2006

THE UNIVERSAL MEDIA BOOK

Shilpi Gupta
University of Kentucky, Shilpigupta81@hotmail.com

ABSTRACT OF THESIS

THE UNIVERSAL MEDIA BOOK

We explore the integration of projected imagery with a physical book that acts as a tangible interface to multimedia data. Using a camera and projector pair, a tracking framework is presented wherein the $3D$ position of planar pages are monitored as they are turned back and forth by a user, and data is correctly warped and projected onto each page at interactive rates to provide the user with an intuitive mixed-reality experience. The book pages are blank, so traditional camera-based approaches to tracking physical features on the display surface do not apply. Instead, in each frame, feature points are independently extracted from the camera and projector images, and matched to recover the geometry of the pages in motion. The book can be loaded with multimedia content, including images and videos. In addition, volumetric datasets can be explored by removing a page from the book and using it as a tool to navigate through a virtual $3D$ volume.

KEYWORDS: Computer Vision, Tracking, Human Computer Interaction, Augmented Reality, Volumetric Visualization

Shilpi Gupta

November 29, 2006

THE UNIVERSAL MEDIA BOOK

By

Shilpi Gupta

Dr. Christopher Jaynes
Director of Dissertation

Dr. Grzegorz Wasilkowski
Director of Graduate Studies

November 29, 2006

RULES FOR THE USE OF THESIS

Unpublished dissertations submitted for the Master's and Doctor's degrees and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part requires also the consent of the Dean of the Graduate School of the University of Kentucky.

THESIS

Shilpi Gupta

The Graduate School

University of Kentucky

2006

THE UNIVERSAL MEDIA BOOK

THESIS

A dissertation submitted in partial fulfillment of the
requirements of the degree of Master of Science in the
College of Engineering at the University of Kentucky

By

Shilpi Gupta

Lexington, Kentucky

Director: Dr. Christopher Jaynes, Department of Computer Science

Lexington, Kentucky

2006

ACKNOWLEDGMENTS

I would first and foremost like to thank my Master's Thesis advisor, Dr. Christopher Jaynes. I would like to thank him for giving me a chance, for his constant support and encouragement in all research efforts, for always being a motivating force, and for making me aware of the creative potential of computer science. I have learned a lot under his guidance, academic and otherwise, and I will always be grateful for that. I would sincerely like to thank Matt Steele, Nate Sanders, and Steve Webb. When I first came into the lab as an undergraduate, well, let's just say, I had a lot to learn. I would never have been able to get where I am today without their continuous willingness to help, and their encouragement. Thanks to all my colleagues in the lab for their useful class and lab discussions, and for their friendship. I would like to acknowledge Dr. David Nistér for making available the feature point processing and Preemptive RANSAC code used in this system. And of course, I would like to thank my family always for their love. Thank you Dad for your kindness and creativity, and thank you Mom for your youth.

Table of Contents

List of Figures

List of Tables

List of Files

s_gupta_thesis.pdf (8.6 M)

Chapter 1

Introduction

As computing systems increasingly impact our lives, the way humans acquire and interact with information continues to evolve. The progression from static, room-sized computing centers, to personal computers, to mobile, hand-held devices has shaped our relationship with technology by making computer-based information systems feasible, practical, and even necessary implements of our everyday lives. In addition to the efficiency of computing resources, the availability of peripheral devices, such as cameras, projectors, wireless sensors, audio systems, and so forth facilitates the synthesis of multiple resources where large collections of data can be processed simultaneously and presented as a cohesive system. Such multi-modal systems promote innovative means of human-computer interaction (HCI) where traditional approaches to acquiring information may be challenged and redefined.

Currently, the standard way to visualize and interact with information involves a monitor, a keyboard, and a mouse. This paradigm, in use since the 1970's, has obviously proven to be powerful for a variety of tasks, including word processing, database information retrieval, computer-aided design, graphical modeling, browsing the World Wide Web, and so forth. This system of interaction, however, has some fundamental limitations. One limitation involves an indirect transformation from task conception to task realization via a pipeline of symbolic manipulation. In standard interfacing, there exists an abstraction between input devices, such as the keyboard and the mouse, and graphical output devices, such as the monitor, for providing a system of visual representation [32]. The mouse, for example, is a device that accesses symbols (perhaps graphical icons in a windowed desktop environment), and these symbols, in turn, represent objects or functions. Direct control by a user is bequeathed to a sequence of processes needed to achieve a certain goal. At its core, the traditional paradigm limits the availability and usability of the tangible world that surrounds the user, as well as prevents the exploitation of humans' natural capabilities of physical movement in a physical space. The user's mental and physical focus is solely directed towards a fixed screen and towards the input devices used to access information on that screen. The magnitude of collaborative work amongst a group of users is also constrained. Humans naturally communicate through a rich and complex set of gestures and facial expressions (in particular, cues provided by gaze awareness [5]). Limiting such communication to a narrow line of sight can reduce the

benefits gained when users interact with each other.

Common to these limitations is the lack of a natural, or global continuum among resources and the users of these resources. The underlying "problem" may be summarized by the following statements from Pimentel's *Virtual Reality: Through the New Looking Glass*:

> Historically, the interface has been designed to leverage human capabilities to the advantage of computers, not humans. All too often, it would seem that the interface has been as much a way for the programmer to control the behavior of the user as a way for the user to control the behavior of the computer. In turn, this often places great limitations on a human's ability to use a computer. [23]

1.1 Alternative HCI

Alternative methods of human-computer interfacing often seek to bridge the disjoint means of control and visualization by creating an environment that responds to, as well as engages, the natural behaviors and cognitive capabilities of humans.

Non-traditional HCI techniques have progressed through an interesting array of developments. One of the first pivotal interaction devices that opened the doors of visual interfacing with computers was Ivan Sutherland's Sketchpad [30]. Sketchpad, developed in 1962, was a revolutionary computer program that explores computer graphics and interactive graphical user interfaces. The system introduced the light-pen, a device that allows users to point to and sketch geometric primitives directly onto the computer screen. Sketchpad can retain topological information about connected primitives to form symbols that can be manipulated as a whole entity.

Many innovations related to non-traditional interaction fall into the broad categories of virtual reality and augmented reality. Both of these paradigms strive to transform a static environment where users are confined to the limited capabilities of standard input and output devices into a system of multi-modal interaction that expands users' overall sensory experience.

The goal of a virtual reality (VR) system is to completely immerse a user into a new realm where awareness of the real world is suppressed while the user inhabits and responds to a virtual $3D$ world of computer generated input. Given such a system, a user may explore hypothetical worlds, or they may be subjected to simulations of real-world scenarios. VR systems attempt to automatically respond to the actions of users as they navigate through virtual worlds.

The idea of submission to an illusory environment was manifested early on with the invention of Morton Heilig's Sensorama in 1962 [23]. The Sensorama is a mechanized arcade-style apparatus that aims to immerse a user into a virtual multi-sensory

experience. One recorded experience, for example, is a motorbike ride through the streets of Brooklyn. A user sits in a vibrating seat and holds on to handle-bars while the system simultaneously produces sounds, aromas, and artificial breezes that simulates the real biking experience.

One of the first systems recognized as having potential as a virtual reality experience was the head-mounted display (HMD), first conceived by Ivan Sutherland in 1966 [23]. A typical HMD, worn on a user's head, provides a continuous feed of stereoscopic video content displayed in front of the user's eyes, immersing the user into a virtual world. The user's head (and sometimes eye) movements are tracked so that the graphics of the virtual world are rendered according to the changing viewpoint of the user. HMDs are in prevalent use for flight simulations where a pilot under training sits in a mock cockpit that mechanically pitches, yaws, and rolls in response to the pilot's actions. The Visually Coupled Airbourne Systems Simulator (VCASS), demonstrated in 1982 by Thomas Furness, is an HMD where symbolic representations of the outside world are projected onto screens inside a helmet [23]. The use of simplified diagrams to represent the world helps to reduce the distraction often caused by overwhelming amounts of visual input coming from $3D$ models of the real world. HMDs are often coupled with other input devices. For example, DataGlove, a pivotal wired glove device produced in 1985 by VPL Research, is able to measure the degree of motion in each of the finger joints [23]. The ability to track specific movement within the hand allows a $3D$ model of the hand to be rendered into the virtual world and allows a user to accurately grasp and manipulate objects in that world.

Other immersive systems are designed as spaces in which a user or group of users walk around and interact with the environment. In 1970, Myron Krueger presented Videoplace, a reactive environment where the computer responds to and anticipates the gestures of users by first capturing the user's motions via a video camera and subsequently projecting their silhouettes, composed into a computer-generated environment, onto a screen [23]. A user can also manipulate graphical objects on the screen using gestural movements. Videoplace was one of the first systems to explore the nature of human physical response to computer-generated environments in a non-intrusive manner: no specialized or cumbersome HMD-type devices are used, and the underlying technical system remains invisible to the user.

An increase in computing power and affordable resources led to the development of more advanced virtual reality systems. The Cave Automatic Virtual Environment (CAVE), realized in 1991, contains up to six walls built from rear-projection screens onto which high-resolution projectors render imagery to create an immersive display environment [9]. A user wearing stereoscopic shutter glasses becomes immersed in a virtual environment where objects may float in space as the user walks within the

display. The user's position in the world is tracked using electromagnetic sensors, and imagery displayed on the screens is correctly rendered from the user's point of view. In another VR system, the Metaverse Collaboratively Rendered Environment (Co.R.E.) from 2002, an arbitrary number of commodity projectors are loosely configured to project onto the surfaces of an arbitrary number of walls [17]. Imagery is seamlessly rendered onto the display surfaces from the point of view of a tracked user as they move within the display. The system allows for automatic calibration, as well as reconfigurability of projectors so that the display setup is flexible to suit varying space needs and application requirements. Each of these systems are designed to allow users to navigate through $3D$ information while the computer automatically responds to the movements of users. Such VR systems cater to a variety of applications, including scientific and medical visualization, terrain simulation, $3D$ gaming, and so forth.

In contrast to virtual reality, augmented reality (AR) serves as an intermediate interface between the real world and a virtual world. The goal of AR is to augment a user's perception of the world by seamlessly merging computer-generated information, or virtual data, with real-world objects or scenes. Some types of AR systems augment live or pre-processed video sequences with data, while others augment tangible objects in the user's space, often via projective devices. In this work, we focus on the latter. Integrating virtual data into familiar spatial domains readily falls into the idea of ubiquitous computing, which may be characterized by two main traits:

> (1) Ubiquity: Interactions are not channeled through a single workstation. Access to computation is "everywhere."
>
> (2) Transparency: This technology is non intrusive and is as invisible and as integrated into the general ecology of the home or work place as, for example, a desk, chair, or book. [5]

An early example of a system interested in exploring tangible user interfaces for an AR system is the DigitalDesk of Wellner, presented in 1991 [35]. A physical desk with a projector and camera hung overhead form an interactive environment where paper documents, as well as gestural movements such as finger pointing and tapping, are used as inputs for augmentation. The desk can also be augmented with virtual (projected) paper or other graphics. For example, a piece of paper inscribed with a ten-digit number is placed on the desk. A projector renders a graphical calculator beside the paper. Rather than using a finger to sequentially enter each number into the calculator, the user first points to the number and then taps the desk (underneath which are sound sensors), which then signals the camera to read the whole number and automatically enter it into the calculator. This example demonstrates how the computing environment automatically and unobtrusively responds to a user's actions. The achieved task saves time (one tap enters all the numbers into the calculator) and

helps to alleviate human error caused by manually entering numerical data.

