LOW LATENCY DISPLAYS FOR AUGMENTED REALITY

Peter C. Lincoln

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2017

Approved by:

Henry Fuchs

Greg Welch

Oliver Bimber

Anselmo Lastra

Montek Singh

Bruce Thomas

Turner Whitted

© 2017
Peter C. Lincoln
ALL RIGHTS RESERVED

ABSTRACT

Peter C. Lincoln: Low Latency Displays for Augmented Reality
(Under the direction of Henry Fuchs and Greg Welch)

The primary goal for Augmented Reality (AR) is bringing the real and virtual together into a common space. Maintaining this illusion, however, requires preserving spatially and temporally consistent registration despite changes in user or object pose. The greatest source of registration error is latency—the delay between when something moves and the display changes in response—which breaks temporal consistency. Furthermore, the real world varies greatly in brightness; ranging from bright sunlight to deep shadows. Thus, a compelling AR system must also support High-Dynamic Range (HDR) to maintain its virtual objects' appearance both spatially and temporally consistent with the real world.

This dissertation presents new methods, implementations, results (both visual and performance), and future steps for low latency displays, primarily in the context of Optical See-through Augmented Reality (OST-AR) Head-Mounted Displays, focusing on *temporal consistency in registration*, *HDR color support*, and *spatial and temporal consistency in brightness*:

1. For *registration temporal consistency*, the primary insight is breaking the conventional display paradigm: computers render imagery, frame by frame, and then transmit it to the display for emission. Instead, the display must also contribute towards rendering by performing a post-rendering, post-transmission warp of the computer-supplied imagery in the display hardware. By compensating in the display for system latency by using the latest tracking information, much of the latency can be short-circuited. Furthermore, the low latency display must support ultra-high frequency (multiple kHz) refreshing to minimize pose displacement between updates.

2. For *HDR color support*, the primary insight is developing new display modulation techniques. DMDs, a type of ultra-high frequency display, emit binary output, which require modulation to produce multiple brightness levels. Conventional modulation breaks low latency guarantees, and modulation of bright

LEDs illuminators at frequencies to support kHz-order HDR exceeds their capabilities. Thus one must directly synthesize the necessary variation in brightness.

3. For *spatial and temporal brightness consistency*, the primary insight is integrating HDR light sensors into the display hardware: the same processes which both compensate for latency and generate HDR output can also modify it in response to the spatially sensed brightness of the real world.

To my family and friends

ACKNOWLEDGMENTS

First, I would like to thank my advisors Henry Fuchs and Greg Welch, whom have guided me and supported me through all these years. I would also like to thank my other committee members for their support and advice: Oliver Bimber, Anselmo Lastra, Montek Singh, Bruce Thomas, and Turner Whitted.

In my time at UNC, I have worked on several papers, both on this thesis' topic and others, and I would like to thank my co-authors as well: Alex Blate, Juan Cendan, David Chesnutt, Henry Fuchs, Adrian Ilie, Anselmo Lastra, Andrew Maimone, Arjun Negendran, Andrew Nashel, Remo Pillat, Diego Rivera-Guiterrez, Ryan Schubert, Montek Singh, Andrei State, Herman Towles, Greg Welch, Turner Whitted, Mary C. Whitton, and Feng Zheng. I would also like to thank the other people who helped on projects related to those papers or other Research Assistantship (RA) work, including Jonathan Bidwell, Mingsong Dou, Yinan Fang, Jan-Michael Frahm, Zachary Kaplan, Mod Marathe, Bill Mauchly, and Ying Shi. I would also like to specifically thank Kurtis Keller, Jim Mahaney, John Thomas, and Herman Towles for their engineering support on each of my projects at UNC. I would also like to thank Alex Blate for collaborating on the Low Latency Displays project as well as designing and fabricating some of the electronics hardware necessary for making the display work. I would also like to thank the UNC Computer Science Department staff for administrative support over the years.

I would like to thank several people for sitting in on photos used in this thesis: Fay Alexander, Steve Partin, Brett Piper, and Hope Woodhouse in Figure 1.2; and Brett Piper in Figure 4.8.

While working as an RA for Henry Fuchs on the Low Latency Displays project, I have been funded by multiple sources, which I would like to thank:

- National Science Foundation Award CHS IIS-1423059: "Minimal-Latency Tracking and Display for Head-Worn Augmented Reality Systems," Dr. Lawrence Rosenblum, Program Manager

- The BeingThere Centre, a collaboration between Eidgenössische Technische Hochschule (ETH) Zürich, Nanyang Technological University (NTU) Singapore, and the University of North Carolina (UNC) at Chapel Hill. The Centre is supported by these three institutions and by the Singapore National

Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media Programme Office.

- The BeingTogether Centre, a collaboration between Nanyang Technological University (NTU) Singapore and University of North Carolina (UNC) at Chapel Hill. The BeingTogether Centre is supported by the National Research Foundation, Prime Minister's Office, Singapore, under its International Research Centres in Singapore Funding Initiative.

While working as an RA for Henry Fuchs and Greg Welch on other projects, I have been funded by multiple sources, which I would also like to thank:

- Multi-View Telepresence and Telepresence Wall
  - CISCO
- Avatar
  - Office of Naval Research Award N00014-09-1-0813: "3D Display and Capture of Humans for Live-Virtual Training," Dr. Roy Stripling, Program Manager
- Multi-Projector Blending
  - Office of Naval Research Award N00014-09-1-0813: "3D Display and Capture of Humans for Live-Virtual Training," Dr. Roy Stripling, Program Manager
  - National Science Foundation Award CNS-0751187': "IAD Integrated Projector-Camera Modules for the Capture and Creation of Wide-Area Immersive Experiences," Dr. Lawrence Rosenblum, Program Manager

PREFACE

This dissertation is primarily based on the following publications: Lincoln et al. (2016) and Lincoln et al. (2017), which have appeared in the indicated peer-reviewed journal or conference proceedings:

1. Lincoln, P., Blate, A., Singh, M., Whitted, T., State, A., Lastra, A., and Fuchs, H. (2016). From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1367–1376

2. Lincoln, P., Blate, A., Singh, M., State, A., Whitton, M., Whitted, T., and Fuchs, H. (2017). Scene-adaptive high dynamic range display for low latency augmented reality. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, New York, NY, USA. ACM

TABLE OF CONTENTS

LIST OF TABLES xiii

LIST OF FIGURES xiv

LIST OF ABBREVIATIONS xvi

1 INTRODUCTION 1

1.1 Vision 1

1.2 Motivation 2

1.2.1 Latency 3

1.2.2 Dynamic Range 4

1.3 Thesis Statement 5

1.4 Contributions 5

1.5 Structure 6

2 BACKGROUND 7

2.1 Introduction 7

2.1.1 Units 7

2.2 Related Work 7

2.2.1 Existing Near-Eye OST-AR Displays 7

2.2.2 Registration Error 9

2.2.3 Latency Requirements 9

2.2.4 Latency Mitigation Strategies 10

2.2.4.1 GPU Improvements 11

2.2.4.2 Frameless Rendering 11

2.2.4.3 Post-Rendering Warp 12
2.2.5 High-Dynamic Range 13
2.3 DMD Operations 14
2.3.1 DMD Chip Basics 14
2.3.2 Conventional DMD Projector Basics 15
2.4 Modulation 16
2.4.1 Pulse Width Modulation 17
2.4.2 Pulse Density Modulation 19
3 LOW LATENCY RENDERING 21
3.1 Introduction 21
3.1.1 Contributions 22
3.2 Render Cascade Pipeline 22
3.3 Prototype: Simplified Render Pipeline 24
3.3.1 Components and Implementation 25
3.3.2 Visual Results: 6 bit/pixel Grayscale Implementation 31
3.3.2.1 Experimental Setup 31
3.3.2.2 Discussion 32
3.3.3 Visual Results: 16 bit/color HDR Implementation 34
3.3.4 Latency Analysis: 6 bit/pixel Grayscale Implementation 34
3.3.4.1 Signal Path 35
3.3.4.2 Measurement Instrumentation 36
3.3.4.3 Latency Component Analysis 37
3.3.4.4 Latency Range Analysis 39
3.3.5 Latency Analysis: 16 bit/color HDR Implementation 40
3.3.5.1 Signal Path 41
3.3.5.2 Measurement Instrumentation 41
3.3.5.3 Latency Component Analysis 42

3.3.5.4 Latency Range Analysis 43

4 HIGH-DYNAMIC RANGE FOR LOW LATENCY DISPLAYS 45

4.1 Introduction 45

4.1.1 Contributions 46

4.2 Display Scheme: Estimated Error Modulation 46

4.2.1 Results 47

4.2.2 Limitations 47

4.3 Display Scheme: Pseudorandom Pulse Density Modulation 49

4.3.1 Results 50

4.3.1.1 Perceptual Comparison of PWM and PR-PDM 51

4.3.2 Limitations 53

4.4 Display Scheme: Direct Digital Synthesis 53

4.4.1 Color Calibration 56

4.4.2 Results 58

4.4.3 Limitations 61

4.5 Scene-Aware Spatial Brightness Control 63

4.5.1 Results 64

4.5.2 Limitations 65

4.6 Summary 68

5 SUMMARY AND CONCLUSIONS 71

5.1 Summary 71

5.2 Future Work 72

5.2.1 Inertia 72

5.2.2 Latency and General Pose Tracking 73

5.2.3 Render Cascade Pipeline 73

5.2.4 High(-er) Dynamic Range 74

5.2.5 Scene Brightness Detection 75

5.2.6 Video Latency 76
5.2.7 Human Perception Studies 76
5.2.8 Alternative Spatial Light Modulators 77
5.3 Conclusions 77
BIBLIOGRAPHY 79

LIST OF TABLES

1.1 Example forms of error in AR systems 3

3.1 Summary of latency components for the 6 bit/pixel system 39

3.2 Summary of latency components for the 16 bit/color system 44

4.1 Summary of color mixing parameters for the 16 bit/color system's LEDs 58

4.2 Computed color calibrated 16 bit/color DAC codes 59

4.3 Comparison of modulation technique requirements 69

4.4 Comparison of modulation technique behavior 69

5.1 Comparison of implementation performance 72

LIST OF FIGURES

1.1 The reality-virtuality continuum 2

1.2 A sample scene of people in a room with HDR brightness 5

1.3 A user "wearing" our AR display 6

2.1 Sample comparison of simulated 4 bit gray level modulation schemes (PWM and Delta-Sigma) by method and desired intensity level 18

3.1 A sample, conventional render pipeline 22

3.2 Two sample PRW pipelines 23

3.3 The PRW pipeline implemented in our prototype 25

3.4 System assembly for the 6 bit/pixel system 26

3.5 System assembly for the 16 bit/color system 27

3.6 Implemented data path framework for image generation for the 6 bit/pixel grayscale system .. 28

3.7 Subset of implemented data path diagram for image generation for the 16 bit/color system ... 29

3.8 A comparison between conventional displays and our latency compensation algorithm using the 6 bit/pixel mode 33

3.9 A comparison between our latency compensation algorithm and a conventional display using the 16 bit/color mode 35

3.10 Light measurement sensor and barrier 36

3.11 Sample motion-to-photon latency measurement of the 6 bit/pixel system 38

3.12 Accumulated samples of motion-to-photon latency measurement of the 6 bit/pixel system 40

3.13 Sample motion-to-photon latency measurement of the 16 bit/color system 42

3.14 Accumulated samples of motion-to-photon latency measurement of the 16 bit/color system .. 44

4.1 Input and output imagery for stable tests with the EEM scheme 48

4.2 Output imagery for tests with the EEM scheme where the input imagery varied over time 48

4.3 A sample frame using our latency compensation algorithm under fast panning motion 50

4.4 Measured 6 bit grayscale modulation patterns for PR-PDM by desired intensity level 52

4.5 Simulated signals and spectra of 6 bit PWM and PR-PDM for targeting a constant DC signal 54

4.6 Implemented data path diagram for the 16 bit/color system 57

4.7 Per-color measured light sensor data for each 16 bit DAC code 59

4.8 Panorama of demonstration setup used with the 16 bit/color system 60

4.9 A raw GPU screenshot used as input for testing DDS 60

4.10 An HDR pair of colored virtual teapots as recorded through the display 62

4.11 A raw GPU screenshot used as input to the illumination algorithm. 64

4.12 The augmented scene illuminated by bright flashlights 66

4.13 The augmented scene illuminated by a desk lamp 67

4.14 Sample comparison of simulated 4 bit grayscale modulation schemes (PWM, Delta-Sigma, PR-PDM, and DDS) by method and desired intensity level 70

LIST OF ABBREVIATIONS

1080p	Ten-Eighty Progressive Scan, 1920×1080
2D	Two-Dimensional
3D	Three-Dimensional
$\Delta\Sigma$	Delta-Sigma
ALS	Ambient Light Sensing
AR	Augmented Reality
ASIC	Application Specific Integrated Circuit
AV	Augmented Virtuality
CIE	International Commission on Illumination (Commission internationale de l'éclairage)
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DAC	Digital-to-Analog Converter
DC	Direct Current
DDS	Direct Digital Synthesis
DLP®	Digital Light Projection
DMD	Digital Micromirror Display
DOF	Degree of Freedom
DP	DisplayPort™
DRAM	Dynamic Random Access Memory
DVI	Digital Video Interface
EEM	Estimated Error Modulation
FOV	Field of View
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit

HDMI	High-Definition Multimedia Interface
HDR	High-Dynamic Range
HMD	Head-Mounted Display
HUD	Heads-up Display
IBR	Image-based Rendering
IC	Integrated Circuit
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
LFSR	Linear-Feedback Shift Register
LSB	Least Significant Bit
MCP	Mirror Clocking Pulse
MR	Mixed Reality
MSB	Most Significant Bit
OLED	Organic Light-Emitting Diode
OS	Operating System
OST	Optical See-through
PC	Personal Computer
PDM	Pulse Density Modulation
PR-PDM	Pseudorandom Pulse Density Modulation
PRW	Post-Rendering Warp
PTM	Pulse Train Modulation
PWM	Pulse Width Modulation
RAM	Random Access Memory
RC	Resistor-Capacitor
RGB	Red, Green, and Blue
SAR	Spatial Augmented Reality
SI	International System of Units (Système international d'unités)

SLM	Spatial Light Modulator
SoC	System on a Chip
SRAM	Static Random Access Memory
TI	Texas Instruments
VGA	Video Graphics Array
VR	Virtual Reality
VST	Video See-through
VSYNC	Vertical Sync
XGA	Extended Graphics Array, 1024×768

CHAPTER 1

INTRODUCTION

1.1 Vision

The worlds of the real and the synthetic are colliding more often now than ever before. The year 2016 has seen major consumer releases of several consumer-grade Virtual Reality (VR) headsets (*e.g.*, Oculus Rift CV1[1], HTC Vive[2], and PlayStation VR[3]), and many other companies are also working on releasing VR headsets of their own. In addition, with the proliferation of smartphones around the world, mobile VR headsets (*e.g.*, Samsung Gear VR[4], Google Cardboard[5], and Google Daydream[6]) are also gaining some traction. When users put these these devices on their heads, each of these systems "transport" people elsewhere in the world or into other worlds; however, they also disconnect users from reality by hiding it behind the obscuring goggles. In VR, only virtual objects are visible; the real world need not exist. In Augmented Virtuality (AV), while virtual objects comprise the majority of the combined scene, some real objects are also present.

On the other hand, Augmented Reality (AR)—another form of Mixed Reality (MR) (*see* Figure 1.1)—which mixes the virtual world into the real, is beginning to overshadow VR. AR adds virtual objects into the real world. The iOS and Android game *Pokémon Go*[7] has been downloaded over 500 million times by people all around the world (Lynley, 2016). While other games have offered virtual imagery overlaid onto live video of the real world, *Pokémon Go* has been the first to achieve that level of popularity, enabling its users to wander the world while looking through their phone screen and camera to find life-size virtual creatures.

[1] https://www.oculus.com/rift/

[2] https://www.vive.com/

[3] https://www.playstation.com/en-us/explore/playstation-vr/

[4] http://www.samsung.com/global/galaxy/gear-vr/

[5] https://vr.google.com/cardboard/

[6] https://vr.google.com/daydream/

[7] http://www.pokemongo.com/

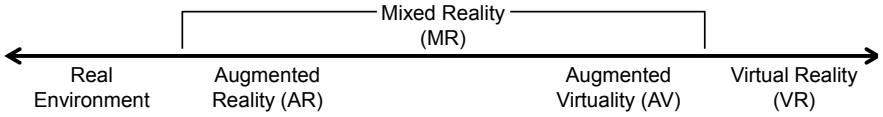


Figure 1.1: The reality-virtuality continuum, originally proposed by Milgram et al. (1995).

Games like these are example uses of Video See-through Augmented Reality (VST-AR) displays, where the augmenting virtual imagery is overlaid on top of a live, tracked video stream as captured by a camera. The user is no longer looking directly at the world; a display sits in-between and presents both at the same time. This gives the system a great level of control: while the user is free to move, the system can briefly delay the "live" video to allow for processing and rendering time.

