tables collide. Students can experiment with the effects of mass and velocity
- Fluid Dynamics
  - Flow lines are rendered to show fluid motion around objects placed on the tabletop

Interactive Physics: Orbital Motion

- Students learn about 2D orbital motion and Newton's law of gravity

  $F = ma = G M_0 M_1 / d^2$

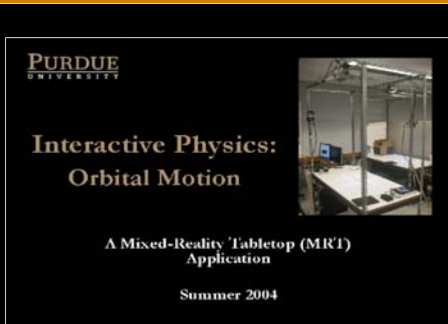
- Students and teacher set the mass of an object placed on their respective tables
- The teacher sets the scale of the universe
- The student sets the initial velocity vector for the orbiting object

Presentation

- Introduction
- System Overview
  - MRT Station
  - Station Pipeline
- Key Components
  - Synchronization
  - Calibration
  - User Interface
- Applications
  - API Framework
  - Interactive Classroom
  - Physics Tutorial

➔ Conclusions

Interactive Physics: Orbital Motion



In Conclusion…

- MRT creates a common tabletop for interaction among human users and objects
- MRT composes and synchronizes virtual and real objects for shared virtual venues involving local and remote users
- MRT demonstrates a low-cost scalable system that integrates multiple data streams over a uniform distributed platform

MRT Configuration and Performance

- Station specs:
  - Pentium 4 @ 3.2 Ghz, 512 Mb RAM
  - 100 Mbit Ethernet
  - 640x480 resolution camera triggered at 20 FPS
  - 1024x768 DLP projector at 60 FPS
  - (total cost ~$4000)
- Per frame processing:
  - video capture and warp: ~15 msec
  - object tracking: 1 to 10 msec (depending on object count)
  - network streamed video: ~7 msec
- Overall performance:
  - 20 FPS, limited by projector synchronization

Thank you!

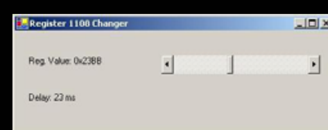
http://www.cs.purdue.edu/~aliaga/mrt.htm

Future Work

- Provide richer virtual interactions and scenario creation (e.g., urban planning, emergency response training, …)
- Use multiple projectors and/or cameras to reproduce approximate 3D renderings
- Extend to more pervasive display and capture surfaces (“Mixed Reality Room”)
- Enhance user’s perception by improving camera/projector synchronization (e.g., DLP synchronization, projecting non-black images, …)

Synchronization Drift

- Small delay (33 ms) between projector receiving signal and actual display
  - Drifts slowly over time
- Configure camera to delay after receiving trigger signal
  - Shutter delay is bits 16-31 of camera register 1108h
  - Set the register via “Register 1108 Changer”
  - Provides a graphical slider for setting camera delay



Acknowledgments

- Microsoft Research, Learning Sciences and Technology
- Computer Science Department, Purdue University
- Oliver Colic