In more recent work, the Escritoire of Ashdown and Robinson also use a physical desktop as a tangible workspace for interaction with documents and media [2]. However, rather than augmenting real paper, the system strictly augments and manipulates virtual paper. Two overlapping projectors are used to create a "foveal display", where one projector covers the full field of view of the table, providing a low resolution display that establishes a "full visual context", while a second projector centered above the table is confined to a smaller frustum to provide the high resolution display needed for detailed work. The desk is a digitizer tablet, and interaction with virtual paper occurs with cordless pens held in each hand. Using the pens, virtual documents and images can be arranged (including document stacking) and annotated on the desk. Linked Escritoire desks in remote locations can also share a workspace for collaboration. The system supports traditional paper-desk interaction, where cues such as desk organization are used to remind people to perform certain tasks at certain times, or where grouping or sequencing information can be used for creating storyboards or other design layouts. One obvious advantage of the system is with the click of a button, a messy desk can "disappear", and the state of the virtual information and its current layout on the desk can be saved for a later period.

The Urp system of Underkoffler and Ishii from 1999 is an AR system that uses a tangible interface for urban planning [33]. Architectural models of buildings are placed on a table. Using a projector and camera, the system casts graphical shadows of the models onto the table based on a particular time of day, show reflections that would be cast from glass facades, and show a simulation of windflow that would occur at the base level of the buildings. Shadows corresponding to models are transformed accordingly as the models are moved and rotated. The position of the sun is interactively controlled by setting the hands of a clock tool. Urp is useful for pre-visualizing problems such as intershadowing between buildings, solar reflections that can create glare problems for drivers, and intense airflow that could make opening building doors too difficult. Urp exemplifies how "tangible interfaces give physical form to digital information, employing physical artifacts both as *representations* and *controls* for computational media...The physical forms of Urp's models (representing specific buildings), as well as their position and orientation upon the system's workbench, serve central roles in representing and controlling the state of the user interface" [32]. Urp is also well suited for collaborative work where design concepts and concerns can be immediately visualized and shared with other users.

In 2002, Piper et. al. presented the Illuminating Clay system, a $3D$ tangible interface for landscape analysis [25]. A projector renders the topography of a desired landscape onto a clay model. As users manipulate the clay, the changing geometry of the landscape is captured in real-time by a laser scanner, and a new topography

is computed to allow the projector to render an updated mapping onto the clay. Landscape planning requires input from multiple disciplines, including road builders, environmental engineers, and landscape designers, and thus this system is useful for collaborative work; a group of users can dynamically change the landscape to reflect their particular concerns while discussing and integrating the ideas of other users into their decisions.

In a similar vein to [25] and [33], Raskar et. al. describe Shader Lamps, where projectors are used to augment neutral-colored models (a miniature model of the Taj Mahal, for example) in order to represent the inherent color, texture, and material properties of the actual object [27]. A $3D$ graphical model of the object is rendered and aligned onto the surface of the physical model. An intensity blending algorithm for feathering and cross-fading between multiple projectors is also used for creating seamless imagery. This system supports an intuitive interface where a user can walk around the object and inspect it up close or at a distance, and serves well for collaborative work.

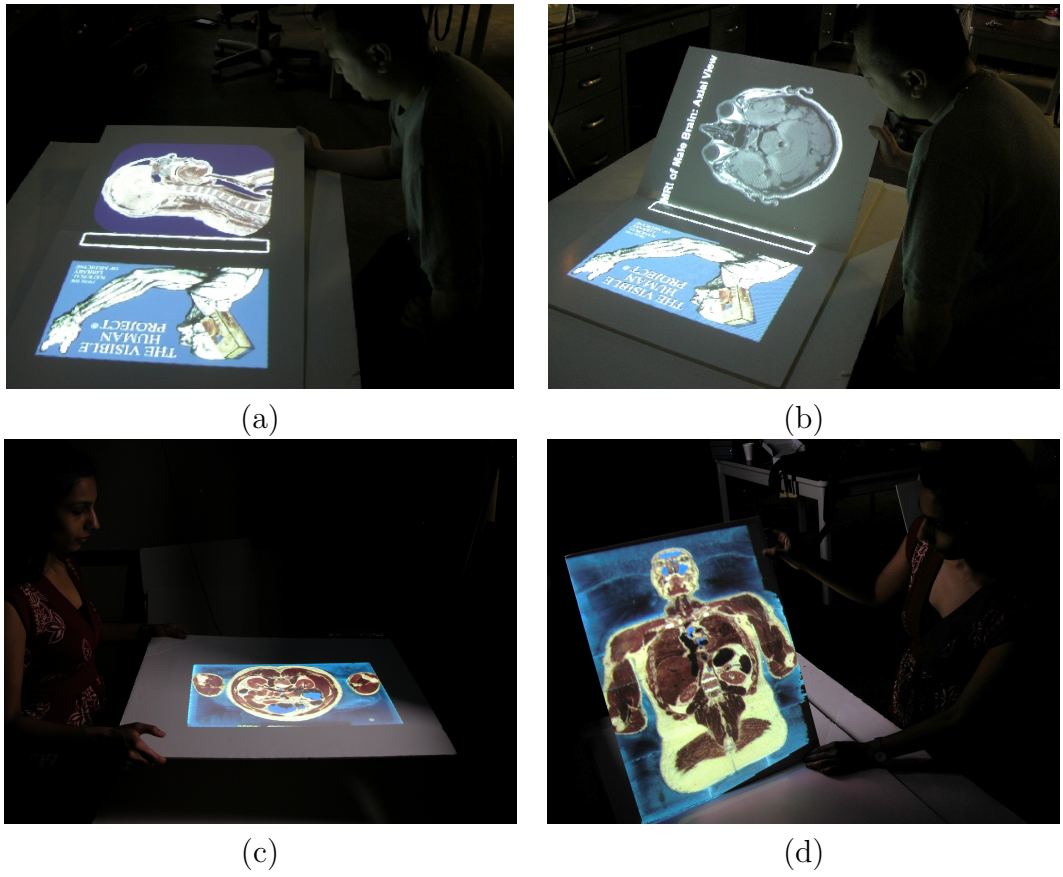
There has been a growing interest in using AR for information retrieval and guidance. In the Smart Bookshelf of Crasto et. al., for example, the state of a set of books on a bookshelf are monitored using a camera [8]. If a user wants to locate a specific book on the shelf, they enter a query into a database and a projector highlights the spine of the corresponding book so that it may quickly be retrieved by the user. Pinhanez et. al. use a steerable projector-camera device intended to help customers find products in a large store [24]. At the entrance of the store (or at other locations within the store), a user selects a department or product from an interactive menu. Once an item has been chosen, a sequence of arrows are projected onto boards hung from the ceiling that guide the user to the correct location of the product.

1.2 Project Overview

In this thesis, we have created an augmented reality system called the *Universal Media Book* (UMB). The UMB is a tangible interface that resembles a traditional book wherein a user can turn pages of a book back and forth in order to view the content contained on each page. The book itself is a bound collection of blank pages, and content is rendered onto the page surfaces using a projector. Pages are rigid for this project. The system automatically detects page turn events and accordingly updates the data to be displayed on a page.

The book can be used in two primary modes of operation. First, pages can contain multimedia (text, images, video) content that can be explored in the same manner as a regular book, where pages can be turned and content remains at a fixed position relative to the page. Secondly, a page can be removed from the book and used as a

tool to explore cross-sections of a virtual volumetric dataset at arbitrary orientations. Repositioning the page through the virtual volume reveals new slices of data correctly rendered onto the page. Figure 1.1 depicts the Universal Media Book being used to explore an image, a video, and a volume from the Visible Human Project [34].



(a) (b)

(c) (d)

Figure 1.1: A projector augments the Universal Media Book with data from the Visible Human Project. (a) A user looks at a sagittal view image of a face. (b) A video fly-through of axial MRI scans of a skull is played and remains correctly aligned to the page extents while a user turns the page. (c), (d) A user dynamically explores an anatomical volumetric dataset of a male cadaver through varying directions and orientations.

Computer vision techniques have been used to implement the UMB. Vision-based methods can non-invasively use the data from rich sources of image-based sensor inputs (video cameras, for example) to extract a variety of information about the world. Because computers can now efficiently process the large amounts of data generated by these images at real-time rates, computer vision is a natural choice for ubiquitous computing, where physical and digital information are merged to form reactive environments. Sensor devices do not have to be worn, as with an HMD, and can be placed in unobstructive locations in the environment.

In the UMB, a projector renders content onto the pages of a book while a video camera monitors the pages. The coupling of cameras with projectors has been a topic of interest for many years within the computer vision community, and has been used for such applications as $3D$ scene reconstruction, online monitoring and calibration of multi-projector displays, tracking positions of mobile projectors as they move through a scene, and real-time update of projected information as it augments moving objects. In the work presented here, a camera and projector pair are used to maintain an accurate estimate of the temporally coherent geometry of planar display surfaces in motion. Specifically, as pages are turned by a user, the location of page boundaries at each frame must be computed so that the projector may render content correctly registered to the page's current position and orientation in the world.

Tracking in the Universal Media Book is challenging for a number of reasons. Firstly, the display surfaces are featureless, so traditional approaches to tracking, where physical sensors, fiducials, or other surface markings are used to derive geometric information about the world cannot be used. The absence of physical tracking devices on the pages is important for creating the illusion of a "real" book. The underlying tracking system should remain as transparent to the user as possible to achieve the notion of ubiquity. Secondly, because projected data may change from one frame to the next (for example, when augmenting the surface with a video), assumptions about temporal consistency of features between successive frames cannot be exploited for tracking. Thirdly, the projector and camera have intrinsically different color spaces, and this can lead to error when trying to analyze and compare image content from each device simultaneously.

In our method, the contents of the projector's framebuffer are utilized independently in each frame by first converting the framebuffer to the color and geometric space of the camera, and then extracting stable feature points from the corrected framebuffer and camera images. Feature points are matched between the images and used to derive an estimate of the page surface geometry.

1.3 Contributions

The Universal Media Book constitutes an interface that synthesizes real-world objects with digital data. The interface itself is a familiar interface whose real-world counterpart is used on a daily basis by most people. The book integrates unobtrusively into the environment, and the AR system is able to react automatically to page movements.

Why is it even interesting and useful to augment a physical book? Firstly, a user's experience in interacting with data in a tactile or more visually stimulating way may be enhanced, compared to what could be a relatively boring experience us-

ing a standard $2D$ document or video viewer [6]. In addition, media not possible on a traditional book, such as video, becomes feasible. From an application standpoint, a book can be made to resemble a real book that is too fragile to be handled on an everyday basis. For example, [15] presents the idea of using a projector to augment a replica of the *Gutenberg Bible*, whose remaining copies are carefully stored in museums, protected from the public's touch due to its historical significance and fragility. This would allow users to experience what it was like in the mid-fifteenth century to feel and turn the pages of one of the world's first mass produced books, some of whose copies were printed originally on vellum. A practical utility of augmenting a blank book is that multiple books can be stored electronically, thereby reducing the amount of physical space needed to store texts. The omission of external markers promotes the use of a flexible, off-the-shelf interface where almost any set of boards may be used as pages for the book, and thus users do not have to be concerned with procuring and handling any specialized equipment.

In addition to the standard turning of book pages, the added functionality of interactive volumetric visualization provides the user with an intuitive interface for data navigation. The system is different than normal volumetric visualization systems that use a stationary monitor and mouse in that it gives the user direct physical control over the tool used to access parts the data. A page in the UMB serves as the tool that is held by the user, and is used to slice through a $3D$ dataset in order to view different cross-sections at arbitrary orientations. Using a mouse can sometimes be difficult when trying to access certain parts of a complex $3D$ dataset, especially when trying to smoothly transition through the data in a spatially coherent way. Using a direct tangible tool, the user can more intuitively explore the dataset, and can, for example, control such variables as the velocity at which they want to move from one point of the dataset to another.

The contributions of this thesis may be summarized as follows:

- A robust tracking framework that uses a camera and a projector to track the pages of a book without the use of any external features such as edge markers or infrared sensors.

- The Universal Media Book, an alternative interface for interacting with digital information that consists of a tangible book with blank pages augmented with multimedia data via a projector. The system automatically detects page turn events.

- An intuitive interface for viewing cross-sections of volumetric datasets from arbitrary orientations of the display surface.

The following chapter will describe previous work related to each of the aforementioned contributions.