Optical See-through Augmented Reality (OST-AR) displays, however, let users continue to look directly at the real world; the virtual imagery is optically combined with the real, which greatly increases the complexity. Optical combination, without any matting, often results in the virtual imagery appearing translucent, but the real world remains visible behind and around the virtual; the disconnect is lesser. The dynamic range (brightness) of the real world also greatly exceeds that of conventional displays. As a result, presenting both worlds with consistent appearance on conventional displays can be impossible in common circumstances. Lastly, and most importantly, with the real world directly visible, even a small amount of time lag (latency) between the user moving and the virtual imagery updating can be visible and highly distracting. Our brains interpret eye-perceived visual displacements differently depending on information from the other senses; when the senses do not agree, we misinterpret the motion (Wallach, 1987). For instance, the appearance of a real object can appear to change in the same way when it is moving and rotating or when the person is moving and rotating, and our vestibular-ocular system distinguishes it based on the other senses. In VR or in AR where the augmentation dominates the user's view, the user may misperceive the environment as moving when, in fact, only the user's head is moving and the system was slow to update. A fully effective (and non-distracting) OST-AR display should resolve these issues, among others.

1.2 Motivation

The main purpose of AR is to mix the virtual and real worlds together; it should provide an effective and persistent illusion that the virtual and real worlds coexist in the same environment. On the path to creating

Table 1.1: Example forms of error in AR systems.

Static	*Dynamic*	*Latency*
• Aliasing • Calibration • Display Non-Linearity • Modeling (Rigid) • Optical Distortion • Photometric • Pose Tracking	• Modeling (Shape Change) • Motion Prediction • Photometric • Pose Tracking	• Computation • Display • Pose Tracking • Rendering

this illusion, however, many sources of error exist (Holloway, 1995, 1997; Livingston and Ai, 2008); many of these are listed in Table 1.1. A complete AR system should consider all possible forms of error.

The work presented in this thesis focuses on subsets of two of the major issues that plague AR displays: latency and the dynamic range of the real world (photometric). Solving these sources of error would greatly improve users' experiences with these types of displays, so long as the solution for one does not nullify the solution for others.

1.2.1 Latency

Maintaining the illusion of AR requires that the two worlds be registered to each other, spatially and temporally. Of the many types of registration errors existing in AR systems, system latency is the largest, outweighing all others combined (Holloway, 1997; Brooks, 1999). Latency induced errors take many forms, though mostly as "swimming" and "judder." Both types of errors are highly distracting to many users, and can induce simulator sickness (Pausch et al., 1992) in VR situations.

Errors due to latency often results in virtual imagery lagging ("swimming") behind the intended position while the user's perspective is moving. Swimming errors result from the end-to-end latency (the difference in time between when the user moves and the change in imagery in response) being too large. In early graphics systems, this type of latency was typically expressed as the depth of the rendering pipeline and the display's frame rate, where each frame buffer in the path added another frame of latency. Previous Graphics

Processing Units (GPUs) used deep pipelines to improve throughput, but with the consumer advent of VR, these pipelines are shortening. AR (especially for OST displays) has even tighter requirements.

Latency errors can also result in the virtual imagery appearing to smear and strobe ("judder"). Judder errors result from the discretization of time, typically by the frame rate of the display, but also by the sample rate of slow tracking systems. Both the HTC Vive and the Oculus Rift CV1 use 90 Hz low-persistence displays for VR, which helps; although, for OST-AR, updates at these rates can cause quite visible displacement (*see* Section 3.1). However, some systems or complicated graphics applications are unable to keep to a 90 Hz schedule and require additional techniques to compensate for the slower render rate, which (without this compensation) would have further increased the latency.

Past approaches to reducing latency have included applying a Post-Rendering Warp (PRW)—*i.e.*, last minute improvements—to the imagery using the latest tracking information (Mark et al., 1997; Smit et al., 2010). In addition, the notion of updating displays in raster order (top-to-bottom, left-to-right) is fading as new display technologies lend themselves to very high frame rate, frameless, or random-order update schemes (Bishop et al., 1994; Dayal et al., 2005). Some of these techniques were originally developed when graphics systems could not keep up with standard 60 Hz displays; now they can be extended to support kHz-order displays, which suffer from the same relative problem.

1.2.2 Dynamic Range

The real world is a High-Dynamic Range (HDR) environment (*see* Figure 1.2 for an example). Thus, HDR displays for AR are necessary for the user to comfortably and simultaneously perceive both the real and virtual worlds in close visual proximity. This closeness may be physical (*i.e.*, a tree's shadow on a bright sunny day) or temporal (*i.e.*, entering or exiting a windowless building). While the human visual system can see both extremes simultaneously, conventional cameras cannot (in a single exposure) capture both. Furthermore, conventional displays (*i.e.*, an 8 bit/color display) lack the necessary range.

VST-AR displays can sidestep the problem by auto-adjusting (optimizing) the exposure as a balance. In this way, the conventional camera captures a conventional bit depth image for output on a conventional display; the virtual overlay need only match the same limited range. Given an HDR camera, renderer, and display, one could extend this VST solution to the dynamic range of the real world. However, for a low latency, OST-AR display, the situation is more complicated: the sensors and display need to be low latency in addition to supporting HDR.

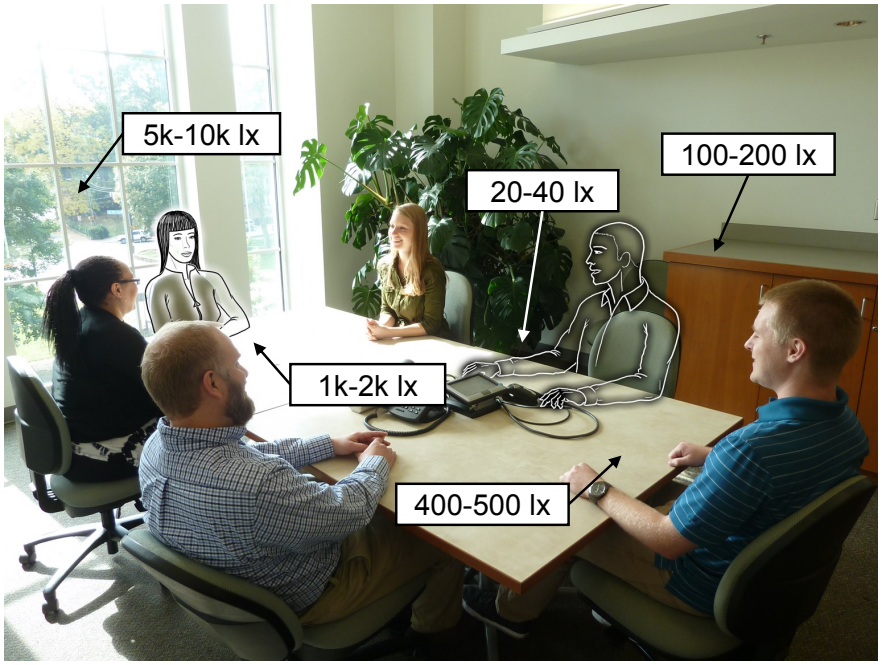


Figure 1.2: A sample scene of people in a room with HDR brightness, captured by a standard camera, annotated with relative brightness measurements from physical points around the scene, and augmented with two simulated people.

1.3 Thesis Statement

In Augmented Reality displays, the combination of performing Post-Rendering Warps in display hardware to mitigate latency registration error and using scene-adaptive, High-Dynamic Range color to improve the consistency of virtual and real imagery decreases the disagreement of the two worlds.

1.4 Contributions

The main contribution of this dissertation is presenting the implementation and design principles of a Head-Mounted Display (HMD) for OST-AR supporting HDR, RGB (Red, Green, and Blue) color. This design offers some characteristics not previously achieved by OST-AR HMDs: an average end-to-end latency of 80 µs for a 6 bit monochrome display and of 124 µs for a 16 bit/color HDR display. These displays use a new low latency rendering approach that breaks the standard display paradigm of sense→render→warp→transmit→display. Instead, the render procedure is split among devices by performing a high-frequency (about 16 kHz) PRW in the display hardware (*i.e.*, sense→render→transmit→sense→warp→display). Furthermore, we use new modulation approaches suitable for low latency. Applying this operation to high-frequency,

Figure 1.3: A user “wearing” our AR display. The monocular virtual imagery is presented to the user’s left eye.

binary-based displays requires both connecting the tracking system and coupling the latency compensation process directly to the display modulation mechanism. Contributions specific to each of these approaches are listed in Sections 3.1.1 and 4.1.1. An example of the author using an early iteration of the implemented display is presented in Figure 1.3.

1.5 Structure

Chapter 2 provides an introduction to relevant components of AR and low latency displays and an overview of prior approaches. Chapter 3 describes an approach to performing low latency rendering. Chapter 4 presents approaches to modulating binary Spatial Light Modulators (SLMs) for low latency and HDR. Lastly, Chapter 5 provides a summary and conclusion.

CHAPTER 2

BACKGROUND

2.1 Introduction

This chapter provides a background into the realm of AR and concepts necessary to understand the work presented in the later chapters. Section 2.2 reviews related work and existing systems in AR and low latency display techniques. Our system uses a Digital Micromirror Display (DMD) (Hornbeck, 1997), which is a type of SLM, at the core of the display; Section 2.3 provides an explanation on how DMDs work. Lastly, Section 2.4 provides background on simple modulation techniques, which can be applied to binary displays, like DMDs.

2.1.1 Units

In the discussion that follows in this document involving multiples of units, SI prefix notation is used: k = 1000, M = 1 000 000, *etc.* For consistency, SI prefix notation is also used for computer memory sizes of bits (b or bit) and bytes (B): 1 kbit = 1000 bit, 1 Mbit = 1 000 000 bit, *etc.*

2.2 Related Work

There has been a lot of past and recent work on AR displays and reducing latency for AR, VR, and conventional displays. This section provides background regarding all of the above.

2.2.1 Existing Near-Eye OST-AR Displays

In recent years, there have been several commercial releases or announcements for near-eye AR displays or systems. These systems are intended to be head-worn, with a display located near the user's eye(s), mostly for everyday use; however, many vary on use case. This proximity complicates matching the focus between the

display, which is only a couple centimeters from each eye, and the world. However, the primary complication when designing an OST-AR display is providing a human-matched Field of View (FOV) (Maimone, 2015). Nearly all commercial displays provide an FOV (about 40° along the diagonal) much smaller than even eye glasses ($\geq 100°$).

Some, like the OST Google Glass[1] or the VST Vuzix M100 or M300 Smart Glasses[2], are marketed as stand-alone devices. They combine an Android™-based smartphone-level System on a Chip (SoC) device with a small display, worn on glasses (or frames). The display, however, is not centered in the user's vision but is instead offset to the side of one eye's FOV, leaving a mostly unobstructed view of the real world. These types of systems may not truly be AR, but instead provide a user a smartphone computer experience in constant, hands-free view. They may provide context-sensitive information or a Heads-up Display (HUD), but they are limited in ability for providing an MR experience.

Other systems go for a more mainline AR experience where the virtual imagery is in the center of the view. The Meta 2 Augmented Reality Headset[3] provides a wide FOV (90° diagonal), OST-AR experience. However, it is a tethered experience; it is connected to a traditional PC. Despite this, backpack PCs—currently being offered to support room-scale VR experiences—could be applied to it as well. On the other hand, systems like HoloLens[4] or Lumus' development kits[5] provide stand-alone OST-AR experiences. The former provides an integrated mini-PC running a special version of Windows®, while the latter runs Android™. Integrated tracking on all of these systems allows the user to turn their head or move around the room while keeping a virtual appearance integrated into the same space, with varying levels of quality.

While these existing OST-AR displays may differ in implementation, part of the display experience, however, is common to all: namely the frame rate and render loop. None of these displays exceed a full-color frame rate of 120 Hz, though some are color sequential at higher speeds (*e.g.*, the HoloLens operates at 60 Hz overall, but each color appears for a fraction of the 1/60 s frame time). While the render loops could support deferred rendering or GPU preemption (*see* Section 2.2.4.1) for a last-minute PRW, they remain subject to displaying an image (or a color plane of an image) for a full frame time (or sub-frame time). This would be a

[1] `https://www.google.com/glass/start/`

[2] `https://www.vuzix.com/Products`

[3] `https://www.metavision.com/`

[4] `https://www.microsoft.com/microsoft-hololens/`

[5] `http://www.lumus-optical.com/`

best-case latency of 1/120 s for a 120 Hz simultaneous color display (or 1/180 s for a 3-color 60 Hz sequential display), which can produce a noticeable displacement between real-world and virtual object correspondences (*i.e.*, a registration error).

2.2.2 Registration Error

A major failure point for AR systems is registration error: if real and virtual objects are supposed to coexist in the same space, but they do not appear in the correct place, then the illusion is broken. Holloway (1995, 1997) and Livingston and Ai (2008) performed error analyses for AR systems. Sources of error takes many forms: static (*e.g.*, optical distortion, calibration, display non-linearity, aliasing, position tracking, and orientation tracking), dynamic (*e.g.*, position and orientation tracking), and latency. Of all of these, however, the greatest source is latency, outweighing all others combined, even at moderate head movement speeds of 500 mm/s and 50 °/s (Holloway, 1997).

Most of the static errors (*i.e.*, errors occurring while the user or scene is not moving) can be resolved by improving system calibration (*i.e.*, properly measuring the relationships among objects, the display, the user, and the scene). Certain tracking errors (both static and dynamic), however, require the use of a tracking system of sufficient quality for the task, which may or may not yet exist, depending on the task's scope. Latency induced errors, which usually manifest dynamically, require every component in the system to iterate quickly and report quickly. Care must be taken when developing a system to minimize errors to improve the user's experience, though latency has often been the hardest to solve. Failure to compensate sufficiently for the overall system latency can lead to simulator sickness in users (Buker et al., 2012).

2.2.3 Latency Requirements

When designing an AR HMD, or any other head-worn equipment, one must consider head motion speeds and accelerations. The rates at which a user moves or turns his or her head is related to the amount of registration error that a fixed duration of latency produces. For VR, Yao et al. (2014) suggested targeting 20 ms of motion-to-photon latency on their Rift display. Bailey et al. (2004) evaluated head-worn HUD latency requirements for OST-AR piloting operations, which requires a display error of less than 5 mrad (less than 0.286°) (SAE, 2001). In this case, to meet a head turn speed of 10 °/s would require a latency of less than 25 ms; 100 °/s would require less than 2.5 ms of motion-to-display lag. List (1983) measured users head rotation speeds and found them to exceed those speeds. When looking for a target in an unknown location,

users turned between 90 °/s and 240 °/s; however, when they turned to known locations, users rotated at upwards of 437 °/s. This would seem to require $0.286°/(437\,°/\mathrm{s}) = 0.654\,\mathrm{ms}$ of latency to meet the 5 mrad standard. Note, however, that for our approximately 30° FOV (horizontal) prototype display, a 5 mrad error manifests as an error of almost 10 pixels, which is quite noticeable. Thus, the non-noticeable tolerances may be tighter.

When considering more moderate head turns, the latency time tolerances may increase. Jerald and Whitton (2009) and Jerald (2009) performed human latency tolerance measurements with a simulated zero-latency HMD and artificially induced scene-motion. They theorized and demonstrated that latency thresholds are inversely proportional to peak head-yaw acceleration. When measuring oscillating users, they found that for their use cases, the users' ability to detect any latency ranged from 3.2 ms to 60.5 ms (50 % thresholds ranged from −3.2 ms to 103.2 ms), though their user's head motion profiles varied and the users had no real-objects to measure latency against. As a result, additional study may be needed to find real-world tolerances in OST-AR.

2.2.4 Latency Mitigation Strategies

There have been many past approaches to reduce the effects of system latency in display systems. Perhaps counter-intuitively, many of them involve *delaying* certain operations until the last possible moment. As long as the user's motion is detected fast enough with high speed tracking systems (*e.g.*, Oculus VR, 2013) or predicted well enough (*e.g.*, Azuma, 1995; Didier et al., 2005) and displayed just-in-time, then there is an opportunity to keep overall end-to-end latency sufficiently low. Several of the techniques discussed in this section were originally developed when graphics hardware was incapable of reaching the frame rate of displays for the desired scenes (a case which remains valid today for complex scenes in video games). When the rendering system runs slower than the display, this further increases the overall end-to-end latency as old display frames are repeated.

2.2.4.1 GPU Improvements

Recent advents in GPUs have targeted improving latency in desktop monitors and in consumer VR displays. G-SYNC™[6] by NVIDIA® and FreeSync™[7] by AMD are methods for dynamically modifying the refresh rates of monitors. Typically, displays run at a fixed frame rate (*e.g.*, 60 Hz), and GPUs run as fast as possible; this difference can cause tearing of the displayed imagery. Enabling VSYNC (Vertical Sync) prevents the tearing—by locking the GPU's frame rate to match the monitor's and by delaying processing of subsequent frames—but it can cause latency and stutter when the amount of rendering computation varies. G-SYNC™ and FreeSync™ each allow the GPU to transmit a variable frame rate stream (up to the maximum supported by the monitor), where each frame is transmitted as soon as it is produced. This eliminates the appearance of the stutter and reduces the latency.