Chapter 2

Related Work

2.1 Tracking and Surface Estimation

To correctly register computer-generated data with a real-world object, it is necessary to maintain a geometric estimate of the object's position and orientation. In the Universal Media Book, we need to continuously estimate the $3D$ parameters of planar pages and derive their location in the world so that the projector can render content correctly correctly aligned to the pages. Several approaches have been explored to recover the surface estimate of an object using a camera with computer vision techniques for AR systems. These approaches may be divided into marker-based and markerless techniques, where the former uses external sensors or fiducials mounted or projected onto the object for tracking the object's location, while the latter exploits the image content itself to derive the necessary geometric information.

In Borkowski et. al., a hand-held display screen with a dark border is augmented with a projected image [4]. A rigidly mounted steerable projector-camera system is used to transfer content projected on a large screen onto the smaller hand-held screen, automatically correcting for $3D$ translations and rotations as the screen is moved. A Hough transform is used to track the screen borders so that the four corners of the screen may be derived and a homography from screen coordinates to camera coordinates computed. Assuming a fixed camera to projector homography, the projector to screen mapping results from the composition of the two homographies. Although a steerable projector-camera unit increases the field-of-view of the application, the requirement of specialized equipment is not always feasible. With the UMB, we wish to use commodity elements that are easily obtainable. Also, the display surfaces of the UMB contain no markers, and feature detection occurs within the projected image itself.

Raskar et. al. describes an approach to recovering projector pose as the projection from a mobile projector-camera unit translates across a planar display surface [26]. Four laser pens attached to the unit are used to continuously project four fiducials onto the surface. The illumination from the pens are observed by the camera, and a homography from the camera image plane to the surface is computed. Assuming a calibrated projector-camera unit, and using an inertial sensor to find the rotation parameters of the mobile unit, the projector pose, up to an unknown translation, may

be computed.

The active pursuit algorithm of Gupta and Jaynes uses a calibrated static projector-camera setup, and, in a similar idea to [26], use the projector to render four fiducials onto a hand-held planar display surface that is in motion [10]. The system tracks these fiducials in order to recover the camera to projector homography. Black fiducials mounted onto the display surface are also tracked in order to define the extents of projection. The homography from the previous frame defines the search region for fiducial detection in the next frame.

Malik et. al. [19] and Simon et. al. [29] implement vision-based plane trackers for augmenting a video sequence. Although this type of augmented reality differs from the system we present here, wherein a physical object in the world is augmented using a projector, the tracking methods share similarities to the general framework of the UMB. [19, 29] detect feature points (Harris corners) on a plane (or planes) in the scene and use the robust statistical estimation technique of RANSAC to compute the homography that represents the transformation from a plane in the previous frame to the corresponding plane in the current frame. Before RANSAC is used in [29], correspondences between matchpoints from the previous and current frames are found using cross-correlation in a 7×7 window. After RANSAC, the pose of the camera is computed from the homography and a camera calibration matrix in order to align the real and virtual coordinate systems for scene augmentation. The major source of error is drift because the pose for each new frame is computed by successively multiplying the homographies from previous frames, causing error to accumulate over time. However, some techniques are presented to decrease this error. In [19], a patterned planar target (a black background embedded with white rectangles) is placed in a scene and tracked for augmentation with a $2D$ image or a $3D$ model. Before tracking begins, the system looks for four corners of a valid target region in a video frame, and then matches it to a corresponding canonical pattern using a homography from the target region extents to known world extents. During tracking, the previous frame's homography is used to predict the search windows used for corner detection in the pattern, and detected corners are used to compute a homography for the next frame. Given this homography, the boundary of the target may be augmented. The system can also handle changes in illumination because of the high contrast coloring of the target. [19, 29] and the UMB are all robust to partial occlusion because multiple features are available for tracking. In the UMB, the homography generated from matched feature points between the camera and projector are not used between frames for tracking, and unlike [19], temporal consistency between features cannot be exploited for tracking because it is assumed that scene contents may change in each frame.

Other works have considered markerless augmentation systems in the context of

a camera-projector setup. Cotting et. al. use a structured light technique that embeds imperceptible binary patterns into images projected by a DLP projector for tasks such as surface depth acquisition and target tracking [7]. Sectors in the DLP micro-mirror modulation sequences where mirrors are consistently on or off (a "clear" sector) are exploited for embedding the patterns (which may include an array of bars, a checkerboard, crosshairs, etc.). A camera synchronized to the projector analyzes the pattern at a precise interval, while the user does not perceive this pattern in the imagery due to the rate of projection. The system is useful for immersive and augmented environments because it is non-obtrusive and allows for simultaneous display and acquisition of data. Some disadvantages include slightly degraded imagery due to the embedded pattern and an involved setup phase (if trying to achieve maximal performance) needed for measuring light pulses of the projector.

Yang and Welch also operate on a projected image to correct for projector drift for a static display surface [38]. The technique gradually updates the shape of a display surface, which may be non-rigid, with respect to a projector that is augmenting it. Similar to the UMB, feature-based correlation is used to match the projector's contents to the image from a camera observing the scene. A Kalman filter estimates where a projector pixel should be on the display surface, and limits the feature search space in the camera image to a small region. When a feature correspondence is found, a $3D$ point on the display surface is estimated. Because the goal of the work is to automatically update the display surface geometry over an extended period of time and thus achieving a real-time response of the estimate is not necessary, only one sample point in the projected image is processed per frame. In our work, we must update the geometry of the display surface in each frame so the system can appropriately react to the constant movement of the surface.

The Everywhere Displays project of Pinhanez et. al. is capable of projecting information onto multiple display surfaces by steering a projector's frustum via a pan-tilt mirror [24]. Augmented information is correctly warped via planar homographies corresponding to the world planes that are to be illuminated. The approach assumes fixed display surfaces and generates a table of appropriate homographies for each pan-tilt configuration. Our method supports arbitrary placement of the display surface in the world (within the projector and camera frustums), and augmentation occurs dynamically for arbitrary configurations of the display plane.

2.2 Paper and Book Interfaces

Using paper-based media as an interface between physical and digital worlds "allows for greater flexibility in the way information is represented and stored, with a richer set of input techniques than currently possible with desktop displays" [16]. Such an

interface exploits users' familiarity with paper interaction and enhances the possibilities of using a tangible interface to acquire information in innovative ways.

Holman et. al. simulate a digital paper interface by using an overhead projector to render the windows found in a typical desktop windowing environment onto physical paper [16]. The idea is to make a desktop environment, which is typically confined to a monitor, more flexible and intuitive by allowing the user to spread the windows on a larger surface area, make direct annotations onto the paper using a stylus, and to manipulate the windows by folding and moving the paper in a way that is naturally done with real paper. The paper, as well as a stylus and the fingers, are mounted with infrared markers for tracking. A set of hand gestures, such as dragging a window from the monitor to a piece of paper, are learned by the system through a training phase before runtime.

Hirooka and Saito consider a projector-based augmentation of a real book with non-planar pages for the purpose of enhancing experiences with cultural heritage information [15]. The idea is to allow the user to perceive the tactile qualities, as well as the visual appearance, of an original text. To correctly project onto a page, piecewise homographies are used to model the page's deformation. Each page contains a set of printed fiducials that divides the page into rectangular regions and registers the page surface to the camera. The corner positions of a projected color checker pattern are used to register the page surface to the projector image plane. Using the homographies generated from these mappings, a digital image can be approximately warped to fit the shape of the page. The practical usability of the system is limited, however, because the color pattern must be projected each time the page shape changes, and thus a page cannot be continuously turned. In the UMB, pages can be turned continuously at interactive rates.

Billinghurst et. al. describe the MagicBook, wherein virtual models are overlayed onto the pages of a physical book to create an AR and VR experience [3]. The book pages contain standard flat images and text. However, when a user views the page through a hand-held display device, a virtual $3D$ model registered to the page appears to float on top of the page. The user may also immerse themselves in the virtual world and become an avatar in the computer-generated scene. To align the virtual model to the page, a camera (mounted onto the hand-held display) uses computer vision techniques to detect black borders surrounding the pages so that the position and orientation of the camera relative to the page may be computed. While the system supports different virtual media and interaction, the user-interface is in part abstracted away from the book itself and users interact directly with a hand-held display. In the UMB, our goal is to augment the capabilities of a physical book while retaining most aspects of traditional book interaction.

2.3 Volumetric Visualization

Using a standard monitor-keyboard-mouse interface for visualization of complex volumetric datasets is often challenging and un-natural because of the indirect manipulation of representational symbols used to access areas of the dataset. A number of tangible user interfaces have been designed for visualizing such datasets in ways that give more intuitive navigational control to the user. The idea proposed in each of these systems is to use physical hand-held objects, often termed *props*, as direct representational tools for extracting pieces of information from a volume.

An early work in two-handed interfaces for volume interaction is that of Hinckley et. al., wherein a set of small props, each mounted with tracking sensors, are manipulated by a user for neurosurgical visualization [14]. A user holds a ball, representing a patient's head, in the non-dominant hand and a cutting plane in the dominant hand. A $3D$ model of the patient's head is displayed on a monitor. Rotating the ball congruently changes the orientation of the computer model. Once a desired direction has been reached and locked into place, the user moves the cutting plane next to the ball at arbitrary orientations so that different cross-sections of the brain, corresponding to the plane's orientation, may be displayed on the monitor.

Although [14] considers interaction with familiar objects to allow users to utilize the spatial relationships defined by these objects as cues for intuitive visualization, the final output of data still remains confined to a monitor. In [1, 36], the display mechanism is moved from the monitor onto larger projection spaces that span a table or workbench. The StudyDesk of Wohlfahrter et. al. explores the use of props for interaction on a tabletop surface built from a rear-projection system [36]. A semi-transparent, semi-reflective plane and a plastic pen, each equipped with six degrees of freedom trackers, are held in each hand by a user, who also wears a tracked see-through head-mounted display for stereoscopic viewing. When the system is in $2D$ mode, the pen is used for selecting items from menus, manipulating graphical controls, and arranging items such that the projected data remains aligned to the plane as it is moved by the user. When switched to $3D$ mode, the pen is used to drag a projected volume to the desired position in space, and the plane (the transflective side) is used to slice through the volume so that cross-sections may be viewed at arbitrary orientations. When the reflective side of the plane is used for viewing, the user sees a reflection of the volume directly on the plane. Using this side of the plane allows for an effective increase in the viewing area of the physical display, as well as allows more untracked users to view the display with minimal distortion. In the Visual Interaction Platform of Aliakseyeu et. al., an overhead LCD projector renders data onto the surface of a table, while a back projection system is used for displaying onto a vertical space located at the end of the table [1]. Like the UMB, a volume sits

virtually at a user-defined location on the table. A rectangular plastic frame mounted with tracking sensors is used to slice through the volume for viewing cross-sections of the volume at arbitrary orientations. The cross-sections are rendered onto the table's surface as well as on the vertical projection space. To give the user a sense of spatial awareness of the tangible objects and the virtual volume, the vertical projection also contains a perspective rendering of the table and all its components, including a $3D$ graphical model of the volume and a graphical translucent plane representing the current location of the physical plane within the volume.

In recent work, the Deskrama of Nagakura and Oishi serves as an "interactive space browser" for viewing $3D$ models of architectural designs based on their $2D$ floor plans [20]. A user moves a lightweight LCD panel, embedded with a position and orientation sensor, along an architectural floorplan. A $3D$ model of the interior design is displayed on the screen based on the intersection of the floorplan with the panel. The author's goal is to create an intuitive viewing system that spatially synchronizes a user's movement of a surface through a $2D$ image space and its $3D$ representation. The panel, however, must remain aligned to the floorplan, and thus has limited degrees of freedom for movement.

Like the volume slicing mechanism in the Universal Media Book, each of the above interfaces give users direct physical control over the tools used to access specific data in volumetric datasets. Users employ natural hand movements and spatial cues to guide the tools to a desired area within the volume.

The volume slicing interface in the UMB differs from the other interfaces in two respects. Firstly, no external sensors are used for tracking. Instead, a simple rigid board is used as the interaction tool, and projected data itself is exploited for tracking purposes using robust estimation techniques. The interface tool is thus more readily available and requires no preparation. Also, users do not have to be concerned with obstructing sensors while manipulating the plane, which would affect the tracking results.

Secondly, in the UMB, a projector renders data *directly* onto the slicing plane at the correct perspective relative to the projector's viewpoint. The user's view and hand movements are focused on the same object throughout the interaction. The direct mapping of physical tool to virtual data gives the user an immediate physical visualization of the actual position and orientation of the cross-section within the virtual volume, unlike in [1], for example, where the cross-section is rendered onto the table, thus creating a somewhat disjoint mapping in the position and orientation of the plane and the final display of the actual cross-section. The idea of direct mapping is similar to [20], where the display output of the LCD panel lies directly above its physical location on the floorplan. [36] also shows the cross-section at the location of the plane in space, but a user is wearing a head-mounted display, and the

projection is physically occurring on the back-projection screen of the table. Also, because the projector renders directly onto the surface in the UMB, the physical workspace is more extensible to collaborative work where many people can observe the plane from different viewpoints without having to be tracked, and users do not have to look at a small screen or underneath a frame.

Chapter 3

Technical Details

In this system, a robust tracking framework that utilizes a monochromatic video camera and an LCD projector is used to monitor the state of an augmented page in the Universal Media Book while a user interacts with it. The content projected onto a page may consist of three types of data:

- Text or image: static content is rendered onto a page.
- Video: dynamic content where imagery changes in each frame is played onto a page.
- Volumetric dataset: cross-sections of volumetric data are rendered onto a page that has been removed from the book and that is moved at arbitrary positions and orientations.

The system operates as a continuous feedback loop where for each iteration, the projector first renders an image onto a page and the camera subsequently captures an image of it. Image search regions are extracted from each device, and the projector's framebuffer is geometrically aligned to the coordinate system of the camera, as well as photometrically corrected to resemble the colorspace of the camera. Feature points are extracted directly from the framebuffer and its corresponding camera image, and are correlated to form a set of matchpoints. A robust homography induced by the current display surface configuration in the world and that relates pixels in the camera to pixels in the projector is computed from the matchpoints. The $3D$ normal and extremal points of the page are then estimated with the help of pre-computed calibration information found from each device. Finally, the image to be rendered undergoes a perspective transformation and is then projected onto the page. See Figure 3.1 for an overview of the per-frame tracking system. The following sections describe each element of the system in detail.

3.1 Book Model

The physical book is a collection of pages made from a stack of rectangular white boards bound together on one side. The book is placed on a table within the frustums of a camera and projector pair. A rectangular outline representing the book's spine

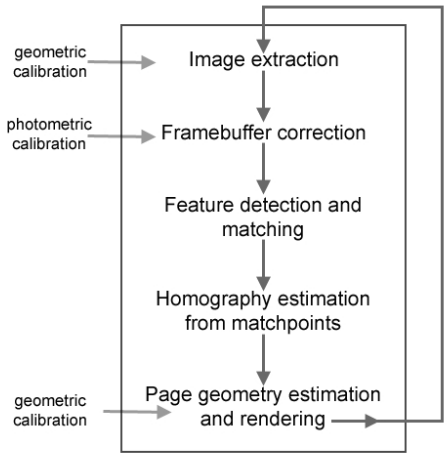


Figure 3.1: Overview of per-frame tracking framework.

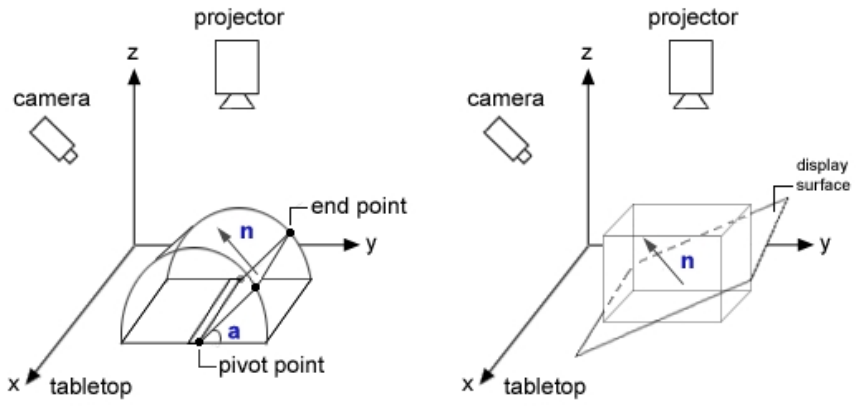
is projected onto the tabletop to serve as a guideline for where the book should be placed so that augmentation and tracking may begin.

The book consists of two modes of interaction. In Mode-1, pages are turned back and forth as in a regular book. In this case, each page is defined by four $3D$ coordinates. Two of the coordinates, called *pivot points*, remain fixed at the spine, while the other two corresponding coordinates, called *end points*, are constantly updated as a page is turned. Page turning is modeled as following a path along the curved edge of a cylinder that is embedded in the tabletop and whose major axis is parallel to the book spine (see Figure 3.2-a). In Mode-2, a page is removed from the book and moved at arbitrary positions and orientations within the frustums of the camera and projector. Here, the page extents are defined as the intersection of the page with a virtual volume whose base resides on the current page (see Figure 3.2-b). In each of these modes, the ultimate tracking goal is to maintain an estimate of the temporally coherent page normal $\mathbf{n}$.

A list of filenames linked to the content of each page are loaded into a simple configuration file before the system is used. In this file, a user may also specify the location of the book on the table, as well as the dimensions of the pages.

3.2 Offline Calibration

Before runtime, simple geometric and photometric calibration procedures are performed that feed into the system at runtime (see Figure 3.1). In the geometric procedure, the camera and projector are each calibrated to a world reference frame associated with the tabletop on which the book is placed. This information is used to



(a) (b)

Figure 3.2: Overview of the system components with the book model. Items in blue indicate parameters that are updated in each frame. (a) Mode-1: Text, images, and videos. (b) Mode-2: Cross-sections of volumetric data.

estimate the $3D$ position of page coordinates. This is a one time calibration phase, assuming the projector and camera remain stationary.

The photometric calibration procedure is performed to map projected color to camera intensity in order to increase the accuracy when matching feature points (see Section 3.3.3) detected in the camera and projector images.

3.2.1 Geometric Calibration: Background

The camera and projector pair function similarly to a standard stereo rig where two perspective cameras used together constrain the geometry of the imaged scene and allow the $3D$ structure of the scene to be inferred. Calibrating a camera involves computing the 3×4 projection matrix $\mathbf{P}$ that maps a homogeneous $3D$ point $\mathbf{X}$ in the world to its corresponding $2D$ image point $\mathbf{x}$ in the camera via a perspective projection transformation. The projection matrix is only defined up to scale. The mechanics of this projection are modeled off of the mechanics that govern an ideal pinhole camera, where exactly one light ray reflected from a point in the world passes through the camera's lens (the pinhole) and images onto a single point of the camera's image plane.

The camera's projection matrix is formulated from five intrinsic and six extrinsic parameters. The intrinsics relate the $2D$ pixel coordinates of the image to a normalized $3D$ coordinate system where the camera's origin, called the center of projection, sits at the origin of this coordinate system, and is oriented looking down the negative z-axis. The intrinsic parameters include the focal length of the camera (the distance from the center of projection to the camera image plane), the $2D$ coordinates of the

principal point (the point at which the camera's principal axis pierces the camera image plane), and the skew of the camera, which accounts for an image plane whose axes are not fully orthogonal. The extrinsics relate the position and orientation of the camera to a world origin, and consists of a $3D$ translation vector that translates the world origin to the camera's origin, and three rotation parameters that align the axes of the world coordinate system to those of the camera's coordinate system. Composing these two mappings into a single matrix $\mathbf{P}$ avoids the need to make explicit reference to the camera's normalized reference frame. A similar derivation is relevant for the projection matrix of a projector. However, inversely to the camera, a light ray travels from the center of projection of the projector to a point in the world. For in depth information on camera calibration and its related projective geometry, the reader is referred to [12].

When a point in the world is imaged in two devices simultaneously, the discrepancy in their image locations provides a cue for depth estimation. Specifically, given the 3×4 projection matrices $\mathbf{P}$ of the camera and $\mathbf{P'}$ of the projector, we seek the pixel locations of the world point $\mathbf{X}$ imaged in each device as $\mathbf{x} = \mathbf{PX}$ and $\mathbf{x'} = \mathbf{P'X}$. Ideally, a $3D$ point $\mathbf{X}$ is computed as the intersection of the rays in 3-space back-projected from each device's center of projection through the corresponding points $\mathbf{x}$ and $\mathbf{x'}$ via the projection matrices (see Figure 3.3). In reality, the rays do not intersect at a single point because there are errors in the measurements $\mathbf{x}$ and $\mathbf{x'}$ Using projectively invariant linear triangulation methods, the the equations $\mathbf{x} = \mathbf{PX}$ and $\mathbf{x'} = \mathbf{P'X}$ can be arranged into a system of linear equations $\mathbf{AX} = \mathbf{0}$ and solved to minimize the reprojection error (the distance between projections of an estimated $\mathbf{X}$ and the measured points $\mathbf{x}$ and $\mathbf{x'}$). $\mathbf{A}$ is composed as follows:

$$A = \begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p'}^{3T} - \mathbf{p'}^{1T} \\ y'\mathbf{p'}^{3T} - \mathbf{p'}^{2T} \end{bmatrix} \tag{3.1}$$

$\mathbf{p}^{i\mathbf{T}}$ is the i**th** row of $\mathbf{P}$, and x and y form an image coordinate pair. The solution of $\mathbf{X}$ will be the unit singular vector corresponding to the the smallest singular value of $\mathbf{A}$ [12].

3.2.2 Perspective Distortion

The perspective distortion induced by the projector's off-axis position with respect to a page must be taken into account during runtime in order to derive an image that remains rectified to the display surface in the world. The mapping that corrects this distortion is composed into a 4×4 projection matrix $\mathbf{T}$ that performs a perspective transformation and a scaling and shifting, and describes the projector's view volume.

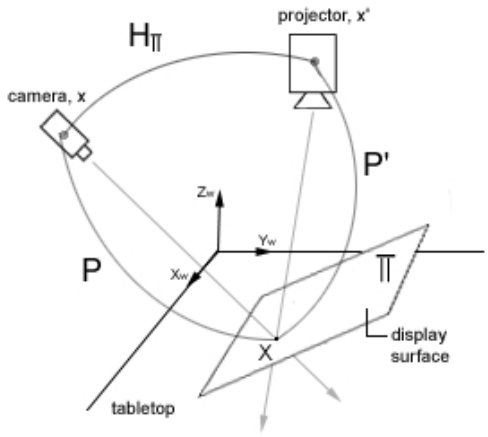


Figure 3.3: In an ideal setup, rays projected from each device's center of projection through respective image points $\mathbf{x}$ and $\mathbf{x}'$ intersect at world point $\mathbf{X}$.

The perspective transformation maps a $3D$ point to another $3D$ point, and describes how a world point is projected onto a screen point. The original $3D$ point is already in the coordinate system of the projector, as specified by a viewing transformation $\mathbf{V}$ that transforms world coordinates into the projector's coordinate system (specifically, $\mathbf{V}$ specifies the projector's position, view direction (the projector's optical axis is oriented towards the center of the tabletop), and the up vector in the world). Matrix $\mathbf{T}$ specifies the bottom-left and top-right points of the near clipping plane of the volume (assuming the projector's world origin is located at coordinate $(0, 0, 0)$), as well as the location of the far clipping plane. These parameters are derived from the intrinsics of the projector. (See [37] for more information on the graphics rendering pipeline). During runtime, at each frame, the perspective projection is applied to the framebuffer contents before projecting onto the display surface.