Improvements specifically targeting VR displays tend towards supporting exclusive-mode rendering and GPU preemption. The former excludes the Operating System's (OS's) compositing engine from managing VR displays, which could have incurred frame delays, and the latter allows for just-in-time, last-moment operations to occur prior to frame transmission with specific timing guarantees.

2.2.4.2 Frameless Rendering

Typical rendering consists of a series of frames, like a movie film reel, where each frame represents a single instant in (or short duration of) time. Each pixel in a frame corresponds to the same time. Frameless rendering, on the other hand, drops that concept, though there are multiple ways in which the concept of frames is ignored.

Bishop et al. (1994) originally proposed a scheme for frameless rendering to avoid two problems: the latency incurred by double-buffering, and the image tearing incurred by not double-buffering. Instead of producing a stream of raster-order frames, they produce a stream of random-order pixels. In this way, they produced a "crude approximation of motion blur." While they only simulated the method, this type of system could be implemented on a display supporting random access updates or on a raster-order display where the display process greatly exceeds that of the render process. The idea is that multiple *display* frames are used to generate a coherent *perceived* (or *integrated*) frame. Dayal et al. (2005) extended the idea by

[6] `http://www.geforce.com/hardware/technology/g-sync/technology`

[7] `http://www.amd.com/en-us/innovations/software-technologies/technologies-gaming/freesync`

adaptively choosing which pixels to update based on knowledge of the scene, increasing visual quality with little increase to the rendering overhead.

An alternative this is more in line with raster-order displays is that of "just-in-time pixels" (Mine and Bishop, 1995). Born in the days of Cathode Ray Tubes (CRTs), the idea is also known as "racing the beam," referring to the electron beam in CRTs. This method produces a video stream that has no concept of a constant piece of time across the frame. Each pixel produced can have a different time than its neighbors; it takes advantage of low-persistence, raster-order displays by generating the appropriate color for each pixel based on its location at the time it is emitted onto the GPU-display connection. With a zero-latency implementation, a yawing user, turning at any speed, would see each pixel appears aligned, while an external observer (using a cloned monitor) would see the rendered world as horizontally skewed. Recent implementations of this method include the "Ultra Low Latency Frameless Renderer" ray-casting VR system by Friston et al. (2016), which used a Field-Programmable Gate Array (FPGA) renderer to produce a beam-racing system with a latency of about 1 ms from tracker to pixel emitted on an Oculus DK2[8].

2.2.4.3 Post-Rendering Warp

Post-Rendering Warp (PRW) is a catchall phrase for methods where a normal rendering algorithm is followed by one or more reprocessing steps that transform ("warp") the conventional output to produce something new. A typical latency-reduction use for PRWs follows an expensive 3D rendering operation occurring that uses old tracking information with some 2D or 3D transform to shift the perspective to match the latest user pose (So, 1997; Pasman et al., 1999; Itoh et al., 2016). Performing this action both supports low frame rate rendering systems (by providing intermediate frames between full renders) and general rendering systems (by realigning the imagery as late as possible). Implementations of PRWs take many forms.

Image-based Rendering (IBR) is a more general PRW technique where the final operation is an image composition algorithm. Regan and Pose (1994) presented the "Address Recalculation Pipeline" as an IBR method. While traditional rendering directly and wholly renders the entire scene for a given time instant, IBR divides up the 3D rendering process object-by-object into multiple 2D+Depth images. Then, at the last possible moment, it uses the latest tracking information to perform a layered composition of the relevant

[8] `https://www.oculus.com/dk2/`

objects' images. Some IBR implementations combine these schemes with more conventional rendering pipelines (Kajiya and Torborg, 1996; Walter et al., 1999; Dayal et al., 2005).

Other PRW methods perform more complicated operations. Given a set of reference 2D+Depth frames and the corresponding perspectives, one could generate a novel perspective by selectively compositing those views (McMillan and Bishop, 1995; McMillan, 1997). Mark (1999) took advantage the coherence of motion to produce fast interpolated frames between expensive full renders, and Didier et al. (2005) used a pose predictor to help guide the warp. Others have used multiple PCs or specialized hardware between the PC and display to produce latency mitigated imagery for the display (Smit et al., 2008, 2010). Oculus has integrated PRW into their VR headset's drivers to help game developers lower the experienced latency when their rendering is too expensive or unstable in frame rate, calling it "Asynchronous Timewarp" (Antonov, 2015).

Complications with warping arise when the perspectives contain insufficient information and the new perspective looks into the missing regions. 3D warps of these perspectives tends to create holes in the output imagery as objects become disoccluded. For example, in Figure 1.2, if the view shifted slightly to the left, then the right elbow of the leftmost person should be displayed as it emerges behind her neighbor's head. Thus, as parts of the scene become "visible" in the new perspective, the renderer needs a hole filling operation (*e.g.*, Mark et al., 1997) to hide the gaps. These methods are never perfect in scenes of overlapping objects, as there will always exist a new perspective where the necessary data is unavailable from the reference perspective(s). Nonetheless, the goal is that any new errors introduced by the algorithm are less disturbing than the latency of the original image or an empty gap in the warped image.

2.2.5 High-Dynamic Range

As previously noted, the real world varies greatly in brightness. The human visual system operates well with color vision ("photopic vision") between about 0.003 cd/m^2 and 35 000 cd/m^2, but can still see luminances less than 0.001 cd/m^2 (Daly et al., 2013). Conventional displays, however, are much more limited (*e.g.*, 0.1 cd/m^2 to 300 cd/m^2). Some displays (*e.g.*, Wetzstein et al., 2010; Mann et al., 2012; Microsoft, 2016) employ darkening materials behind their dim and/or limited range display to reduce the brightness of the real world behind it.

An early implementation of an HDR display used a patterned backlight behind a normal LCD (Liquid Crystal Display) screen, providing a luminance range of 0.1 cd/m^2 to 10 000 cd/m^2 (Seetzen et al., 2003). This display used controllable, high-brightness Light-Emitting Diodes (LEDs) in an array behind the LCD

screen to provide regional brightness control though with localized high resolution detail. By compensating for the low resolution of the backlight in the LCD, they were able to create a display that was not noticeably perceptually different from a purely high resolution HDR display. Pavlovych and Stuerzlinger (2005) performed a similar operation by backlighting an LCD with a digital projector.

Some current commercial displays also support HDR. Early forms of these displays use a similar backlight technique as Seetzen et al. (2003); these LCD displays were not directly intended for displaying HDR content, but instead to provide darker blacks (zero values). Since then, several competing standards for HDR-labeled televisions have emerged: HDR-10 (UHD Alliance, 2016), which supports 10 bit/color; Dolby Vision (Dolby, 2016), which supports 12 bit/color; Hybrid Log-Gamma (Borer and Cotton, 2015), which supports 10 bit/color or 12 bit/color; and Advanced HDR (Technicolor, 2016), which supports 10 bit/color. A new standards war may be brewing, though at least this time the specifications are not incompatible; a display can support multiple standards.

2.3 DMD Operations

One of the most accessible display types for low latency approaches is the DMD (Digital Micromirror Display), manufactured by Texas Instruments (TI) under the brand "Digital Light Projection" (DLP®). Low level control of this fast display type has been demonstrated by numerous groups, including Raskar et al. (1998); McDowall and Bolas (2005); Jones et al. (2009); and Bhakta et al. (2014). Since the presented system makes extensive use of a DMD, it is useful to present how DMDs work. Our system uses the TI DLP® Discovery™ 4100[9] (Texas Instruments, 2013b), which uses a DLP7000 DMD chip (Texas Instruments, 2013a), which is capable of XGA (Extended Graphics Array) resolution (1024×768). The discussion below uses the specification and parameters of this display in its explanation.

2.3.1 DMD Chip Basics

A DMD chip is essentially a binary-data, double-buffered, Random Access Memory (RAM) device, where one of the buffers represents a 2D array (DLP7000 active diagonal length: 0.7 in) of tiny mirrors (pitch: 13.68 µm). The first buffer (the "back buffer," in rendering terminology) is user-writable, where each memory entry in the 2D array is one bit. The other buffer's (the "front buffer") bits control the pitch of the mirrors,

[9] `http://www.ti.com/tool/dlpd4x00kit`

where "0" and "1" correspond to pitches of $\pm 12°$. The mirrors are binary in nature; a programmer cannot specify any stable in-between angles.

Access to the back buffer is exposed by row addressing; the controlling processor can load any row with a complete row's worth of data, and then jump to any other row and load it. The DLP7000 supports pixel clocking speeds between 200 MHz and 400 MHz, and each row requires 16 clock cycles to load completely, supplying 64 binary pixels per clock cycle. Rows are grouped into blocks, each of 48 rows, though the writing process can jump among blocks freely.

To copy data from the back buffer to the front buffer's mirrors, the controlling processor must issue a Mirror Clocking Pulse (MCP). On the DLP7000, the processor can issue an MCP at any time between row loads, as long as at least 4.5 µs have passed since the previously issued MCP. When issuing an MCP, the processor can specify that one, two, four, or all of the blocks should be copied from back to front buffers. However, once an MCP has started, the processor cannot make any changes (back or front) to the affected blocks for 12.5 µs, during which the mirrors in those blocks are stabilizing.

The restrictions on loading blocks undergoing MCPs limits the random-access utility of DMDs. At 400 MHz, loading one block requires $(16 \times 48)/(400\,\mathrm{MHz}) = 1.92\,\mu\mathrm{s}$, which is less than the minimum gap between MCPs: 4.5 µs. As a result, using raster scan order (left-to-right, top-to-bottom) and issuing two- or four-block MCPs is much more efficient, resulting in the maximal binary frame rate (32 kHz at 400 MHz).

2.3.2 Conventional DMD Projector Basics

Standard DMD projectors typically spatially illuminate the entire active surface of the DMD uniformly. Controlling the pitch of each mirror's "0" or "1" values corresponds to deflecting the illumination source either out of the display through the projection lens ("1") or into a light absorbing baffle ("0"). Producing non-binary grayscale values requires controlling the relative proportion of emitted light for each pixel over time. Generating such analog (grayscale) values from binary values is on form of modulation that is central to later discussion (*see* Section 2.4).

DMD projectors typically provide color either by optically combining multiple DMD chips—each with different colored illuminators—or by using color-sequential methods on one chip. In the past, color sequential methods used a spinning color wheel, cycling among red, green, and blue; newer projectors may cycle among colored LEDs, but the basic process is the same. While a single color is active, the DMD undergoes a modulation sequence for that color. For a standard 60 Hz projector, the three colors may alternate at

180 Hz. Some displays may also incorporate white light illumination into the sequence to increase maximum brightness.

DMD projectors control their mirrors' duty cycles based on the received video input. Often this input is supplied by a computer's GPU using a DVI (Digital Video Interface), HDMI (High-Definition Multimedia Interface), VGA (Video Graphics Array), or DP (DisplayPort™) connection, often at 60 Hz. All of these connection types supply video streams in a raster scan format at a data rate commensurate with the video frame rate (*i.e.*, each 60 Hz frame requires 1/60 s to transmit). Since the modulation and color sequential schemes of the projectors often require complete frames to start execution, the controlling processor must buffer a complete frame internally before starting to load the DMD's back buffer, resulting in a display latency of at least one video frame. This delay is unacceptable for low latency OST-AR.

2.4 Modulation

The use of a DMD requires conversion of grayscale pixel values to a sequence of binary frames that approximate those values. This is a form of analog-digital-analog conversion: each "analog" pixel value is represented as an n bit grayscale value, which is converted to a series of binary "1" and "0" values that represent white and black projected values, which in turn, due to persistence in the human visual system, the user sees as an analog grayscale value with 2^n possible levels. Generating analog values from binary values is called modulation. Modulation is extendable to other high speed displays or transmissions that support binary input and need variable output (Jang et al., 2009; Greer et al., 2016).

To display a given 6 bit intensity d over a period of 63 bit-times, the light is turned "on" for d bit-times and "off" for the remaining times. The "on" and "off" pulses are integrated (in this case, by the human eye), and result in the appearance of the desired intensity. The differences between the schemes presented below are in the selection and order of which d bit-times should be "on". The selected algorithm affects the perceived quality of the resulting image (*e.g.*, flicker) and determines the requisite hardware resources (memory storage and bandwidth and computation time). As long as the time to execute the sequence of binary frames is short enough, flicker should be minimal. On a DMD, where the illuminator is typically constantly on, the equivalent operation for turning the light "on" and "off" is instead flipping the mirror to the "on" and "off" angles.

To generalize, suppose a modulation $m(d, s)$ exists, where d is the desired intensity and s is the step index. Most Pulse Train Modulation (PTM) approaches on DMDs require an exponential number of binary frames to execute one full integration cycle: $O(2^n)$, where n is the number of bits in the supported desired value. If the value b (constant for conventional DMD projectors) represents the brightness of the illuminator, then the value g represents the generated intensity of an exponential-time modulation sequence in Equation (2.1) below. Each iteration (step variable: s) of the summation represents one step of the integration cycle, and the full summation represents the whole integration.

$$g = \sum_{s=0}^{2^n-1} b \times m(d, s) \tag{2.1}$$

One could extend values d and g into functions of pixel coordinates, and then this summation sequence could be applied to DMDs. Furthermore, the value of d could change with time, irrespective of the current step within the summation. For notational simplicity, those positional and time variables are suppressed in all examples that follow.

There are two well-known PTM approaches for converting analog signals to bitstreams (*i.e.*, a sequence of binary values): Pulse Width Modulation (PWM) and Pulse Density Modulation (PDM). In both approaches, the pulse generator operates at a high enough frequency so that the moving average closely approximates the analog value, when averaged over a window of time smaller than the human eye's response time. Some examples of execution sequences of these exponential-time PTM schemes are presented in Figure 2.1.

2.4.1 Pulse Width Modulation

In PWM, the width of the pulse of light generated is varied in direct proportion to the desired gray value at the pixel, as illustrated in Figure 2.1's left column. Thus, for a given window size, a 25 % gray value will generate a two-part pulse that is "on" for 25 % of the time and "off" for the remaining 75 % of the time window. When considering a discretized window, generating a 6 bit intensity requires 63 steps per interval. Thus, using the notation of Equation (2.1), the summation function for PWM can be as defined in Equation (2.2a), where m_{PWM} is defined in Equation (2.2b).