3.2.3 Homography: Background

Another important transformation that is computed independently in each frame during runtime, and that is also used to aid in the offline geometric calibration phase is a homography, or collineation. A 3×3 homography is a non-singular projective transformation that maps homogeneous points in $2D$ space to homogeneous points in another $2D$ space (projective space $\mathcal{P}^2$). A homography is defined only up to scale, and encodes a series of transformations, including a translation, rotations, uniform and non-uniform scale, shears, and a projection. The homography transforms general quadrilaterals in a source space to quadrilaterals in a destination space.

Given a set of four or more point correspondences between two images, the eight free parameters of a 3×3 homography matrix can be computed using the direct linear

transform (DLT) method [12]. The DLT seeks to solve a homogeneous system of equations $\mathbf{Bh} = \mathbf{0}$ under the constraint that $||\mathbf{h}|| = \mathbf{1}$. A 2×9 matrix $\mathbf{B_i}$ is computed for each camera and projector correspondence $\mathbf{i}$ (see [12]). The matrices $\mathbf{B_i}$ are stacked to form a single matrix $\mathbf{B}$. The singular value decomposition (SVD) of $\mathbf{B}$ is computed, and the unit singular vector corresponding to the smallest singular value of $\mathbf{B}$ is taken to be the solution to $\mathbf{h}$. The nine values of this vector may then be written as the 3×3 homography matrix $\mathbf{H}$.

3.2.4 Calibration Procedure

To calibrate each device, a calibration target made of two planar boards joined orthogonally is placed on the tabletop (see Figure 3.4). Each plane consists of a set of 48 black Gaussian fiducials arranged as a grid. The $3D$ location of each fiducial on the target is known and is measured in millimeters (the origin $(0, 0, 0)$ of the target, which is also the world origin, is set as the lower left corner of the horizontal plane).

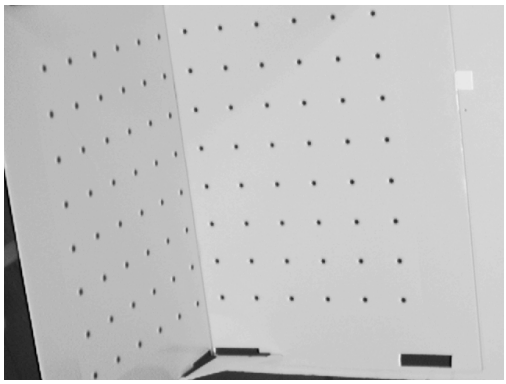


Figure 3.4: Calibration target with two orthogonal planes.

Calibration begins by first computing the homography $\mathbf{H}_1$ that maps pixels in the camera's image to pixels in the projector's framebuffer for the plane on the target that is parallel to the tabletop. To obtain the correspondences needed to compute the homography, a set of known projector points are illuminated sequentially while a camera locates those points via a simple scanline search. The same procedure is followed to compute the homography $\mathbf{H}_2$ for the orthogonal plane of the calibration target.

Next, an image of the target is captured by the camera and binarized in order to isolate the center of mass of each fiducial. The user interactively clicks each fiducial detected in the camera image in order to assign it to a known world correspondence. Now, a set of correspondences exist between each fiducial on the target and pixels in the camera image. To find a projector correspondence $\mathbf{p}$ to a world fiducial $\mathbf{X}$, we

can simply compose the prior knowledge of the camera to projector homography $\mathbf{H_i}$, for $\mathbf{i} = \mathbf{1}, \mathbf{2}$, with the world fiducial to camera pixel mapping, $\mathbf{C}$, for each plane of the target as follows:

$$p = H_i C X. \tag{3.2}$$

Given the world to camera and world to projector correspondences, the eleven intrinsic and extrinsic parameters of each device are computed using Tsai's camera calibration algorithm, which is based on the pinhole model of perspective projection [31]. The 3×4 projection matrices of each device are then composed from these parameters [12].

3.2.5 Photometric Calibration

Many factors contribute to the non-linear response in measured camera values of projected color, including sensor filters, sensor noise, intrinsic gamma correction, gain and shutter speed settings, ambient light in the environment, projector "black offset" (a black pixel of the projector that does not correspond to a measured value of black in the camera), and so forth. In addition, the material properties of the display surface, as well as its position and orientation with respect to each device, influence how light is reflected into the camera.

The complex non-linear relationship in observed intensity in the camera and color values in the projector is approximated using a lookup table (LUT) for each of the red, green, and blue color channels of the projector. We modify the photometric correction scheme of [13] to work for monochromatic camera images. For each channel $\mathbf{j}$, where $\mathbf{j} \epsilon \{\mathbf{r}, \mathbf{g}, \mathbf{b}\}$, the intensity $\mathbf{i}$ in the range $[\mathbf{0}, \mathbf{255}]$ is projected onto the tabletop one at a time, while the intensities of the other two channels are locked at $\mathbf{0}$. (In the case where the color of the book surface is different than that of the tabletop, the illumination should occur on the book surface itself because that is what the camera images during runtime.) The average pixel value $\mathbf{c_j(i)}$ seen by the camera for each projected intensity is stored in the channel's LUT (see Figure 3.5). At runtime, each projector pixel's $\mathbf{R}$, $\mathbf{G}$, and $\mathbf{B}$ value is converted and used to form a corrected pixel $\mathbf{p_{corr}}$ via the LUTs and a black offset factor $\mathbf{L}$ using the following equations:

$$p_{corr} = c_r(R) + c_g(G) + c_b(B) - 2 * L \tag{3.3}$$

$$L = (c_r(0) + c_g(0) + c_b(0))/3 \tag{3.4}$$

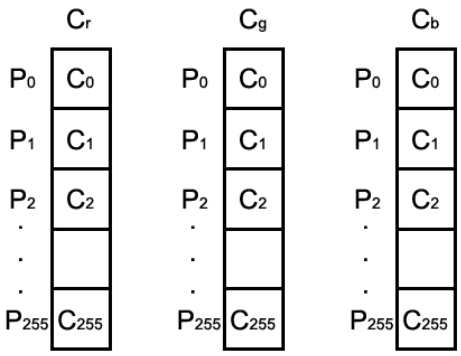


Figure 3.5: Monochromatic lookup tables for the (a) red channel, (b) green channel, and (c) blue channel.

Figure 3.6 shows an example of a framebuffer image that has been corrected using the LUTs. This figure also shows the framebuffer after conversion using a standard RGB to grayscale conversion. These results show that the LUT correction produces a framebuffer image that lies in a closer color space to that of the camera.

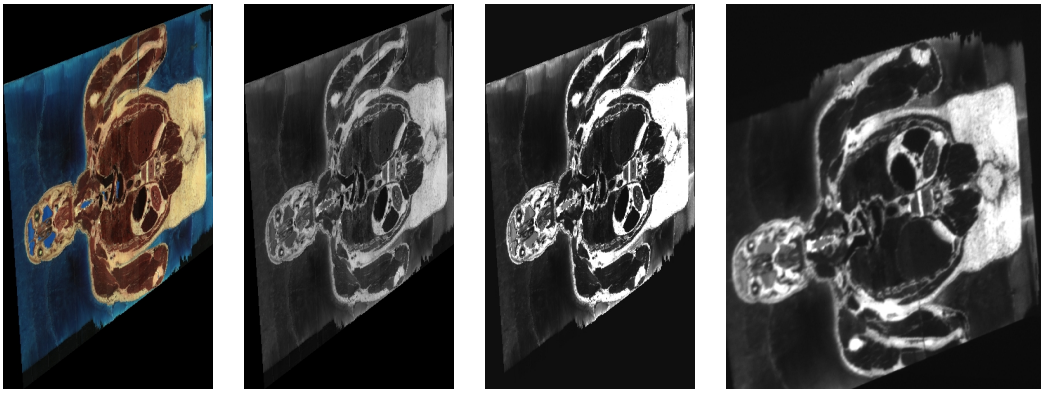


Figure 3.6: Photometric correction of projector's framebuffer to camera color space. (a) Original framebuffer. (b) Converted framebuffer using standard RGB to grayscale mapping. (c) Converted framebuffer using monochromatic LUTs. (d) Observed camera image.

3.3 Tracking Pipeline

Given the information obtained from the offline calibration phases, we are now ready to begin runtime processing (see Figure 3.1). Feature detection is an iterative process that makes use of the previous frame's geometric estimate to search for new correspondences in the projector and camera images. These correspondences lead to a new estimate of the geometry that propagates forward as the book pages are moved.

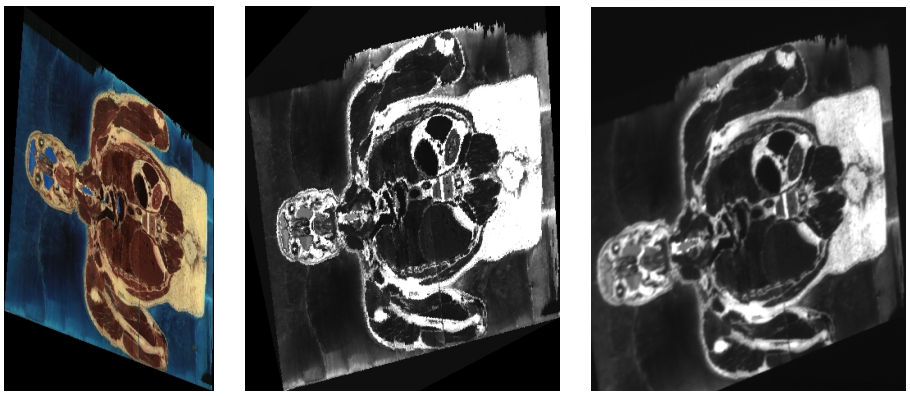
3.3.1 Image Extraction and Framebuffer Correction

Before feature extraction occurs in the current frame, we need to define search spaces for image processing. The four $3D$ coordinates produced in the previous frame that define a page are first rectified to their $2D$ camera and projector correspondences using the projection matrices of each device via $\mathbf{x} = \mathbf{PX}$ and $\mathbf{x}' = \mathbf{P}'\mathbf{X}$. These coordinates define quadrilaterals in each image. Rather than using the full resolution images of each device, image processing is restricted to these subimages to avoid unnecessary processing in regions where no projected information is present. Once these images are extracted, the projector subimage is intensity corrected using the color-to-intensity LUTs as described in Section 3.2.5.

The geometric distortion between projector and camera frames can become especially large in a wide baseline setup or for significant orientations of the display surface, and must be taken into account prior to feature extraction. This is particularly important when using simple yet fast-to-compute features that are not invariant to scale change such as the Harris operator used here (see Section 3.3.2). To compensate for this, the current framebuffer subimage is warped to the camera subimage using a $2D$ homography, $\mathbf{H_{sub}}$, that is computed from the four extremal points of each subimage. For computational efficiency, nearest neighbor interpolation is used when warping the framebuffer. Figure 3.7 shows an example of the framebuffer contents geometrically and photometrically brought into alignment with the camera image prior to feature extraction.

3.3.2 Feature Detection

Now that search regions are defined, feature point extraction is applied to the camera and corrected projector subimages. Feature point detection within an image is often used in computer vision to derive various pieces of information for such tasks as image matching, object recognition, 3D reconstruction, and tracking. Feature point detectors seek to find salient points in an image, where salient can mean a point "for which the signal changes 2-dimensionally" [28]. Such points can be found at various classes of corners in the image, as well as in highly textured regions.



(a) (b) (c)

Figure 3.7: Photometric and geometric correction of the projector image using the LUTs and $\mathbf{H_{sub}}$, respectively, to approximately align it with the camera image prior to feature extraction and matching. (a) Original projector framebuffer. (b) Corrected framebuffer. (c) Observed camera image captured at that frame.

In this work, we use the Harris corner detector [11]. This rotationally invariant feature detector is an intensity based method that detects *corners* in an image, where a corner may be defined as a point in the image whose gradient is large in multiple directions surrounding the point. The Harris detector works by defining a window and shifting this window by small amounts in multiple directions around a point and looking for large changes in intensity values caused by the shifts.