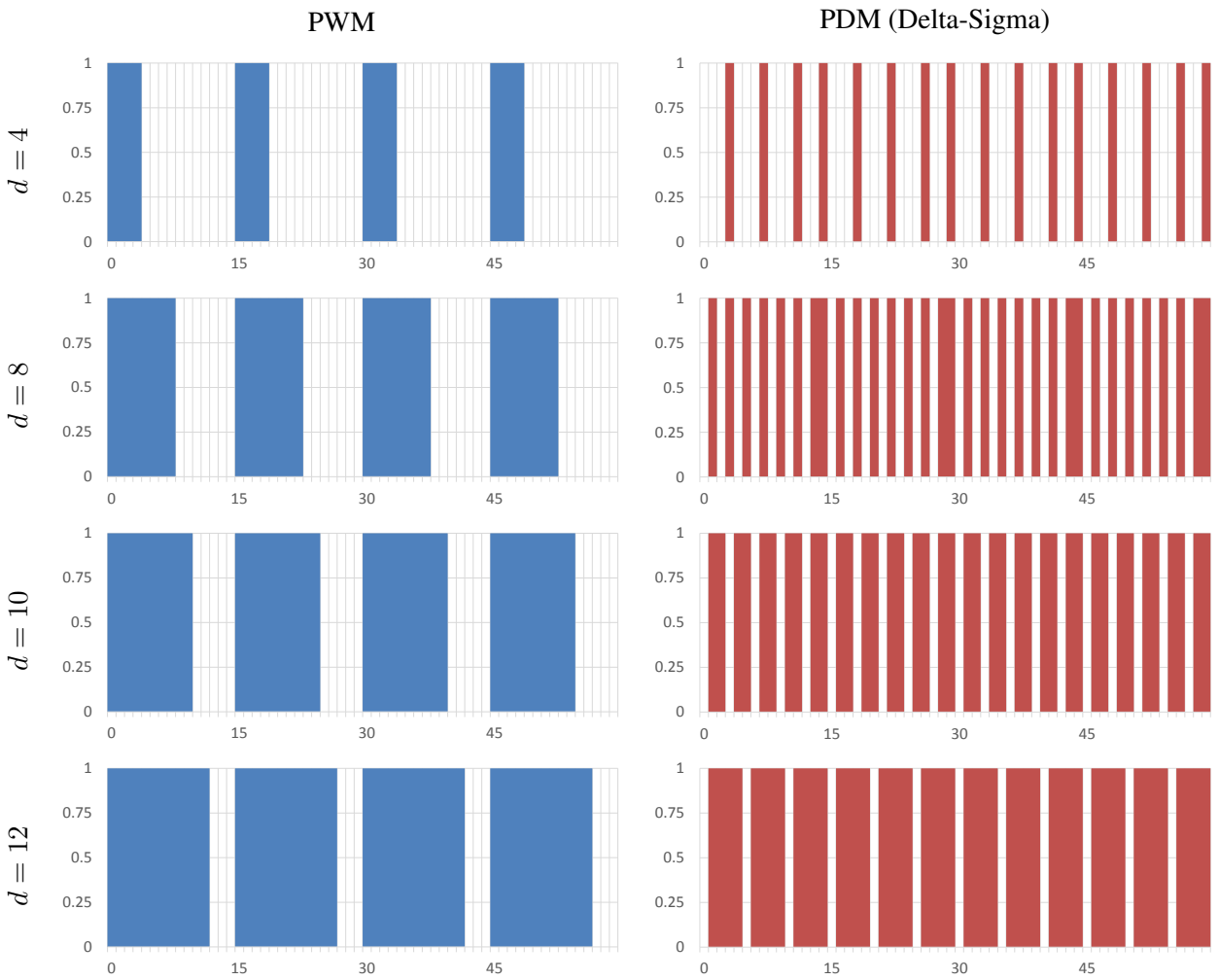


Figure 2.1: Sample comparison of simulated 4 bit gray level modulation schemes by method and desired intensity level. Each chart shows the output intensity (vertical axis) associated with 60 consecutive binary frames (horizontal axis). The value d indicates the desired integrated intensity in the range $[0, 15]$. Color indicates that the pixel is on with the illuminator using the specified intensity; white indicates that the pixel is off. The graphs show 4 complete integration cycles (each of 15 steps); the numbers on the horizontal axes indicate the first step of each integration cycle.

$$g_{\text{PWM}} = \sum_{s=0}^{2^n-2} b \times m_{\text{PWM}}(d, s) \tag{2.2a}$$

$$m_{\text{PWM}}(d, s) = \begin{cases} 1 & \text{if } s < d \\ 0 & \text{otherwise} \end{cases} \tag{2.2b}$$

2.4.2 Pulse Density Modulation

In PDM, each pulse is of a fixed width (one binary frame) but the density of "on" pulses is varied in direct proportion to the desired gray value. Thus, a 25 % gray value will generate one "on" pulse followed by three "off" pulses, and a 10 % gray value will generate one "on" pulse followed by nine "off" pulses. These pulses are issued in a repeating pattern, with each pulse a fraction of the integration cycle's duration. For many intensity values, this pattern can repeat during one integration interval. In effect, PDM is equivalent to a much higher frequency PWM for those values, which results in output that is much smoother in time.

The ideal method for generating PDM pulses is known as Delta-Sigma Modulation ($\Delta\Sigma$) (Aziz et al., 1996), illustrated in Figure 2.1's right column: an "on" pulse is generated if the accumulation of all the prior "on" pulses ("sigma") is an entire unit ("delta") less than the integration of the desired gray value over that time. While Delta-Sigma is a very powerful approach with many variants (*e.g.*, higher-order integrators, continuous- vs. discrete-time, *etc.*), for the specific case of fixed-rate binary values, it is equivalent to Bresenham's line drawing algorithm (Bresenham, 1965): an increment in x is accompanied by a change in y only if the deviation from the desired line would be more than one unit. However, Delta-Sigma implementations impose a significant memory cost: they require at least as much memory as storing one desired image, since an error term must be maintained for each pixel. Using the notation of Equation (2.1), the summation function for Delta-Sigma can be as defined in Equation (2.3a), where $m_{\Delta\Sigma}$ is defined in Equation (2.3b); $e_{\Delta\Sigma}$, the error term, is defined in Equations (2.3c) and (2.3d); and d_{max}, the maximum supported desired value, is $2^n - 1$.

$$g_{\Delta\Sigma} = \sum_{s=0}^{2^n-2} b \times m_{\Delta\Sigma}(d, s) \tag{2.3a}$$

$$m_{\Delta\Sigma}(d, s) = \begin{cases} 1 & \text{if } e_{\Delta\Sigma}(d, s) \geq d_{\text{max}} - 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.3b}$$

$$e_{\Delta\Sigma}(d, 0) = d \tag{2.3c}$$

$$e_{\Delta\Sigma}(d, s) = \begin{cases} e_{\Delta\Sigma}(d, s-1) + d & \text{if } e_{\Delta\Sigma}(d, s-1) < d_{\text{max}} - 1 \\ e_{\Delta\Sigma}(d, s-1) + d - d_{\text{max}} & \text{otherwise} \end{cases} \tag{2.3d}$$

Note that subsequent summations in this notation should instead define $e_{\Delta\Sigma}(d, 0)$ as the last value of $e_{\Delta\Sigma}$ in the previous summation sequence; $e_{\Delta\Sigma}(d, 0)$ is assigned d for the first execution only.

CHAPTER 3

LOW LATENCY RENDERING

3.1 Introduction

Rendering complex scenes remains a time-intensive task, and the high-throughput render pipelines on modern computers do not always lend themselves to keeping latency low. Recent advances in GPUs, with optimizations for VR, have provided reductions in end-to-end latency (*see* Section 2.2.4). Unfortunately, performing all of the latency compensation in the PC, prior to transmission to the display, is insufficient for OST-AR. It may be sufficient for VR because VR has less stringent requirements for latency than OST-AR.

Consider the following example using an OST-AR HMD. Suppose the user's eye sees a 1080p display with a 60° horizontal FOV using a 60 Hz refresh rate. Suppose also that the user has an accurate zero-latency tracker, and that there is no system latency for processing that tracking data into the GPU's output. Thus, in this example, the only sources of system latency are for transmission and display. Conventional video transmission methods and displays typically require one frame time (the inverse of the frequency) to handle both steps simultaneously, though some displays, like DMDs, require one frame time for transmission and one more for display. If the user were panning his or her head at a constant 60 °/s, a relatively slow speed for humans, then, over the course of one frame, the world would appear to turn behind the image by 1°, which would be one-sixtieth of the display or 18 pixels (or double for a buffering display, like conventional DMDs). Despite the generous assumptions on the tracker and system, at least an order of magnitude increase in frame rate would be required to keep the frame duration latency error down to less than a pixel at these slow head turn speeds.

In order to perform low latency rendering in more relaxed conditions for AR, one needs a different means of rendering and displaying images. Some of the processing previously performed in the GPU should be delayed even later. This chapter presents such a method, in which part of the overall rendering process is delegated post-transmission. In this way, the overall rendering work is split between a conventional rendering

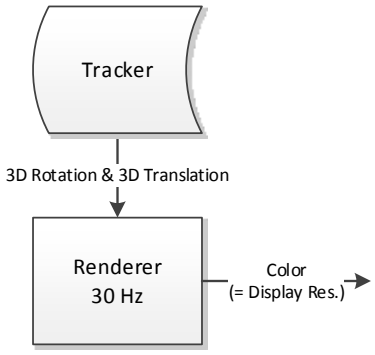


Figure 3.1: A sample, conventional render pipeline. Operation rates are given as examples for illustrative purposes only.

system (*i.e.*, PC and GPU) and the display controller, the latter of which performs some of the necessary processing at very high rates (on the order of 16 kHz). Section 3.2 presents a theoretical solution, while Section 3.3 presents an implementation of a subset of the theoretical solution.

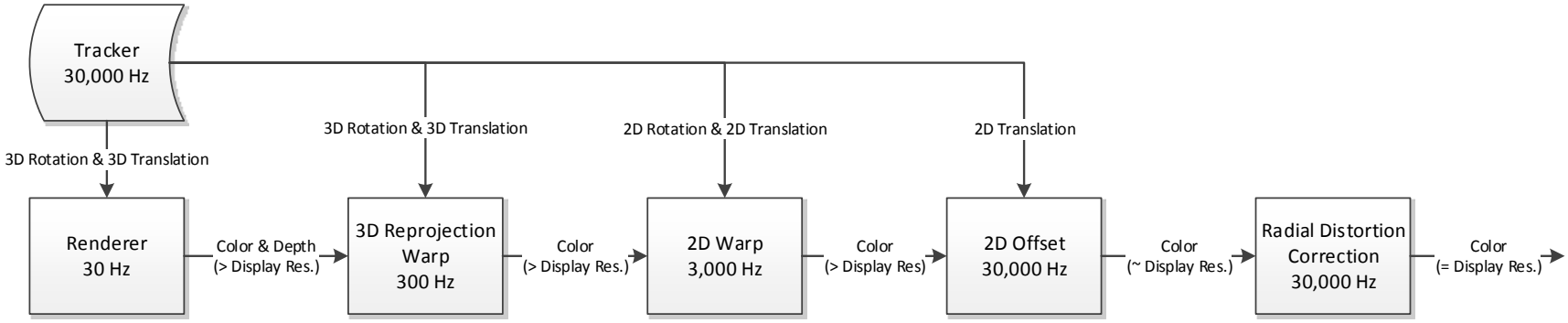
3.1.1 Contributions

- A real-time implementation of a simplified render cascade pipeline, namely the 2D-translation component, in FPGA hardware with an average end-to-end latency of about 80 µs to 124 µs, depending on color depth support.

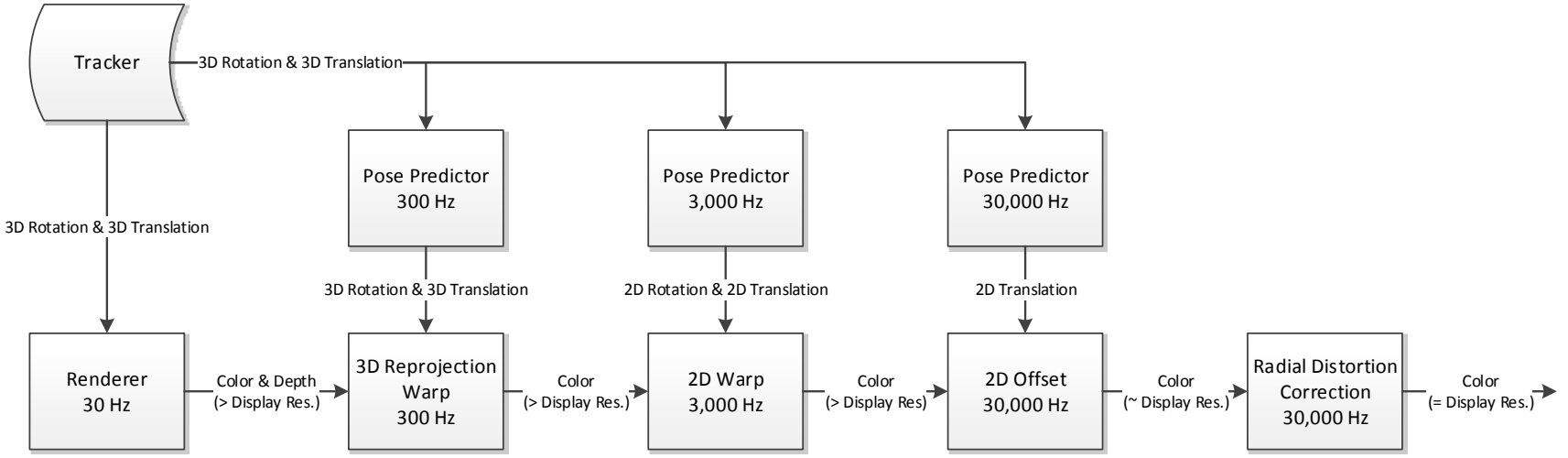
3.2 Render Cascade Pipeline

As part of a solution towards solving the AR latency problem, consider the following solution as an extension of PRW (*see* Section 2.2.4.3): a render cascade pipeline. While PRWs have typically occurred in the GPU at video rates, consider extending a normal render pipeline with distinct stages of warps, where some may occur outside of the GPU (*i.e.*, in the display controller hardware). All stages in this kind of system need not be limited by video rates or external video transmission mechanisms such as VGA, DVI, HDMI, or DP. Each new stage, between a typical 3D render and the final light-emitting display, performs a warping operation to mitigate the latency between the previous stage's output and the latest tracking information. Later stages could perform simpler actions than prior stages, thus enabling the later stages to execute more frequently.

Consider a typical rendering pipeline, as presented in Figure 3.1, which uses a tracking system to supply pose data to a rendering system. The output of that system feeds a standard display at video rates.



(a) A sample pipeline with a fast tracker.



(b) A sample pipeline with a slow tracker and fast prediction.

Figure 3.2: Two sample PRW pipelines with faster later stages than earlier stages. Operation rates are given as examples for illustrative purposes only.

Consider the notional example in Figure 3.2(a), which depends on a high-frequency, low-latency tracker to supply information to each stage before displaying on a high-frequency display. This example takes a slow 30 Hz renderer and processes the imagery to make it suitable for a 30 kHz display, which is in the typical range for DMDs. As some of these processes (at the declared rates) exceed the current capabilities of GPUs and the bandwidth of current video cable specifications, some of the later steps would need to occur after transmission from the GPU. In the absence of such a high-quality tracker, Figure 3.2(b) presents an alternative where the motion profile of the tracked object is sufficiently characterized to be predicable. Unfortunately, certain stages of motion (*e.g.*, the initial jerk at the start) are difficult to predict, and the results may be worse for it.

While these pipelines attempt to reduce the detectability of latency, in some ways, they also increase it. A normal 30 Hz renderer would typically be ready to output its imagery immediately after each frame, but the animation content of that frame (in this notional system) would be delayed by at least 3.7 ms ($1/300\,\mathrm{Hz} + 1/3000\,\mathrm{Hz} + 2 \times 1/30\,000\,\mathrm{Hz}$), and this assumes instantaneous transference of intermediate frames; a real system would require more time. As long as the delays in a real system would continue to be short enough to maintain each stage's assumption of small required changes, the final output would be improved (*w.r.t.* pose) when compared with the typical render pipeline, but certain content would be delayed. For instance, if the internal structure or color of a virtual object changed, that information would be delayed by the pipeline. It remains to be seen by user studies if this would be more distracting, but the swimming resulting from misaligned imagery is already known to be distracting, and the described render cascade pipeline could reduce or perceptibly eliminate this misalignment.

3.3 Prototype: Simplified Render Pipeline

In an effort to test a real-time version of the render cascade pipeline, discussed in the previous section, consider a simpler example for an OST-AR HMD. Suppose that one uses a fast 2-DOF tracker, which provides accurate and low latency pose data on the user's viewpoint's pitch and yaw angles. Let the PRW operations be reduced to two: a radial distortion correction operation and a 2D offset translation operation. Figure 3.3 provides a simple diagram of the dataflow for this reduced pipeline. This section presents the implementation and results of a prototype for this simplified render pipeline where we perform the standard render and radial distortion correction steps on a GPU and the 2D offset step in the display controller.

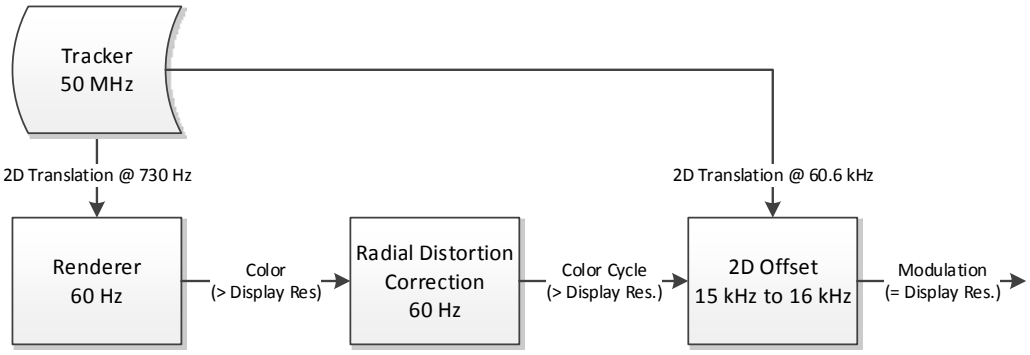


Figure 3.3: The PRW pipeline implemented in our prototype. Operation rates were measured using the constructed prototype.

3.3.1 Components and Implementation

Our prototype hardware setup for an OST-AR HMD using a DMD is presented in Figures 3.4 and 3.5. It employs a combination of traditional PC/GPU rendering with a post-rendering, post-transmission FPGA-based display update process. Virtual elements of the AR scene are rendered on the PC[1] and transmitted via DP to the display processing system for latency mitigation and emission. That system consists a Xilinx Virtex®-7 FPGA board[2] interfaced to a TI DLP® Discovery™ 4100 Kit[3], which is composed of a Xilinx Virtex®-5 FPGA board[4], an XGA DMD module, and projective optics. The DMD is capable of displaying binary frames to the user at up to 32 kHz though our implementation limits it to rates on the order of 15 kHz to 16 kHz. Figures 3.6 and 3.7 present the implemented data paths.