Specifically, a change $\mathbf{E}$ in the intensity $\mathbf{I}(x, y)$ for a pixel (x, y) by a shift (u, v) for a window $\mathbf{w}$ may be defined as:

$$E_{u,v} = \Sigma_{\vee a,b \; in \; w}[I(x+a, y+b) - I(x+u+a, y+v+b)]^2 \tag{3.5}$$

To account for an infinite number of possible discrete shifts of the window around the pixel (x, y), the above equation can be formulated into a 2×2 autocorrelation matrix. The autocorrelation matrix $\mathbf{U}$ may written as follows:

$$U = \begin{bmatrix} (\Sigma dx)^2 & \Sigma dxdy \\ \Sigma dxdy & (\Sigma dy)^2 \end{bmatrix} \tag{3.6}$$

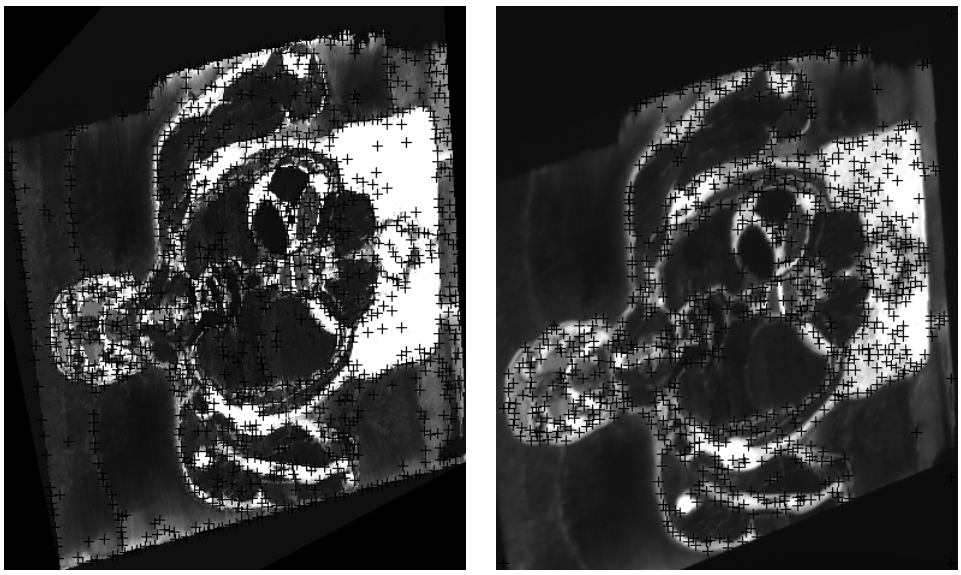
dx and dy are the gradients of a pixel in the horizontal and vertical directions, respectively, and the sum for each term occurs within the window around the pixel of interest. The eigenvalues λ_1 and λ_2 of matrix $\mathbf{U}$ serve as a description of the neighborhood of the pixel under consideration. If both eigenvalues are small, the point is found in an area with little intensity variability, or a flat region. If one eigenvalue is significantly larger than the other, the point is found on an edge, where window shifts in directions perpendicular to the edge give rise to large intensity changes, and

window shifts in directions along the edge give rise to small intensity changes. Finally, if both eigenvalues are large, the point is considered to be a corner, where shifts in all directions around the point give large changes in intensity values. To avoid explicitly computing the eigenvalues of **U**, the quality of a corner response **R** may be computed with an empirical constant **k** as

$$R = determinant(U) - k * trace(U) \tag{3.7}$$

To detect Harris corners, we use the optimized real-time feature detection framework of [22]. The implementation makes use of the MMX instruction set found on Intel Pentium processing units, as well as optimizes for cache performance and processing time. The image is convolved with a binomial filter of the form [14641] before the corner responses are computed, and the the constant **k** is set equal to 0.06.

Non-maximal suppression is performed on a 5×5 neighborhood around each feature point. In non-maximal suppression, a feature point is included in the final set if its corner response **R** is larger than all other corner responses in the window surrounding the feature point under consideration. To prevent an over-saturation of feature points, a maximum of 5000 feature points, distributed in 10×10 buckets of the image, may be detected. Figure 3.8 shows the features extracted from a framebuffer and camera image pair.



(a) (b)

Figure 3.8: Detection of Harris corners, as indicated by the black crosshairs. (a) Features extracted in the color corrected and warped framebuffer. (b) Features extracted in the corresponding camera image.

3.3.3 Matching

Each feature point in the projector image is matched to feature points in the camera image. The problem of matching features between images is known as the correspondence problem, and is of fundamental importance to many computer vision related tasks where finding accurate correspondences can lead to robust geometrical estimates. Matching is a difficult problem because many differences may be present between two images of the same scene, including large changes in viewpoint that lead to images that are not geometrically similar, image noise due to camera sensors, non-linear changes in illumination caused by the intrinsic of a device, non-linear changes in illumination caused by external lighting conditions, and so forth.

To find the best candidate match, normalized correlation in an 11×11 window is used at potential corresponding points in each image. The standard formulation for normalized correlation for two image patches $\mathbf{I_1}$ and $\mathbf{I_2}$ over corresponding points (s, t) defined within the image patches is as follows:

$$\gamma = \frac{\Sigma_s \Sigma_t (I_1 - \hat{I}_1)(I_2 - \hat{I}_2)}{\sqrt{\Sigma_s \Sigma_t (I_1 - \hat{I}_1)^2 \Sigma_s \Sigma_t (I_2 - \hat{I}_2)^2}}, \tag{3.8}$$

where $\hat{\mathbf{I}}_\mathbf{1}$ and $\hat{\mathbf{I}}_\mathbf{2}$ are the means of the intensity values in each patch. The value of the normalization coefficient γ produces values scaled within the range -1 to 1, where values closer to 1 yield a high probability of a good match and values near -1 yeild a low match probability. Using normalized correlation accounts for changes in illumination across patches by considering changes in brightness (subtracting the mean from each intensity value) and changes in contrast.

It is in the matching phase that photometric calibration (see Section 3.2.5) becomes important. Although normalized correlation is partially invariant to affine intensity changes in the image (for example, for the image $\mathbf{I}$, the change $\mathbf{aI} + \mathbf{b}$, where $\mathbf{a}$ is a scale factor and $\mathbf{b}$ is a shift factor), the complex non-linear relationship in color mappings between a camera and a projector are not modeled.

Using the matching implementation of [22], the 11×11 patches centered around the feature points in each image are first vectorized and stored consecutively in memory as a 128-byte vector. A subset of the terms in equation 3.8 are pre-computed for each patch in each image. Then, the scalar product between two image patches $\mathbf{I_1}$ and $\mathbf{I_2}$ is computed efficiently using MMX instructions. The correlation coefficient is found by plugging these pre-computed values into equation 3.8

The correlation coefficient of each feature point in the projector's warped subimage is checked against every feature point in the camera's subimage that lies within a certain disparity limit of the projector point under consideration. In our setup, we use a disparity limit of 10%. The larger the distortion between two images, the larger the disparity limit may need to be. A match is finalized when it passes a mutual

consistency check. In this check, a correspondence will be approved if and only if the point in the camera image that yields the highest correlation score with the projector image point under consideration mutually chooses the same projector point. The set of matched projector points are unwarped by $\mathbf{H_{sub}^{-1}}$ to re-align them to their original coordinate system.

3.3.4 Sources of Error

There are a few sources of error that can lead to erroneous matchpoints between features. Firstly, the camera is in focus for only a range of planes in the world, and the inherent camera sensor noise, as well as the low resolution of the camera image compared to that of the projector causes a course sampling of the projected image.

Secondly, when the projector renders data, depending on the orientation of the display surface with respect to the optical axis of the projector, and given the projector's limited depth of field, a projector pixel will hit the display surface with varying degrees of spread, rather than at a precise point. This causes adjacent pixels to meld together, creating *circles of confusion* which will lead to an even further degradation of the projected imagery from the camera's perspective.

Finally, the geometric warp of the framebuffer by $\mathbf{H}_{sub}$ is interpolated using nearest neighbor interpolation on an intensity-corrected image, and so the Harris corner detection in the framebuffer is occurring in a resampled space. Also, when the framebuffer is unwarped by $\mathbf{H_{sub}^{-1}}$, the Harris corners detected in the warped image may not correspond to their exact location in the original framebuffer due to sub-pixel error incurred during the transformation.

All of these facts limit the degree of reliable correlation coefficient estimates, and may cause the system to produce mismatches, or outliers, between feature points in each image. So, given the set of matchpoints, we use robust statistical methods to find the best set of correspondences, or inliers, that can lead to the best estimate of the geometric entity we are interested in. In our case, we seek a homography.

3.3.5 Robust Homography Estimation

In each frame, we seek a new homography $\mathbf{H_\Pi}$ induced by the display surface that best relates pixels in the camera to pixels in the projector. The homography must be re-computed in each frame because as the display surface moves, the geometric estimate derived from the previous configuration of the system is no longer valid.

A standard approach to parameter estimation is the robust statistical method of RANSAC [12]. Given a set of datapoints (observations), a set of hypotheses describing the desired geometric entity are generated from the minimal number of observations needed to describe the model, and the hypothesis yielding the highest number of

observations that are inliers is considered the best estimate. For example, if we are trying to estimate the best $2D$ line that fits a set of $2D$ datapoints, a hypothesis is formed from two randomly selected points in the dataset, because only two points are needed to describe a line. Specifically, the RANSAC algorithm proceeds as follows:

1. Randomly select the minimum number of data points needed to describe the model and form a hypothesis.

2. Test each datapoint against this hypothesis, and compute the error in each point's fit to the model. If the error falls below a predefined error threshold, the datapoint is an inlier for this hypothesis. Otherwise, it is an outlier.

3. If the number of inliers exceeds a predefined threshold, stop and use the current hypothesis as the best one. Otherwise, repeat steps 1 and 2 until the best inlier count is reached.

4. As a final step, refine the model by re-estimating the hypothesis using all the inliers.

Rather than using RANSAC, we use the Preemptive RANSAC method of [21] to estimate a homography from the matchpoints. Preemptive RANSAC is a robust preemptive scoring scheme that discovers the best hypothesis of a minimal model given a set of observations. In our case, a four-point minimal hypothesis set is needed to compute a homography. The algorithm generates scores that describe the log likelihood of an observation with each of the hypotheses. After this scoring is performed with a subset of observations, the hypotheses with the best scores are retained and sorted, while the other hypotheses are eliminated. The process is repeated until only a single hypothesis remains. The method, as outlined in [21], may be summarized as follows:

1. Generate a fixed number of hypotheses $h = 1, .., M$ by randomly permuting the observations $o = 1, .., N$.

2. Compute the scores $L_1(h) = \rho(1, h)$ for $h = 1, .., f(1)$ for a subset of the hypotheses and set $i = 2$. $f(i) = \lfloor M * 2^{\lfloor \frac{1}{B} \rfloor} \rfloor$ indicates how many hypotheses are to be kept at each stage, and B is the block size that governs when hypotheses are to be reordered. The scoring function $\rho(o, h)$ returns a scalar value representing the log likelihood $L(h)$ of the hypothesis indexed by h: $L(h) = \Sigma_{o=1}^{N} \rho(o, h)$.

3. Reorder the hypotheses so that the range $h = 1, .., f(1)$ contains the best $f(i)$ remaining hypotheses according to $L_{i-1}(h)$.

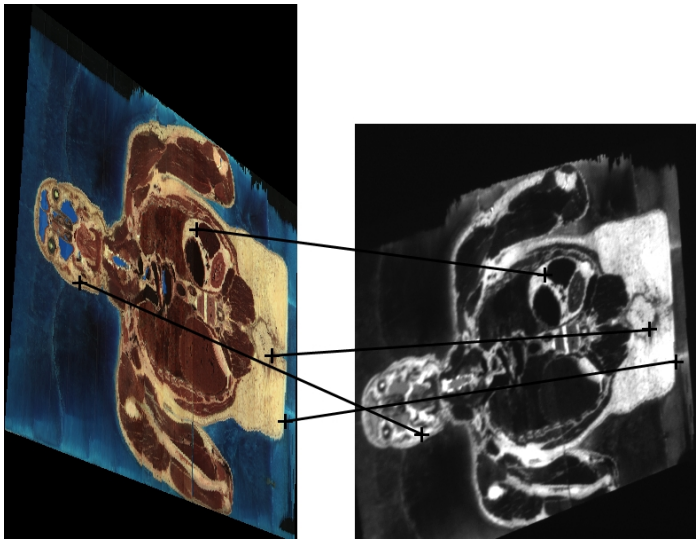


Figure 3.9: The four corresponding points selected by the Preemptive RANSAC algorithm. The correspondences relate the projector framebuffer to the current camera image for a particular frame.

4. Quit with the best remaining hypothesis if $i > N$ or $f(i) = 1$; otherwise, compute the scores $L_i(h) = \rho(i, h) + L_{i-1}(h)$ for $h = 1, .., f(1)$; increase i and then repeat step three.

Figure 3.9 shows the four points that led to the best homography estimate $\mathbf{H}_\Pi$ that were selected from the set of points shown in the projector and camera images of Figure 3.8

In Preemptive RANSAC, hypotheses are weighed against each other rather than by a global quality measure used in the RANSAC framework. The method is efficient and useful for systems requiring real-time response rates because each hypothesis does not have to be tested against each observation. The preemption function dictates that only a subset of hypotheses are maintained after a set of iterations that test only one observation at a time.

Before accepting the final hypothesis, the Euclidean distance between each of the four points used to compute the homography is evaluated to ensure that the separation between each point falls above a minimum threshold in order to avoid potentially degenerate estimates computed from points spaced too closely together. If this does not hold, the Preemptive RANSAC routine is iterated again.

3.3.6 Page Estimation and Rendering

To find the location of the page in the world, the four $2D$ matchpoints used to compute the homography are reconstructed to their their $3D$ correspondences using

each device's projection matrix as described in Section 3.2.1. The page's normal vector is then computed by taking the cross product of two vectors formed from the $3D$ points. The dot product between the normalized page normal vector **n** and the up vector **w** in the world (the normal to the surface of the table, which is also the z-axis) is used to compute the orientation angle **a** of the page with respect to the table (see Figure 3.2-a):

$$a = arccos\frac{(n \cdot w)}{(||n||||w||)} \tag{3.9}$$

The coordinates of the page endpoints are then computed from the radius (page width) of the cylinder and the angle **a** using the following equations:

$$end_z = pivot_z + radius * (sin)(a) \tag{3.10}$$

$$end_y = pivot_y + radius * (cos)(a) \tag{3.11}$$

Once the page endpoints are known, they are linearly interpolated with the previous frame's page endpoints in order to smooth the tracking results.

Before rendering projected content, the perspective transformation detailed in Section 3.2.2 is applied to the framebuffer contents in order to derive an image that will remain rectified in the world.

3.4 Page Turn Detection

A page turn event is detected when the normal vector **n** of the current page is found to lie orthogonal to the up vector in the world. At this point, the data is deleted from the page, the projector goes blank for a predefined amount of time in order to allow the user to make a complete page turn, and new data is augmented onto the next page whose initial normal vector is set as the world up vector (see Figure 3.10). The depth value (z-coordinate) of the two pivot coordinates are also updated to reflect the changing thickness of the book after a page is turned.

Pages can also be turned backwards to view the previous page's content. This is accomplished by projecting an image on the left page and continuously tracking its normal in the same way described for tracking the right page's normal. A page turn in the reverse direction is detected when the page's normal becomes approximately parallel to the optical axis of the camera. At this point in our setup, the camera can no longer observe the page's content due to occlusion.

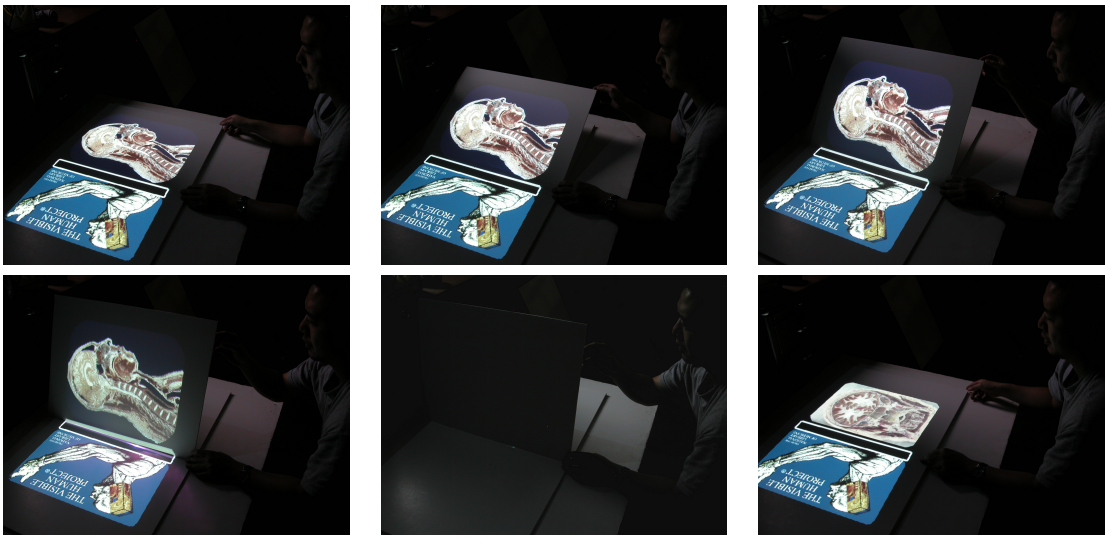


Figure 3.10: Page turn sequence leading to a page turn event.

3.5 Volumetric Rendering

If a page contains a volumetric dataset, the user removes the current page from the book and uses it as a display surface to dynamically navigate through a virtual volume in arbitrary positions and orientations (see Figure 1.1-c,d). A user simply needs to reposition the page within the volume to view a new cross-section of the volume.

The virtual volumetric dataset is defined to lie in the world residing on the current page. The dataset is represented as a 3D texture-mapped cube of size 1 in an object coordinate system. Using the 3D texturing capabilities now available on most commodity graphics hardware, an arbitrary slice of data can be extracted from the volume by defining texture coordinates (that correspond to world coordinates) within the unit cube. Because the page is no longer bound to the book spine and can move freely in the world, the world coordinates needed to define a planar patch are found as the intersection points of the plane with the volume extents (see Figure 3.2-b). These intersections points are linearly interpolated with the previous frame's intersection points in order to smooth the tracking results. Figure 3.11 shows multiple orientations of the display surface from the camera's perspective as it slices through the virtual volume.

When a page is displaying a volumetric slice, it is necessary to disambiguate a page turn event (see Section 3.4) from an orientation of the page that has been removed for volumetric viewing. To do this, a user simply needs to press a predefined keyboard button which causes the system to delete the volume and display a static image on the page, wherein normal page turning may proceed.

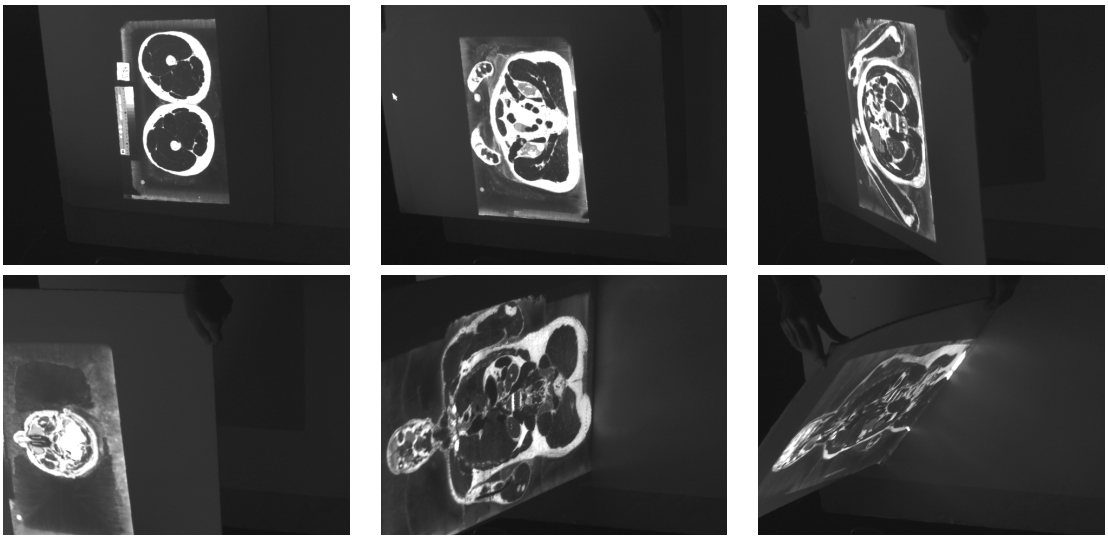


Figure 3.11: Volumetric cross-sections from a 579 frame run as seen from the camera's perspective. The numbers of each frame in the sequence are (from left to right and top to bottom): 1, 30, 84, 139, 206, 262.

Chapter 4

Experimental Results

4.1 Setup

The system consists of the following items:

- One Dell XPS 600 running Linux, 3.8GHz, 2GB RAM, 107GB hard drive, on-board IEEE 1394 6pin, 512 MB 7900 GTX Graphics Card.
- One Sony XCD-710 Firewire Camera, transmitting 640x480 monochromatic images at 30fps.
- One ViewSonic LCD Projector, 1028x764 native resolution with VGA input.
- Two tripods.
- A set of boards that make up the book interface.

The projector and camera are each mounted on tripods that are set 1.5 meters apart. Such a wide baseline setup is necessary to achieve accurate results for $3D$ point reconstruction from stereo images (see Section 3.2.1). The system is implemented in C++ and uses the OpenGL API for graphics rendering and the libdc1394 API for camera control.

4.2 Evaluation

Table 4.1 lists the average time of each major process in the system. Times were collected over three trials, with each trial running for 200 frames. For each trial, two images (shown in Figure 4.1) were tracked on the right hand side pages of the book, with one page turn event between the images. The search spaces for image processing in the camera and projector subimages were set to 80% of the subimage extents. The overall system framerate when tracking only the right page is **11.80** frames per second, while the overall framerate when tracking both the left and right pages is **9.95** frames per second.

From these results, we can see the tracking bottleneck is in the wait and capture steps. The idle waiting in step 2 is necessary in order to ensure that the image captured by the camera consists only of the newly projected data. In other words,

Figure 4.1: Two images used for obtaining average runtimes for Table 4.1.

Process	**Time (ms)**
1 Display image (perspective projection, $3D$ texture mapping, glXSwapBuffers, glFinish, XSync)	**0.26**
2 Wait (ensure full refresh of newly projected image)	**24.00**
3 Capture camera image	**43.17**
4 Image processing	**14.20**
a Extract projector image (glReadPixels)	0.78
b Get predicted projector image	0.26
c Extract camera image	0.15
d Warp projector image to camera image	3.43
e Detect features in projector image	1.67
f Detect features in camera image	1.68
g Match features in projector and camera images	5.01
5 Compute homography using Preemptive RANSAC	**1.21**
6 Reconstruct four world points from homography	**0.04**

Table 4.1: Average time for each of the major processes in the system.

we need to fully account for the response time of the LCD projector to allow a full screen refresh and to avoid any ghosting effects in the image, which could compromise tracking results.

Within the tracking pipeline, the framebuffer warp (step 4d) and the feature matching (step 4g) account for the two largest processing times. As mentioned in Section 3.3.1, the warp is necessary because we are using a feature detector that is not invariant to changes in scale. Another detector that could have been used that is invariant to scale changes is SIFT [18]. With this detector, the framebuffer would not have to be warped to the camera image, and the accuracy when matching features might increase. However, more involved steps are required to arrive at a description of a SIFT feature point. Comparing each of these methods in our system is the subject of future work.