To track the user's motion, we use high resolution rotary shaft encoders[5], each with 40 000 ticks of resolution per revolution (0.0009 °/tick). Using shaft encoders like this limits the HMD to only angular motion. This "open-loop" tracking system, using quadrature encoding, is processed by a Digilent FPGA

[1]Our system used an NVIDIA® GeForce® GTX™ Titan Black (`http://www.nvidia.com/gtx-700-graphics-cards/gtx-titan-black/`) GPU for rendering.

[2]Our system used the HTG-777 (`http://www.hitechglobal.com/Boards/Virtex-7_FMC.htm`), which has an XC7VX690T-2FFG1761 Virtex®-7 chip (`https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html`).

[3]`http://www.ti.com/tool/dlpd4x00kit`

[4]This board has an XC5VLX50-1FF1153 Virtex®-5 chip (`https://www.xilinx.com/support/documentation/data_sheets/ds100.pdf`).

[5]Our system used US Digital E6 Series encoders (`http://www.usdigital.com/products/e6`).

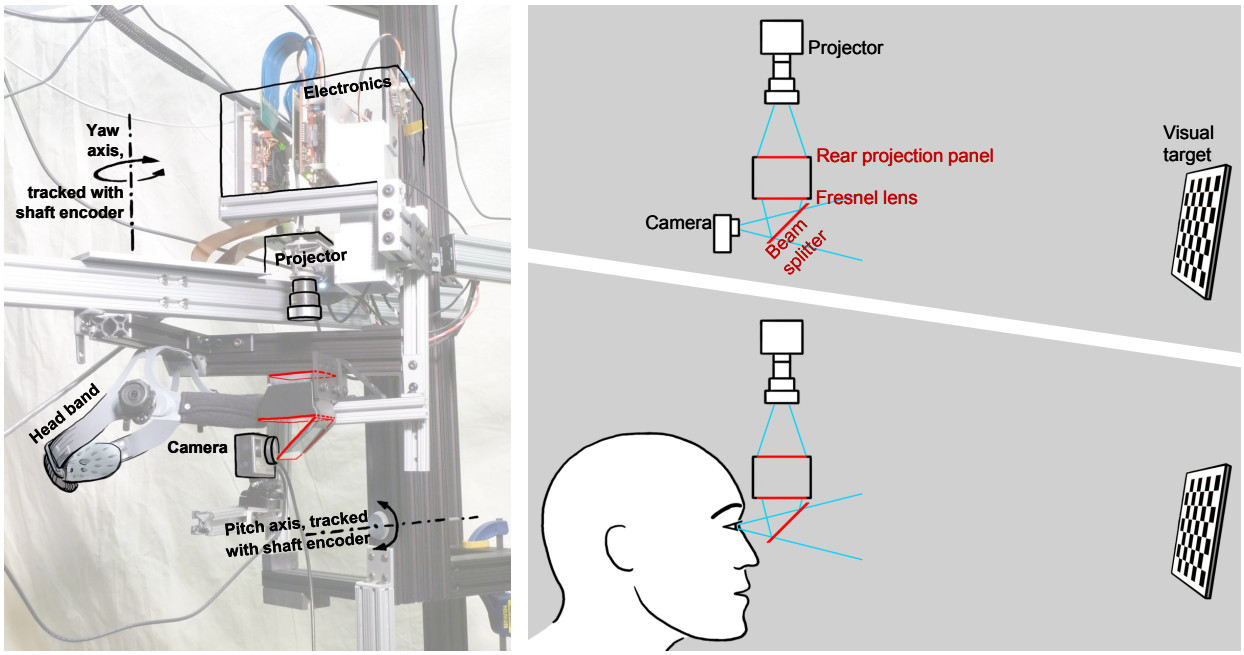


Figure 3.4: System assembly for the 6 bit/pixel system. Display electronics include a TI DLP® Discovery™ 4100 Kit, an HTG-777 FPGA board, and a custom DP input and interconnect board. Projector components include the XGA DMD chip and standard lens assembly. The remaining optics include the rear projection panel, Fresnel lens, and beam splitter (half-silvered mirror). Either a user or a camera can observe the optically combined physical and virtual visual targets.

board[6]. Tracking data is routed both to the render PC and the display processing system using separate high data rate RS-232 serial links. Each device receives tracking updates at a rate in excess of its display update rates.

The PC performs standard AR rendering from the user's (or test camera's) tracked and calibrated perspective, with one significant modification. Typically, one would render the scene to produce a 2D image directly suitable for display at the same resolution as the display from that perspective, performing any necessary radial distortion correction as well. In order to supply additional content for the PRW ("2D Offset" in Figure 3.3) renders a larger viewpoint; the display is capable of showing XGA resolution, but our PC renderer produces imagery in 1080p. The output from the GPU is such that the central 1024×768 region of the 1920×1080 output and that the central region is properly corrected for radial distortion. We used OpenCV's camera calibration library (Bradski, 2000) to perform the viewpoint calibration.

[6]Our system used the Spartan®-3 FPGA Starter Kit board (`https://www.xilinx.com/products/boards-and-kits/hw-spar3-sk-uni-g.html`), which has an XC3S200 Spartan®-3 FPGA (`https://www.xilinx.com/products/silicon-devices/fpga/spartan-3.html`).

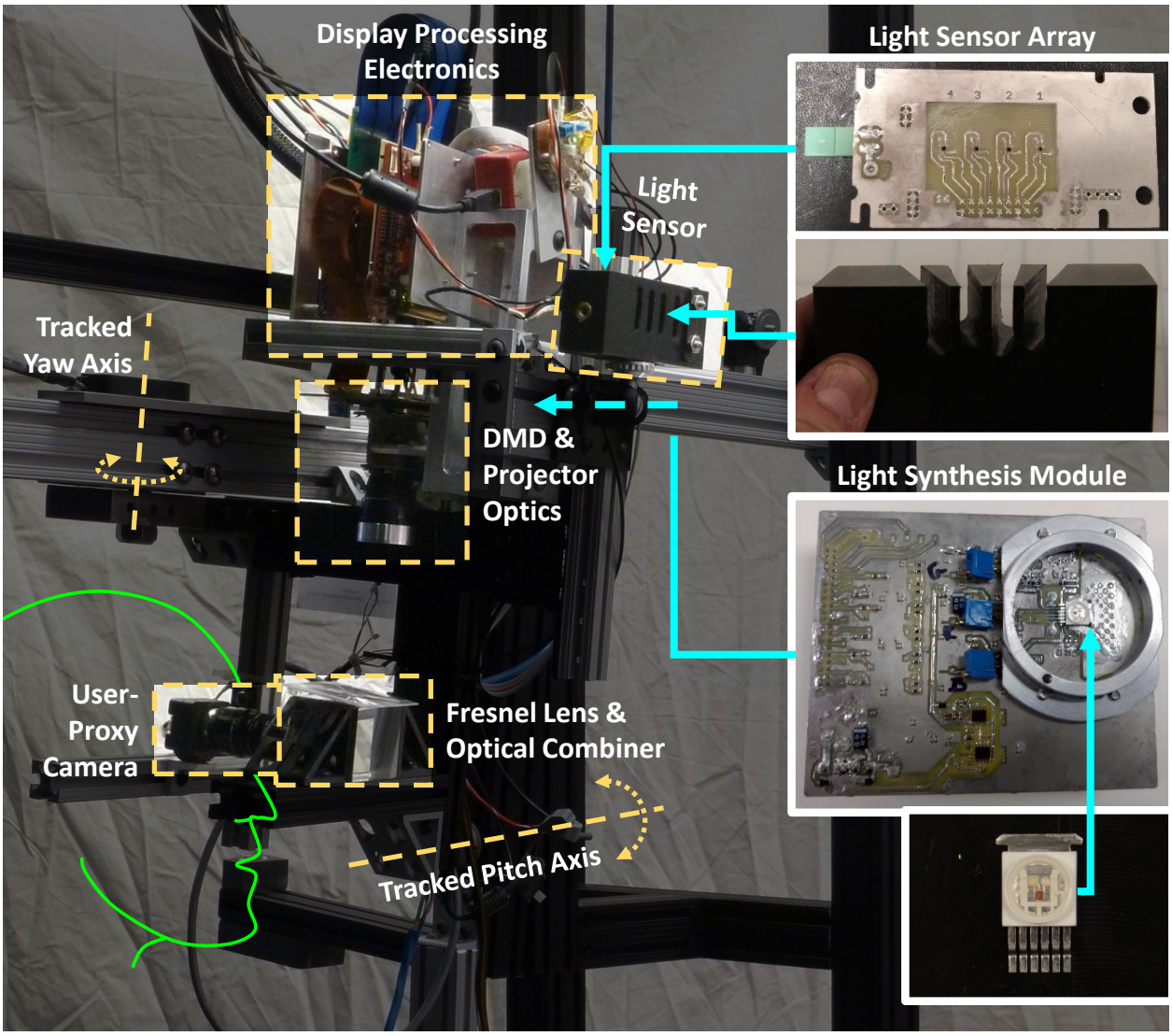


Figure 3.5: System Assembly for the 16 bit/color system. Display processing components include a TI DLP® Discovery™ 4100 Kit, an HTG-777 FPGA board, and a custom DP input and interconnect board. Projector components include the XGA DMD chip and standard lens assembly. The remaining optics include the Fresnel lens and optical combiner (prism). Also present are a light sensor array and an RGB LED light synthesis module. Either a user or a camera can observe the optically combined physical and virtual visual targets.

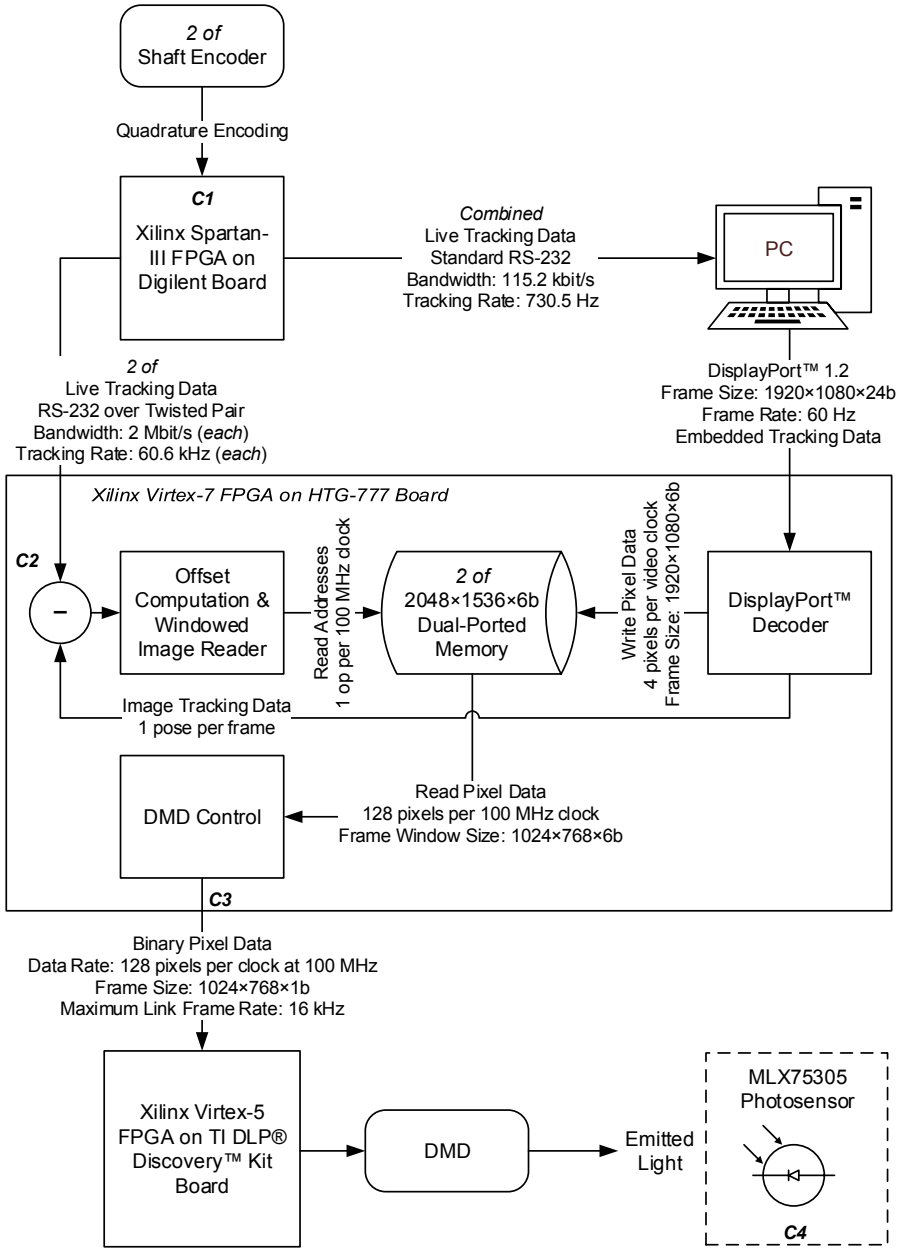


Figure 3.6: Implemented data path framework for image generation for the 6 bit/pixel grayscale system. Labels C1-C4 indicate oscilloscope channels assigned to measurement test points used during latency analysis. The photosensor (C4) is present and used during latency testing only.

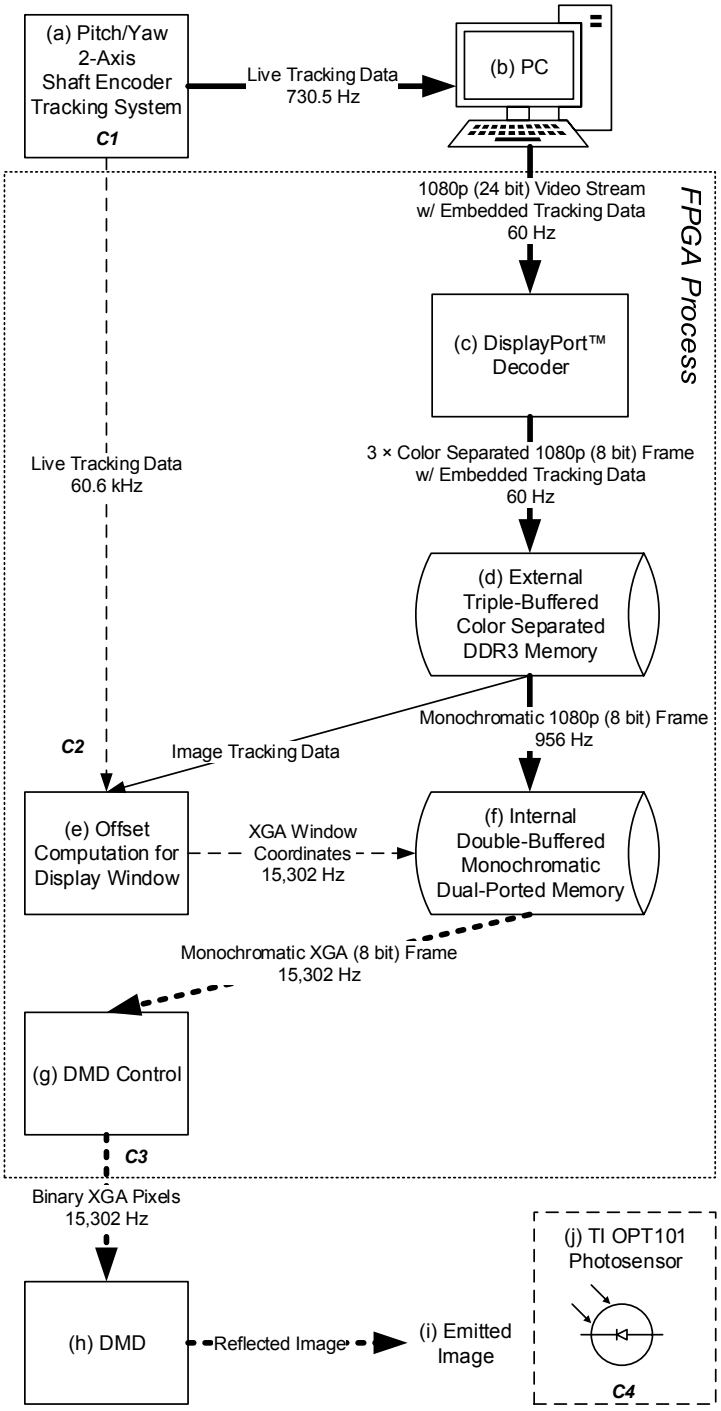


Figure 3.7: Subset of implemented data path diagram for image generation for the 16 bit/color system. Dashed lines indicate the tracking data pipeline: (a), (e), (f), (g), (h), (i). Thick lines indicate the video content pipeline: (a), (b), (c), (d), (f), (g), (h), (i). Segments of composite features (*e.g.*, thick dashed) indicate overlap (*e.g.*, video and tracking) in the pipelines. Labels C1-C4 indicate oscilloscope channels assigned to measurement test points used during latency analysis. The photosensor (j) is present and used during latency testing only.

To achieve low end-to-end latency, we divide the rendering pipeline between the PC and FPGA. As a side channel in the GPU rendered image, we embed the then-current tracking information used to render that image. To ensure that the embedded information remain synchronized to the rest of the image, we enabled VSYNC, which prevents prevent tearing, on the GPU. Independently, the display processor (FPGA) receives the live tracking information directly from the tracking system. When it is time to begin displaying a 16 kHz binary frame, it uses both the viewpoint calibration and difference in the two tracking states to transform the latest image to the current viewpoint. Essentially, the display processor is performing a post-rendering (and post-transmission) warp.

To simplify the processing required on the FPGA for this warp, we make some assumptions about the motion. Our tracking system only supports motion along two rotational dimensions (pitch and yaw), and the center of projection of the moving viewpoint is near to those two rotation axes. We also assume that the difference in angular pose between the render-time pose and the live pose is small. Thus we can simplify our warp into a 2D translation and crop of the GPU-supplied imagery; essentially we select a particular 1024×768 window of the full rendered image for each output display frame. As a result, that "excess" resolution provided by the GPU becomes padding for the offset computation engine. For each dimension, the offset from center (Δ) of this window in the input frame is simply the product of the difference of the rotation angles (ω) with the ratio of display pixel resolution (r) to field-of-view angle (α), as shown in Equation (3.1).

$$\Delta = (\omega_{\text{live}} - \omega_{\text{render}}) \times (r/\alpha) \tag{3.1}$$

In the event that the user managed to move faster than the resolution provides, out-of-bounds pixels are replaced by black by the FPGA logic, though we did not encounter this in our live tests.

Depending on the implemented color output of our display system, we used two different memory architectures for storing and processing the GPU-supplied imagery. These are described below.

6 bit/pixel Grayscale Implementation The Virtex®-7 FPGA receives the DP-provided frames at full resolution and stores them in a double-buffer of 2048×1536, dual-ported (simultaneous reads and writes at different clock rates) memories with 6 bit/pixel grayscale resolution. While we currently receive frames at a resolution less than the storable size, we selected a power-of-two width in order to support the highly parallelized modulation engine (*see* Section 4.3) since it requires reading 128 pixels simultaneously each clock cycle. In order to fit the pair of images in fast internal FPGA SRAM (Static Random Access Memory),

we had to limit the grayscale resolution to 6 bits/pixel, which we found visually sufficient. By using a double-buffered framebuffer in the FPGA, we are generally able to read from a buffer with a consistent tracking state. It is occasionally possible that the writing process may have switched buffers just after the reading process decided on which buffer to read from, but because the reading process (about 16 kHz) is orders of magnitude faster than the writing process (60 Hz), the discrepancy would last for only a small portion of one binary frame. If we were to use only a single framebuffer, then the tracking state would not be consistent across the whole image while the writer was modifying it, which would affect many binary frames. Using two framebuffers to store the desired imagery keeps the reading algorithm and address offset computation simpler, at the expense of doubling the memory required for storing desired imagery.

16 bit/color High-Dynamic Range Color Implementation Like the grayscale implementation, the Virtex®-7 FPGA receives the DP-provided frames at full resolution but instead first stores them in DRAM (Dynamic Random Access Memory) external to the FPGA chip. Each color (red, green, and blue) is stored separately, to optimize reading back of a single color at a time; our display is color-sequential. Each color, in turn, is read back into chip-internal, double-buffered, 2048 $\times$ 1080, 8 bit/pixel, dual-ported SRAM to support another highly parallelized modulation engine (*see* Section 4.4). To ensure consistent brightness among color channels, this DRAM-read/SRAM-write process, operating at 956 Hz, is phase-locked to the color sequence. The SRAM read process produces binary frames at 15 302 Hz. The SRAM reading process for producing offset computations is the same as in the grayscale implementation.

3.3.2 Visual Results: 6 bit/pixel Grayscale Implementation

We conducted an experiment to assess the efficacy of our implementation. More specifically, our aim is to determine how well our implementation maintains registration between real and augmented objects as the user turns his or her head.

3.3.2.1 Experimental Setup

A checkerboard target, with 4 cm squares, is placed within view of the HMD at a distance of about 1.7 m. Our display supports approximately a 37° diagonal FOV, which is much smaller than human vision though not much smaller than existing commercial OST-AR displays (*see* Section 2.2.1). A corresponding two-color

(white and dark gray) virtual checkerboard is displayed on the HMD. As the HMD pans, the two should stay locked together. Due to the nature of the checkerboard pattern, any misregistration will be quite obvious.

A GoPro® HERO4 Black camera was mounted in the HMD approximately at the location of a user's left eye. The camera is rigidly-attached to the HMD rig. The full assembly is shown in Figure 3.4.

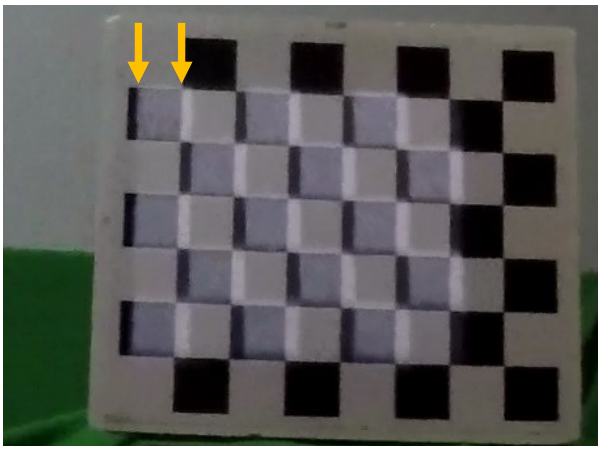
For the experiment, the HMD is rotated back-and-forth over a range of about 30°, which is approximately the horizontal field of view of the display. Rotation was performed by hand. To provide consistent angular velocity among trials, we mounted a protractor scale to the top of the HMD and used a metronome to time the back-and-forth movements. The HMD was rotated such that it moved 10 degrees per beat (for three beats) and then rested for one beat. We executed experiments with metronome speeds of 100 and 300 beats per minute. Thus we see consistent acceleration, constant velocity, and deceleration time periods in both directions. The resulting angular velocities equate to approximately 17 °/s and 50 °/s, respectively.

Videos were recorded at 240 Hz both with and without our latency compensation system enabled at each of the two angular velocities; some sample frames are shown in Figure 3.8. Our movement technique allowed for nearly-synchronized side-by-side comparisons of the respective tests. While the camera uses a rolling shutter, the horizontal axis of the image plane and the tested motion patterns were aligned, so comparisons along that axis remain valid.

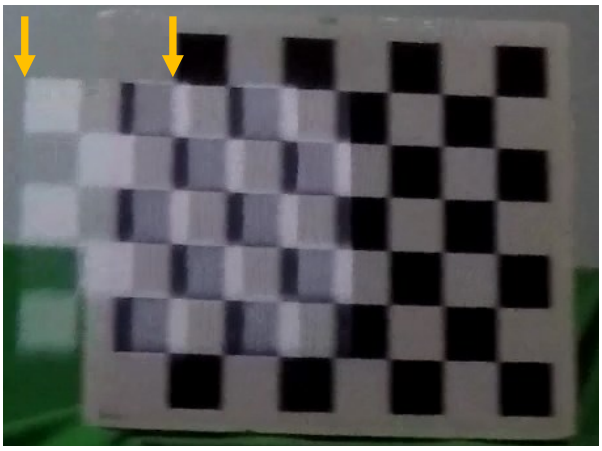
3.3.2.2 Discussion

Representative samples from the resulting videos under motion are shown in Figure 3.8. Figures 3.8(a) and 3.8(b) depict typical behavior with our latency compensation system disabled. Specifically, the image displayed on the HMD is updated only as new frames arrive from the GPU (at 60 Hz). In Figure 3.8(b), which was the 50 °/s case, we see that the displayed image lags the real-world image by almost three squares. In general, the augmented image lags behind the real-world image by an amount proportional to the rate of movement and the two worlds come into registration only *after* the motion ceases. It is worth noting that doubling the GPU frame rate to 120 Hz would not meaningfully mitigate this effect, as the lag would be reduced by at most half of the displacement.

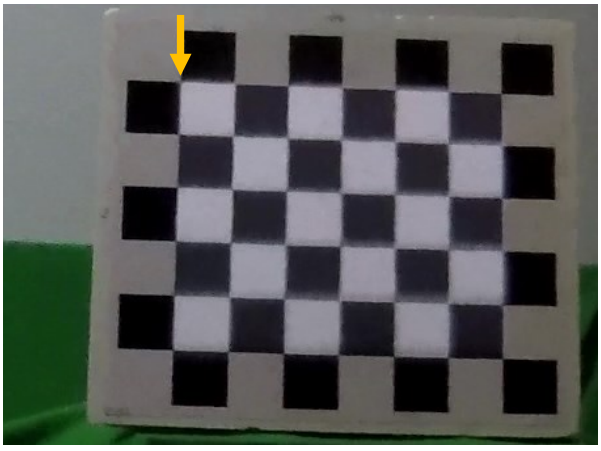
The photos in Figures 3.8(c) and 3.8(d) depict the same scenario but with our latency compensation system enabled. In this case, the real-time tracking data from the rotary encoders is used to perform our 2D offset operation, thereby reducing the misregistration due to motion. We observe that the augmented image is well-registered to the real-world object. Under the faster (50 °/s) pan, we also note that the motion



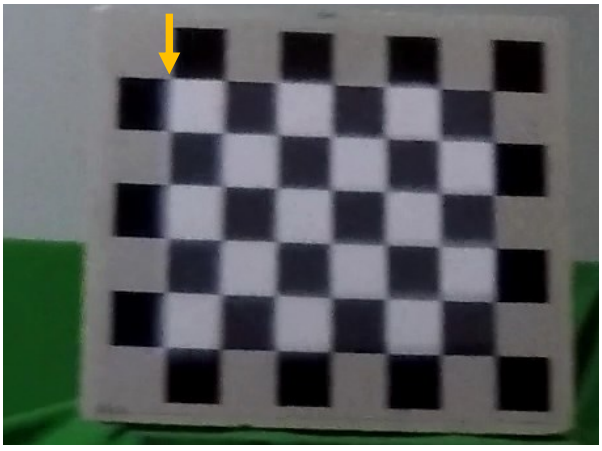
(a) Conventional display method. Panning velocity is approximately 17 °/s.



(b) Conventional display method. Panning velocity is approximately 50 °/s.



(c) Our display algorithm. Panning velocity is approximately 17 °/s.



(d) Our display algorithm. Panning velocity is approximately 50 °/s.

Figure 3.8: A comparison between conventional displays and our latency compensation algorithm using the 6 bit/pixel mode at two different panning velocities. In both cases, the yawing motion of the head-mounted display was right to left, causing the off-kilter checkerboard to appear to move left to right. The virtual overlay consists of a smaller checkerboard matched to the central region of the physical board. Note, as indicated by the yellow arrows, that while the overlay in the conventional display algorithm, (a) and (b), significantly lags behind the physical checkerboard, our algorithm, (c) and (d), displays the overlay on top of it and even provides some motion blur when the object moves quickly.

blur in the augmented image is consistent with that of the real-world object. The blur is smooth and natural, significantly due to updating the input to our modulation scheme at a high rate; as the tracking information has the potential to update every binary frame, the window on the desired image can also update. The eye or camera continuously integrates these moving binary frames, resulting in motion blur.

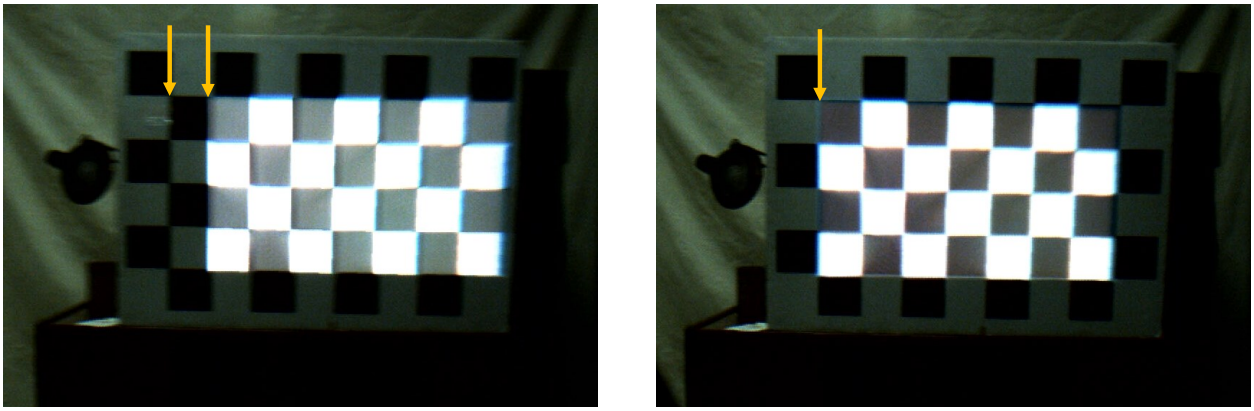
The virtual checkerboard could occasionally become misaligned to its physical counterpart; this typically happened at one end of the motion path. It is worth noting that at those faulty HMD poses, both the conventional display algorithm and our latency compensating algorithm would agree on the amount of misalignment if the HMD were not moving at the time. The source of this unfortunate misalignment was that the rig to which our HMD was mounted allowed for slight freedom of motion along axes that were not tracked, which repeatedly affected the calibration and caused the static divergence between the real and virtual worlds. This highlights the necessity for using high quality and accurate tracking and calibration schemes in order to support AR applications in which the two worlds must be rigidly aligned. It is worth noting that while actively moving, even in the poor calibration zones, our algorithm kept the two worlds closer.

3.3.3 Visual Results: 16 bit/color HDR Implementation

We repeated the checkerboard latency oscillation test from Section 3.3.2.1, though with a larger checkerboard (12 cm squares) located about at a distance of 3 m from the display (*see* Section 4.4.2 for a longer discussion of the room setup and camera used; *see* Figure 3.5 for an image of the system assembly). The HMD was rotated back and forth over a range of approximately 15° by hand at multiple target average velocities: 6 °/s, 12.5 °/s, 25 °/s, and 50 °/s. The movement range was smaller for this system in order to keep the larger checkerboard always inside the display's FOV. Videos were recorded at 45.45 Hz of both the latency-compensation enabled and disabled situations, and representative frames are shown in Figure 3.9. Results were similar as with the grayscale implementation. Furthermore, no color separation effects, which might be expected from a color sequential display, were visible in the results.

3.3.4 Latency Analysis: 6 bit/pixel Grayscale Implementation

The overarching aim of the present research is to minimize the delay between a change in the user's position or orientation and the corresponding change to what is seen on the display. Our system has been carefully



(a) Latency compensation disabled. The virtual checkerboard is out of phase by one complete checker (about 12 cm).

(b) Latency compensation enabled. The two checkerboards are aligned.

Figure 3.9: A comparison between our latency compensation algorithm and a conventional display using the 16 bit/color mode. The physical and virtual scenes each consists of matched checkerboards. In both images, the user viewpoint is turning left at about 16 °/s (during the 12.5 °/s-average test). Note, as indicated by the yellow arrows, the significant difference in alignment, due to latency, between the two images.

instrumented such that this delay can be measured precisely and such that sources of residual latency are identified and well characterized.

3.3.4.1 Signal Path

In our apparatus, head motion is detected using an optical rotary encoder with a resolution of 40 000 ticks per revolution (one tick is 0.009°). Considering the resolution and field of view of our display device, a rotation of about 3.45 ticks corresponds to a single pixel or, equivalently, the worst-case error in rotation measurement is about 0.29 pixels.

As shown in Figure 3.6, the rotary encoder emits quadrature-encoded data which is decoded by an FPGA (separate from the display control FPGA). The present angle is transmitted continuously to the display control FPGA via a pair of 2 Mbit/s serial links. The angle is received by the display control FPGA, where it is used to compute the offset to be applied to the displayed image. This computation takes about 4 µs and is performed between binary frames. Importantly, the compensation is performed once per binary frame using the most recently received angle.

Binary pixel data is streamed to the TI DLP® Discovery™ Kit's FPGA processor in four "quads," each comprising one quarter of the rows of mirrors on the DMD. After each quad is transmitted, an MCP signal is sent and, after a short delay, the respective set of mirrors begins to move as commanded.