It should be noted that a more standard approach to compute the page geometry would be to RANSAC on the parameters of the book's page normal directly. We tested this approach and it does not yield any improvements to the robustness or accuracy of the estimated normal, and the processing time is slower than the proposed method. This is partly because each of the variable number of matchpoints detected in a frame must undergo a complete reconstruction in order to derive the world correspondences needed to form the normals.

Assuming a page is in constant motion, the data augmented onto a page is always rendered at world coordinates that were derived from the previous frame. This is because the page coordinates estimated from projected content in the current frame are used to set the location for projected content in the next frame. Thus, the rendering always occurs at a one frame delay. However, because the system runs at interactive rates and because of the inherent coherency in image data (i.e. images, videos, and volumes), this difference is often not perceivable when the system is in use.

4.3 Limitations

The current system has some limitations. One drawback of relying solely on the the content of projected data to infer world geometry is that if the data does not contain enough textured information, the system may not be able to detect enough matchpoints, and a robust estimate of the geometry may not be able to be computed. This can cause the system to lose track of the page. To alleviate this, if enough matchpoints cannot be found, the projector could render a set of know projector points into the scene, and using the last known homography, search spaces can be initialized in the camera to track the points using a temporal tracking algorithm. Matchpoints can be formed and a homography computed until enough textured content is present. Alternatively, a textured border could be projected onto the page based on the last known valid page location and tracked using the current method.

Another problem is that the resolution of the projected image starts to degrade at orientations where the display surface is almost parallel to the projector's optical axis. Also, the projector is only in focus for a limited depth of field. These problems could be alleviated by introducing more projectors into the system for an increased resolution of the image and greater coverage of the workspace. Also, when the board is oriented at a significant angle away from the user, the user cannot see the projected data. Again, this could be alleviated by using more projectors, or by using a semi-transparent plane for the display surface.

Although the photometric calibration method used works well for the system, it is only a valid estimate for the plane on which the lookup tables are formed.

Considering an adaptive color correction method that takes into account the display surface's orientation could produce more accurate results.

Chapter 5

Conclusions and Future Work

We have introduced a tracking framework that maintains accurate alignment of a projected image to a moving planar display surface that is monitored by a camera. The problem is challenging in that the display surfaces are blank and features can change from frame-to-frame as the projector augments the scene. Thus, traditional methods of tracking external fiducials or sensors mounted onto the display surface, or tracking temporally coherent feature points, does not apply. Instead, features are extracted and matched in the projector and camera images in each frame independently. Because many features are detected in each frame, the system is robust to partial occlusion (see Figure 5.1). Results demonstrate that the tracking algorithm is capable of supporting a real-world, mixed-reality interface called the Universal Media Book, which also includes an intuitive interface for volumetric visualization. Because we do not mark the book pages with fiducials, the underlying tracking system remains invisible to the user, which is important when simulating an interactive environment that should be natural and intuitive, and that integrates well into the user's surroundings.

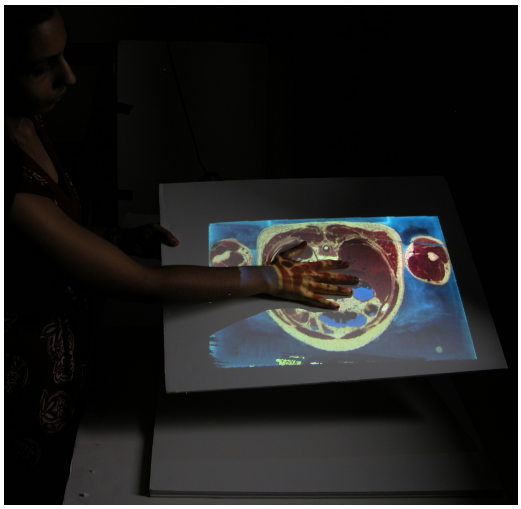


Figure 5.1: Tracking under occlusion.

In the future, we would like to allow the user to orient the virtual volume to any desired position during runtime to provide more viewing flexibility. Also, we would like to increase the mobility of the book interface by allowing the base of the book

to be moved to any location during runtime that is still within the frustums of the camera and projector. For example, the user could move the book to his or her lap to view and turn the pages in the same way one might do with with a regular book.

In addition to the medical datasets we have demonstrated in the paper, another interesting and useful application for the volumetric viewing tool would be to render a volume of a cone so that math students, for example, could use the display plane to slice through the volume in order to see how conic sections are formed by the intersection of a cone with a plane. Equations reflecting the changing parameters of the conic as the plane is moved through the volume could also be rendered onto the display surface.

A near term extension of this application will be to support view-dependent rendering of three-dimensional objects that have parallax outside of the moving plane. Augmentation will not be restricted to planar slices and will support visualization of arbitrary models that sit on the planar surface as it moves. The use of more projectors and cameras can help to increase the resolution, as well as overall coverage of the display so that detection of page turn events, for example, do not have to be limited to the frustums of a single camera and projector. We would also like to perform an extensive set of user studies on the system. Ultimately, the project is driven by our desire to actively illuminate deformable surfaces that are in motion in real-time.

Bibliography

[1] D. Aliakseyeu, S. Subramanian, J.-B. Martens, and M. Rauterberg, *Interaction techniques for naviagation through and manipulation of 2d and 3d data*, in Eight Eurographic Workshop on Virtual Environments, 2002.

[2] M. Ashdown and P. Robinson, *The escritoire: a personal projected display*, in WSCG, 2003.

[3] M. Billinghurst, H. Kato, and I. Poupyrev, *The magic book: a transitional ar interface*, in Computers and Graphics, 2001.

[4] S. Borkowski, O. Riff, and J. Crowley, *Projecting rectified images in an augmented environment*, in Procams, 2003.

[5] W. Buxton, *Living in augmented reality: ubiquitous media and reactive environments*, in Video mediated communication, 1997.

[6] Y. Chu, D. Bainbridge, M. Jones, and I. Witten, *Realistic books: a bizarre homage to an obsolete medium?*, in International Conference on Digital Libraries, 2004.

[7] D. Cotting, M. Naef, M. Gross, and H. Fuchs, *Embedding imperceptible patterns into projected images for simultaneous acquisition and display*, in ISMAR, 2004.

[8] D. Crasto, A. Kale, and C. Jaynes, *The smart bookshelf: a study of camera projector scene augmentation of an everyday environment*, in IEEE Workshop on Applications of Computer Vision, 2005.

[9] C. Cruz-Neira, D. Sandin, and T. DeFanti, *Surround-screen projection-based virtual reality: the design and implmentation of the cave*, in SIGGRAPH, 1993.

[10] S. Gupta and C. Jaynes, *Active pursuit tracking in a projector-camera system with application to augmented reality*, in Procams, June 2005.

[11] C. Harris and M. Stephens, *A combined corner and edge detector*, in Fourth Alvey Vision Conference, 1988.

[12] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2000.

[13] M. Hilario and J. Cooperstock, *Occlusion detection for front-projected interactive displays*, in Pervasive Computing and Advances in Pervasive Computing, Austrian Computing Society, 2004.

[14] K. Hinckley, R. Pausch, J. Goble, and N. Kassell, *Passive real-world interface props for neurosurgical visualization*, in ACM CHI Human Factors in Computing Systems, 1994.

[15] S. Hirooka and H. Saito, *Virtual display system using video projector onto real object surface*, in International Conference on Artificial Reality and Telexistence, 2004.

[16] D. Holman, R. Vertegaal, M. Altosaar, N. Troje, and D. Johns, *Paperwindows: interaction techniques for digital paper*, in CHI 2005, 2005.

[17] C. Jaynes, B. Seales, K. Calvert, Z. Fei, and J. Griffioen, *The metaverse - a collection of inexpensive, self-configuring, immersive environments*, in 7th International Workshop on Immersive Projection Technology/Eurographics Workshop on Virtual Environments, 2003.

[18] D. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60 (2004).

[19] S. Malik, G. Roth, and C. McDonald, *Robust tracking for real-time augmented reality*, in Vision Interface, 2002.

[20] T. Nagakura and J. Oishi, *Deskrama*, in Siggraph Emerging Technology, 2006.

[21] D. Nistér, *Preemptive ransac for live structure and motion estimation*, in IEEE International Conference on Computer Vision, 2003.

[22] D. Nistér, O. Naroditsky, and J. Bergen, *Visual odometry*, in IEEE Computer Society Computer Vision and Pattern Recognition, 2004.

[23] K. Pimentel and K. Teixeira, *Virtual reality: through the new looking glass*, McGraw-Hill, 1995.

[24] C. Pinhanez, R. Kjeldsen, A. Levas, G. Pingali, M. Podlaseck, and N. Sukaviriya, *Applications of steerable projector-camera systems*, in Procams, 2003.

[25] B. Piper, C. Ratti, and H. Ishii, *Illuminating clay: a 3-d interface for landscape analysis*, in Chi, 2002.

[26] R. Raskar, P. Beardsley, J. Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher, *Rfig lamps: interacting with a self-describing world via photosensing wireless tags and projectors*, in SIGGRAPH, 2004.

[27] R. Raskar, G. Welch, K. Low, and D. Bandyopadhyay, *Shader lamps: animating real objects with image based illumination*, in Eurographics Workshop on Rendering, 2001.

[28] C. Schmid, R. Mohr, and C. Bauckhage, *Evaluation of interest point detectors*, International Journal of Computer Vision, 37(2) (2000), pp. 151–172.

[29] G. Simon, A. Fitzgibbon, and A. Zisserman, *Markerless tracking using planar structures in the scene*, in Augmentned Reality (ISAR), 2000.

[30] I. Sutherland, *Sketchpad: a man-machine graphical communication system*, in Technical Report, 2003, 1963-phd thesis.

[31] R. Tsai, *A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses*, IEEE Journal of Robotics and Automation, RA-3, No. 4 (1987), pp. 323–344.

[32] B. Ullmer and H. Ishii, *Emerging frameworks for tangible user interfaces*, in Human-computer interaction in the new millenium, (by J. Carroll), Addison-Wesley, 2001, pp. 579–601.

[33] J. Underkoffler and H. Ishii, *Urp: a luminous-tangible workbench for urban planning and design*, in CHI, 1999.

[34] Visible Human Body, *www.nlm.nih.gov/research/visible/visible_human.html*.

[35] P. Wellner, *The digitaldesk calculator: tangible manipulation on a desk top display*, in ACM SIGGRAPH Symposium on User Interface Software and Technology, 1991.

[36] W. Wohlfahrter, L. Encarnacao, and D. Schmalstieg, *Interactive volume exploration on the studydesk*, in The Fourth International Immersive Projection Technology Workshop, 2000.

[37] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL programming guide*, Addison-Wesley, 1999.

[38] R. Yang and G. Welch, *Automatic projector display surface estimation using every-day imagery*, in 9th International Conference in Central Europe on Computer Graphics Visualization and Computer Vision, 2001.

Vita: Shilpi Gupta

Background

- Date of Birth: January 5, 1981
- Place of Birth: Stafford, Texas

Education

- Master of Science in Computer Science, University of Kentucky, 2006. Anticipate completion and defense of thesis in December 2006.
- Bachelor of Science in Computer Science, University of Kentucky, 2003.
- Bachelor of Arts in Art Studio (with emphasis in Photography) and Minor in French Language, University of Kentucky, 2003.

Professional Experience

- January 2004 - September 2006. Research Assistant at Center for Visualization and Virtual Environments. University of Kentucky, Lexington, KY.
- September 2003 - December 2003. English Language Teaching Assistant in Primary Schools. Dijon, France.
- July 2002 - September 2003. Internship at Metaverse Lab, Department of Computer Science. University of Kentucky, Lexington, KY.
- September 2001 - May 2002. Photolab Technician at Medical Arts and Photography. Lexington, KY.

Publications

- S. Gupta and C. Jaynes, *The Universal Media Book: Tracking and Augmenting Moving Surfaces with Projected Information*, in ISMAR, 2006.
- S. Gupta and C. Jaynes, *Active Pursuit Tracking in a Projector-Camera System with Application to Augmented Reality*, in Procams, 2005.