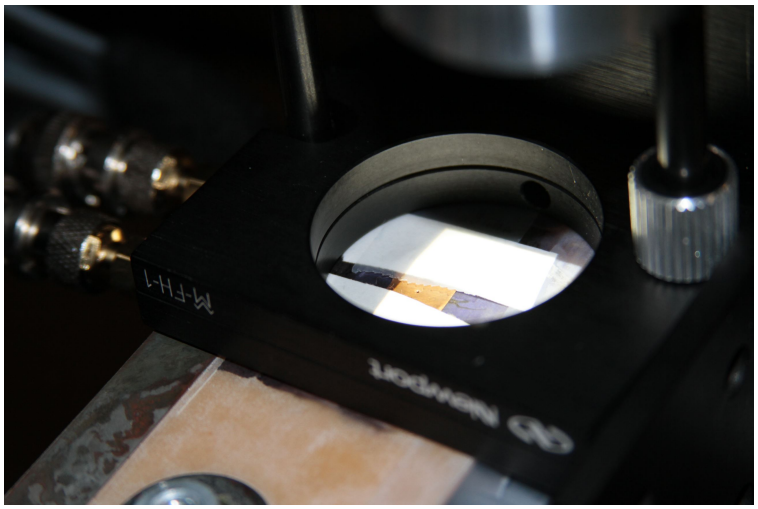


Figure 3.10: Light measurement sensor and barrier. The light-to-voltage sensor is located behind the pinhole (approximately 0.25 mm in width), visible in the copper segment between the two strips of white tape inside the circle. This circle has a diameter of 3.8 cm.

Finally, in order to detect the photons and measure the total latency, a precision light sensor, based upon the Melexis MLX75305[7] light-to-voltage Integrated Circuit (IC), was temporarily fitted onto the focal plane of the HMD, as illustrated in Figure 3.10. This sensor emits an analog voltage signal proportional to the luminous flux hitting it. We measured the −3 dB bandwidth of this sensor at about 55 kHz and the device is specified with a rise time of 6 µs. An opaque barrier is installed above the sensor with a small pinhole located just above the light-sensitive element in the IC; this ensures high directional sensitivity and reduces the effect of ambient light sources.

3.3.4.2 Measurement Instrumentation

To measure motion-to-photon latency, the system was configured such that it either displayed zero intensity or full intensity, depending on the angle of the HMD. We will refer to the angle at which the display changes from zero to full intensity as the trigger angle; the display shows black when positioned anywhere on one side of the trigger angle and white on the opposite side.

The apparatus is instrumented such that signals can be monitored at four critical points in the resulting signal path (*see* Figure 3.6) for the data path location of these test signals):

[7] `https://www.melexis.com/en/product/MLX75305/Light-to-Voltage-SensorEyeC`

1. *Motion Initiated*: The quadrature-decoding FPGA drives a pin high or low depending on the HMD orientation with respect to the trigger point. The delay between the physical movement and this signal is on the order of 100 ns. This signal is for instrumentation purposes only and is not in any way coupled to the display control.

2. *Data Received*: The display control FPGA drives a pin high or low based on the angle received over the serial link in relation to the trigger angle. This signal also corresponds to the new angle being latched into the offset-computation circuitry.

3. *Pixel Transmitted*: The value of the bit of the first pixel of each chunk of data streamed to the DLP® Kit's processor is mirrored to a pin on the control FPGA. In particular, if zero-intensity pixels are being sent, this line is low and if full-intensity pixels are being sent, this line is high. Thus, we observe the transmission of each quad whenever full-intensity is being displayed. This signal is also low when no data is being sent.

4. *Light Emitted*: The analog voltage from the light sensor, positioned on the focal plane of the HMD, is monitored. The sensor is located such that its aperture sees the first quad of the image.

We simultaneously probed these four signals using a Tektronix TDS 684B[8] oscilloscope (1 GHz analog bandwidth at 5 gigasamples per second). The scope was configured to trigger on the rising edge of Channel 1 (*i.e.*, when the HMD crosses the trigger angle, which will result in the display switching from black to white).

3.3.4.3 Latency Component Analysis

An example of a single resulting trace is shown in Figure 3.11; Channels 1 to 4 correspond to the respective signals described above. In this figure, the time base is set at 20 µs per division. The discussion below uses this trace to step through the characterization of the latency components in the present system.

Transmission of Position Data Referring to Channels 1 and 2 (green and blue, respectively), we observe a delay of about 27.5 µs between the motion event and receipt of the new angle by the display processor. This delay is due to the time taken for the angle information to be fully transmitted via the serial link. In the present implementation, the serial link runs at 2 Mbit/s; the transmission of the angle data plus serial framing overhead takes approximately 15 µs. The observed delay indicates that the crossing of the trigger angle

[8] `http://www.rlscientific.com/documenti/TDS684B.pdf`

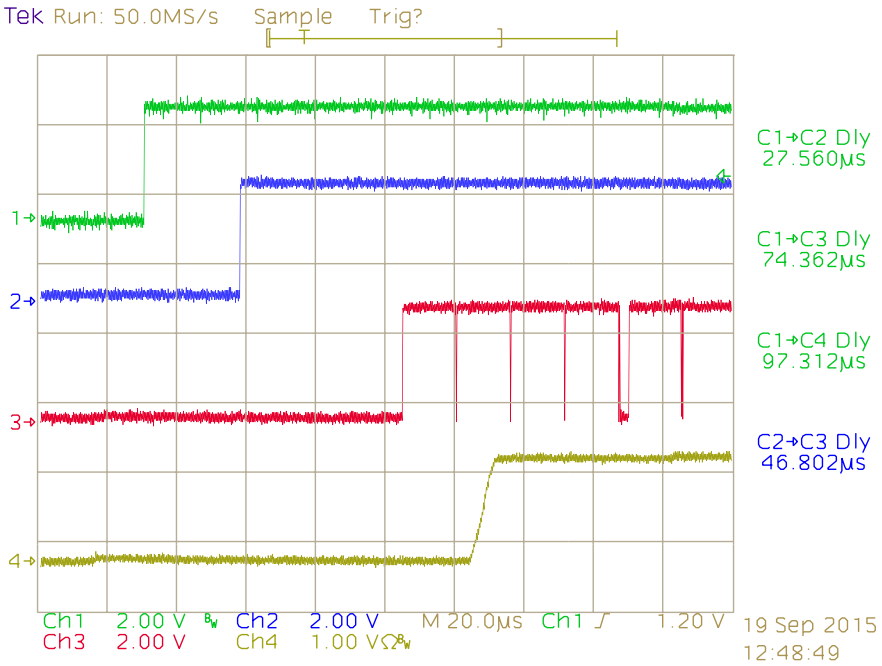


Figure 3.11: Sample motion-to-photon latency measurement of the 6 bit/pixel system. Channels from top to bottom: (1, green) motion initiated, (2, blue) data received by display processor, (3, red) pixel transmitted, and (4, yellow) light emitted. In this example, total latency (C1→C4) was 97.312 µs.

occurred shortly after the quadrature-decoding FPGA had begun transmitting a previous value. In general, with the present implementation, this delay component varies between 15 µs and 30 µs (22.5 µs average), depending on the precise moment the motion occurs. The magnitude of this delay is directly proportional to the bandwidth and latency between the tracking device and the display controller. Ideally, these devices would be closely coupled, bringing this latency term closer to zero.

Binary Frame Phase Referring to Channels 2 and 3 (blue and red, respectively), we observe a delay of about 46.8 µs between receipt of the trigger angle and the beginning of the transmission of the full-intensity data to the DLP® Kit. This delay is substantially due to the new angle arriving during the transmission of the previous binary frame (which is zero intensity and thus does not show on the red trace). Recall that the new angle cannot be applied until the next binary frame. Given that the binary frame rate is 15 552 Hz, the offset computation and binary frame transmission requires about 64.3 µs. Since offset computation requires about 4 µs, as indicated by the gap between binary frames in the C3 trace, this delay will range from 4 µs to 64.3 µs (34.2 µs average). A higher frame-rate device would result in a proportionally smaller value for this latency component.

Mirror Refresh Referring to Channels 3 and 4 (red and yellow, respectively), we see that the light sensor begins registering increased intensity just after the first quad of the binary frame has been committed (seen as the short dip in the red signal). This is as expected due to the operation of the DMD device. The 6 ns to 8 ns rise time in the light level is most probably due to variances in how fast the mirrors switch combined with the rated response time of the sensor itself. We declare photons to have arrived when this signal has risen about 50 %. This time interval is inherent to the DLP device (operating at this frame rate) and is equal to about 23 µs; doubling the binary frame rate would approximately halve this figure.

On the right-hand column of Figure 3.11, several trace-to-trace delays are shown. For this example trace, the value for "C1→C4 Dly," 97.3 µs, is the total latency from the detection of motion (the positive transition of Channel 1) to the detection of photons emitted by the display (the positive transition of Channel 4).

3.3.4.4 Latency Range Analysis

To confirm our characterization of the variable latency sources in the system, we switched the oscilloscope into "envelope" mode, where it essentially paints many successive triggered traces on top of each other. An example of this is seen in Figure 3.12; the positions and colors of the traces are the same as in the preceding discussion. Recalling that the time base is set to 20 µs per division, we can clearly see the fixed and variable components of the serial transmission delay: the minimum distance between the green and blue traces is about 15 µs and, over the many samples taken, the data arrival occurs within about a 15 µs window thereafter. This confirms our analysis of this latency component. Looking at Channel 4 (yellow trace), we see the range of times when light was detected. Measuring from the 50 % (vertical) point on the left- and right-most edges, we see that the range is consistent with the sums of the claimed variability in serial transmission and binary frame phase (about 75 µs). Putting together all of the latency data we find the results presented in Table 3.1.

Table 3.1: Summary of latency components for the 6 bit/pixel system.

Component	*Best*	*Worst*	*Average*
Serial Transmission (C1→C2)	15 µs	2×15 µs	1.5×15 µs
Binary Frame Phase (C2→C3)	4 µs	64.3 µs	(4 µs + 64.3 µs) / 2
Mirror Refresh (C3→C4)	23 µs	23 µs	23 µs
Motion-to-Photon (C1→C4)	42 µs	117.3 µs	79.7 µs

In summary, we have carefully instrumented and characterized the sources of latency between physical motion and photon-to-eye delivery within our system. Disregarding nanosecond-scale processing times and

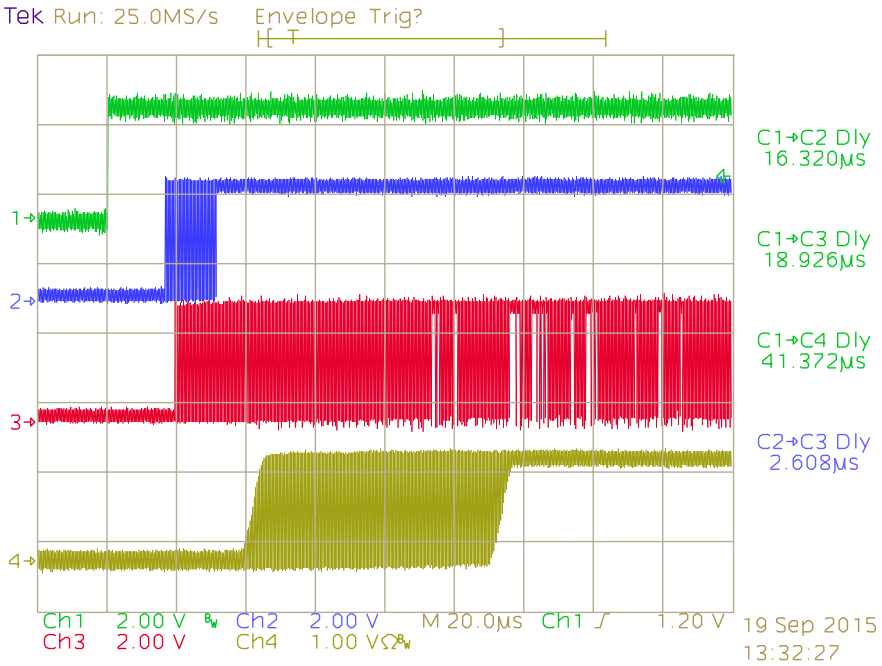


Figure 3.12: Accumulated samples of motion-to-photon latency measurement of the 6 bit/pixel system using same channel labels as Figure 3.11. Delay times at the right are the minimums of those captured. In this aggregate example, total latency (Channels 1 and 4) ranged between 41.372 µs and about 115 µs. This upper bound is indicated by the temporal displacement between the rise in C1 and the middle of the right-most rise in C4.

speed-of-light delays, all of the latencies are due to fundamental limitations of the present display device and the present interconnect between our tracking sensor and the display processor. Our system can consume and make use of tracking data at rates of tens of kHz. From a technology standpoint, the bottleneck in extending this system is in acquiring and coupling tracking data at least an order of magnitude faster than today's technology. It is worth noting that even with this very fast and low latency tracking system, its contribution to the overall latency ranged between 1/4 and 1/3, which is quite significant.

3.3.5 Latency Analysis: 16 bit/color HDR Implementation

We examine end-to-end video latency (the time it takes for a change in appearance generated by the PC to become visible on the display) and motion latency (the time it takes for a viewpoint change to become visible on the display). Video latency affects the color or internal animation appearance of the displayed virtual imagery, while the motion latency affects its registration (alignment) to real scene objects. Causes of latency in the system described here vary only in a few well defined ways from the latencies described of the previous similar system (*see* Section 3.3.4).

The video latency in the color HDR system is higher due to the triple buffering pipeline stage—the component shown as Figure 3.7(d)—which stores the color frames in off-chip DRAM. Since we read from the external DRAM at about 956 Hz, this adds about 1 ms to the video latency compared to using only internal memory (if there would have been enough available).

To measure the end-to-end motion-to-photon latency, we used a similar process as for the 6 bit/pixel system (*see* Section 3.3.4), with differences described below.

3.3.5.1 Signal Path

The HDR system uses the same tracking system and a similar offset computation as the grayscale system (*see* Section 3.3.4.1). In addition, the binary pixel data is transmitted to the TI DLP® Discovery™ Kit in four "quads," like the grayscale system. However, the HDR system's illuminator is not constantly on. Instead, after each complete frame is transmitted and after the mirrors stabilize, the appropriately colored LED emits a brief light pulse (10 µs) of a specified intensity. We replaced the latency measurement light sensor with the TI OPT101[9], driven at 30 V and combined with an analog gain circuit with a gain factor of 0.05, which provided about 58 kHz of bandwidth; as a result, some RC (Resistor-Capacitor) effects are visible in the traces.

3.3.5.2 Measurement Instrumentation

To measure the motion-to-photon latency, we used the same system configuration as used with the grayscale system (*see* Section 3.3.4.2). We also used the same oscilloscope with similar trigger and measurement settings, though the vertical scales of some signals were different. In addition, the details of one test signal differed, as listed below:

4. *Light Emitted*: The analog voltage of the light sensor, positioned below the display's lens, is monitored. The active area of the sensor is located in the approximate center of the projected FOV.

In addition, the system was configured so that only the green LED was used and the output intensity for every LED pulse was identical. These changes do not affect the latency of the system as the same main data path remained consistent. Instead, it enables the detected analog height of each pulse to be consistent, which is especially useful for reading data from the oscilloscope.

[9] `http://www.ti.com/product/OPT101`

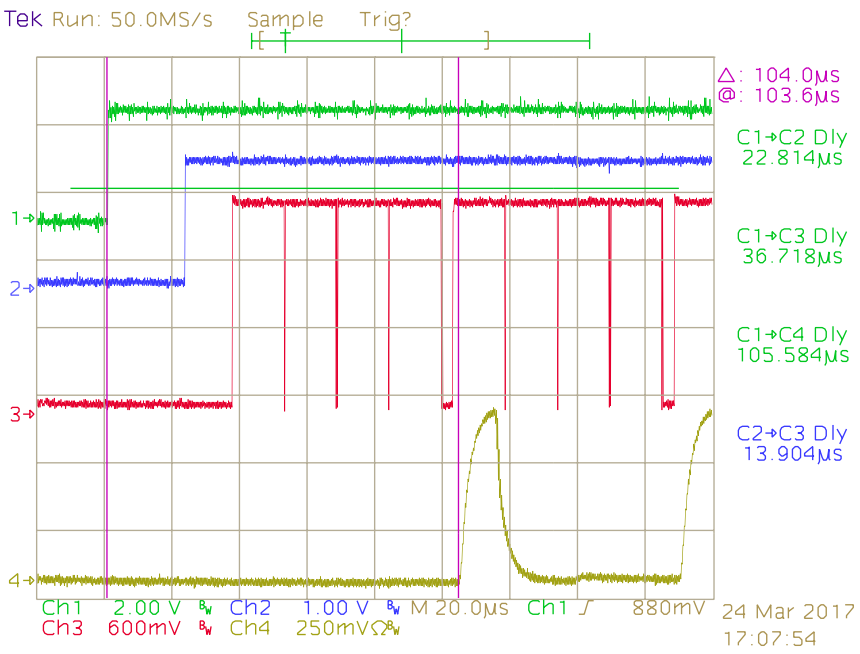


Figure 3.13: Sample motion-to-photon latency measurement of the 16 bit/color system using same channel labels as Figure 3.11. In this example, total latency (C1→C4) was 105.584 µs, which is approximately confirmed by the vertical cursor bars (purple). Note also that two binary frames with visible pixels are present, as indicated by the two pulses of C4.

3.3.5.3 Latency Component Analysis

An example of a single resulting trace is shown in Figure 3.13, where the channels correspond to the signals described in Sections 3.3.4.2 and 3.3.5.2. The basic analysis described in Section 3.3.4.3 is appropriate for this trace, though with different times and the differences described below.

Transmission of Position Data Referring to Channels 1 and 2 (green and blue, respectively), we observe a delay of about 22.8 µs between the motion event and the receipt of the new angle. Like the grayscale system, the HDR system uses the same transmission pathway. Thus the observed delay is consistent with the expected variation between 15 µs and 30 µs (22.5 µs average).

Binary Frame Phase Referring to Channels 2 and 3 (blue and red, respectively), we observe a delay of about 13.9 µs between the receipt of the trigger angle and the beginning of transmission. In this case, the new angle was received near the end of the previous binary frame and will be applied to the first binary frame pictured with non-zero pixels. Since the binary frame rate of this system is 15 302 Hz, the combination

of offset computation and binary frame transmission is approximately 65.4 µs. As the offset computation requires about 4 µs, this delay will range from 4 µs to 65.4 µs.

Mirror Stabilization & LED Pulse Referring to Channels 3 and 4 (red and yellow, respectively), we see that the light sensor begins registering increased intensity following a delay after the final quad of the binary frame is transmitted; we see a delay of about 68.9 µs in the sample trace. This delay is much greater than for the grayscale system since the LED pulse follows a complete binary frame rather than being observable shortly after the first quad's MCP. The logic is configured to use a delay of 4.5 µs following the final quad, and that 4.5 µs delay is consistent with the difference in time between the final fall of one binary frame (C3) and the start of the rise in the light sensor (C4). In reviewing multiple traces, this delay is relatively constant, with a value of about 68.8 µs.

3.3.5.4 Latency Range Analysis

Like for the grayscale system (*see* Section 3.3.5.4), to confirm our characterization of the variable latency sources in the system, we switched the oscilloscope into "envelope" mode. An example of this is visible in Figure 3.14. However, as a result of the LED pulses, which have both a rise and fall in measured voltage, the overall range is not visible. As a result, the range analysis is mostly dependent on our characterization.

The end-to-end, motion-to-photon latency of this system is not affected by the additional buffering (required for storing the received RGB image), as the view-direction driven region selection—represented by Figure 3.7(e)—occurs after the buffering. However, compared with the 6 bit/pixel system, for every binary frame (15 302 Hz), delay is introduced by waiting to activate the color LED until the DMD's micromirrors have stabilized after updating the binary image. This new delay is relative to the start of binary pixel transmission. The illuminator used in the grayscale method was constantly on, so its end-to-end latency only required a quarter of the frame to be processed by the DMD before becoming observable. The 16 bit/color system must wait for the entire binary frame to be loaded and all of the mirrors to be flipped and stabilized before it can activate the LED, at which time we observe the output light. A summary of these latency components is presented in Table 3.2.

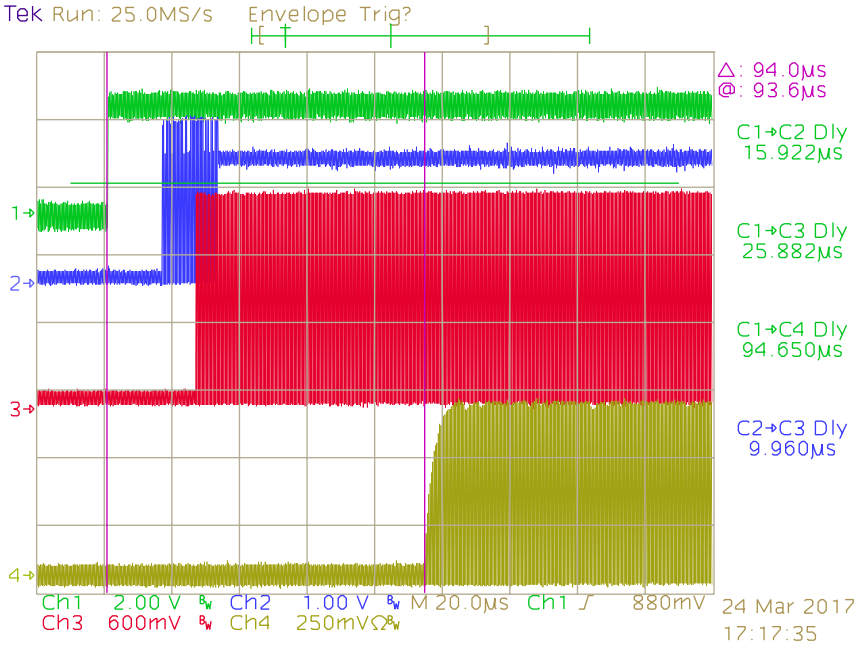


Figure 3.14: Accumulated samples of motion-to-photon latency measurement of the 16 bit/color system using same channel labels as Figure 3.11. Delay times at the right are the minimums of those captured. In this aggregate example, only the minimal total latency (Channels 1 and 4) of the collected samples is clearly visible and, in this case, has a duration of 94.650 µs.

Table 3.2: Summary of latency components for the 16 bit/color system.

Component	*Best*	*Worst*	*Average*
Serial Transmission (C1→C2)	15 µs	2×15 µs	1.5×15 µs
Binary Frame Phase (C2→C3)	4 µs	65.4 µs	(4 µs + 65.4 µs) / 2
Mirror Stabilization & LED Pulse (C3→C4)	68.8 µs	68.8 µs	68.8 µs
Motion-to-Photon (C1→C4)	83.8 µs	164.2 µs	124 µs

CHAPTER 4

HIGH-DYNAMIC RANGE FOR LOW LATENCY DISPLAYS

4.1 Introduction

Implementing a low latency OST-AR display can be divided into two parts: deciding what to show and showing it. Chapter 3 presents a method for deciding what to show, but that method requires a very high frequency display (on the order of kHz) to do so. Conventional displays have typically been 60 Hz, though some VR headsets operate at 90 Hz, and some televisions display at up to 120 Hz. If the display does not operate as fast as an in-display latency compensation algorithm, some of compensation's effectiveness would be lost.

DMDs (or other fast SLMs) are likely candidates for implementing such a high frequency display; conventional DMD projectors, though typically used for presenting 60 Hz video, modulate (*see* Section 2.3) the mirrors at rates up to 32 kHz. Instead of planning the modulation pattern based on conventional input, low latency implementations using DMDs must continuously react to the changing input during each modulation step.

Furthermore, the modulation scheme used for the target display should support generating HDR output. The real world has wide variation in appearance and brightness, with the range including deep shadow and bright sunshine. Any synthetic imagery must also match its surroundings to have a chance at maintaining a coherent appearance. A conventional 8 bit/color display has little chance of providing a matching appearance in these ranges.

This chapter presents several methods and corresponding real-time implementations that move towards providing all of these necessary features: low latency; high frequency; and HDR, RGB color. Section 4.2 presents a prototype implementation of an error-based modulation approach that we determined too expensive

for supporting real-time low latency behavior. Section 4.3 presents an improvement which supports low latency, high frequency display, but could not support HDR color at reasonable rates. As a culmination, Section 4.4 presents an implementation with all of the listed necessary features, and Section 4.5 extends it with scene-aware brightness adaption. Lastly, Section 4.6 provides a summary of all of these methods.

4.1.1 Contributions

- A real-time implementation of the error-based modulation approach (EEM) originally proposed by Zheng et al. (2014).
- A real-time implementation of a grayscale modulation approach (PR-PDM) for low latency display.
- A real-time implementation of a color, HDR modulation approach (DDS) for low latency display with a linear (with color depth) integration frequency.

4.2 Display Scheme: Estimated Error Modulation

Zheng et al. (2014) introduced a modulation approach that was based on PDM but incorporated a model of the human eye's response in an attempt to plan for visual response delays: Estimated Error Modulation (EEM). In particular, their approach assumes that the human visual system remembers the average of the past few frames over a sliding window of time (*e.g.*, 64 binary frames). Thus, a new binary value is produced with the assumption that the previous 64 binary values are still contributing to the eye's response.

Since every binary decision for each output value is dependent on both the desired modulation output, d, and the previously emitted values, execution of this method does not follow the summation structure presented in Equation (2.1) in Section 2.4. However, if one instead considers measuring the generated value at some time defined to be zero as g_{EEM}—shown in Equation (4.1a)—then this modulation scheme, m_{EEM}, follows Equation (4.1b).

$$g_{\text{EEM}} = \sum_{s=0}^{2^n-1} b \times m_{\text{EEM}}(d, s) \tag{4.1a}$$

$$m_{\text{EEM}}(d, s) = \begin{cases} 1 & \text{if } d > \sum_{t=s-2^n}^{s-1} m_{\text{EEM}}(d, t) \\ 0 & \text{otherwise} \end{cases} \tag{4.1b}$$

Since producing a new modulation value requires knowing the recent history of emitted modulation values, this approach must store the entire history of the most recent 2^n binary frames (*i.e.*, values over the entire integration interval) so that it can compute the accumulated value over a sliding window in time. An optimization exists, while not reducing the memory storage cost (instead, it increases it), that reduces the bandwidth cost by a factor of n: store an accumulation value of that history. In this way, for each new binary frame, a pixel's computation requires several memory accesses: reading the desired grayscale value, reading the current accumulated value, reading one old binary value from 64 frames prior (so it can be subtracted from the windowed accumulation), storing the new accumulated value, and storing the new computed binary value. All of this imposes a large requirement on memory bandwidth, thereby limiting the frame rate at which the system can be operated given a fixed memory bandwidth.

4.2.1 Results

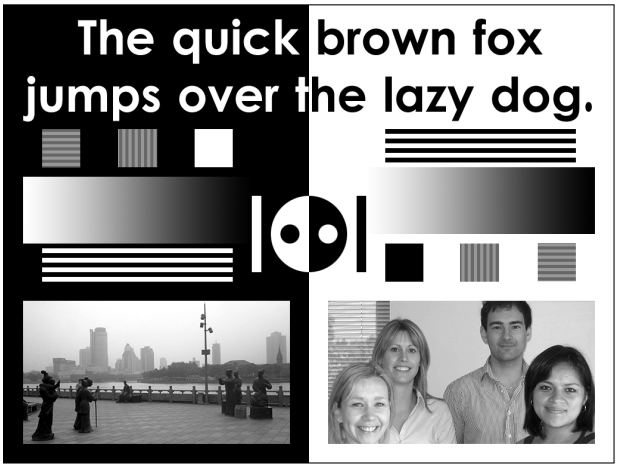
To test EEM, we only used the TI DLP® Discovery™ Kit 4100's Virtex®-5 board and DMD discussed in Section 3.3.1. In the images presented, the output from the DMD display was projected onto a small piece of paper rather than used as an HMD.

The memory required to store the desired, accumulated, and historical images vastly exceeded that of the Virtex®-5's internal SRAM, so all memory operations occurred in external DDR2 DRAM. This limited the effective binary frame rate to 964 Hz, with a 64 frame (6 bit/pixel grayscale) integration frequency of about 15 Hz. This resulted in a very noticeable flicker in the output. Reducing the image resolution to quarter-XGA (512×384) increased the binary frame rate to 4086 Hz, a 64 frame integration frequency of 63.8 Hz, which eliminated the human perceptible flicker.

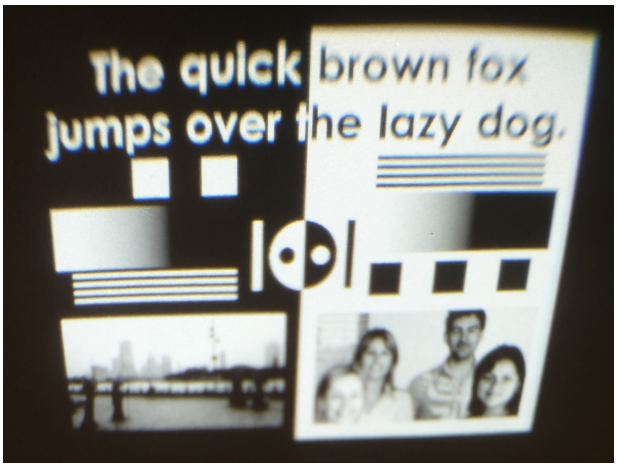
As shown in Figure 4.1, when the desired input to the display was a static image, the output also appeared stable. However, any dynamic change to the input produced poor output, as pictured in the sequence in Figure 4.2. Essentially, any region of the output imagery that contained non-zero or non-maximal desired values would show copies of previously present imagery at the same location, corrupting the output.

4.2.2 Limitations

The reason that echoes of previous images appear in the output is a bit subtle. Suppose the system starts where two neighboring pixels are each black (0 % intensity) and white (100 %). The expected and realized output will be sequences of "0" and "1" values respectively. Now, suppose that both pixels' desired values



(a) Input test pattern. Photos from Wikimedia Commons: "'Leaving home' Statue in Ningbo" ©Siyuwj; "Client Advisory Team Photo" ©Mshannon, both CC-BY-SA-3.0 / GFDL.



(b) Output result when input was stable from power-on.

Figure 4.1: Input and output imagery for stable tests with the EEM scheme.

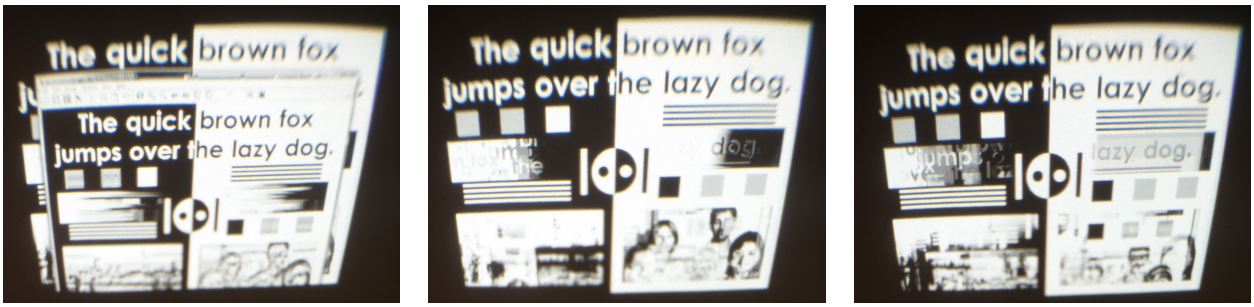


Figure 4.2: Output imagery for tests with the EEM scheme where the input imagery varied over time (increasing from left to right). The input consisted of a sequence where the input in Figure 4.1(a) filled the background and a window containing a smaller version of the same image was moved across the screen.

change to 50 % intensity. The two pixels will have different histories and integrated accumulation values, and so the new sequences will differ. The formerly black pixel has been too dark, historically, and so it will start generating "1" values, until the integrated threshold reaches 50 %. On the other hand, the formerly white pixel has been too bright, historically, and so it will start generating "0" values, until the integrated threshold lowers to 50 %. From then on, both pixels will slowly oscillate between the binary values "0" and "1" but out of phase. When the integrated rate is too low (*i.e.*, human perceptible), then this high-contrast edge will be visible, despite the integrated values for both pixels being equal. In effect, the attempt to account for the eye's memory has resulted in adding feedback into the computation, thereby causing highly undesirable oscillatory behavior.

While the idea of incorporating the eye's response delay into PDM is an interesting one, in practice it often leads to unseemly flicker and historical echoes. This makes this algorithm a much more resource and memory expensive implementation of PWM rather than the more optimal PDM.

4.3 Display Scheme: Pseudorandom Pulse Density Modulation

We base our approach on PDM, but deliberately introduce randomness into the modulation; we call this Pseudorandom Pulse Density Modulation (PR-PDM). The goal was to eliminate the output flicker of PWM-like approaches (including those by Zheng et al. [2014] and in Section 4.2) yet obtain an implementation complexity and memory cost much less than that of ideal PDM (*i.e.*, Delta-Sigma modulation). While we assume that a human eye would integrate over time, we do not explicitly model for it, but instead target flicker frequencies that are not human perceptible.

The key idea is that for each binary frame, a pixel's output value is chosen to be a "1" or a "0" randomly using a probability proportional to its gray value. In particular, we use a hardware-implemented pseudorandom number generator, using the well-known Linear-Feedback Shift Register (LFSR) (Koeter, 1996) approach, to generate pseudorandom numbers at the same bit depth as the desired imagery. An n bit LFSR generates sequences of values ranging from 0 to $2^n - 2$ (or from 1 to $2^n - 1$, depending on implementation) in a repeating but permuted order. Using the notation from Section 2.4 of Equation (2.1), then the modulation scheme follows Equations (4.2a) and (4.2b), where $R_n(s)$ is the s-th random number of an n bit LFSR.

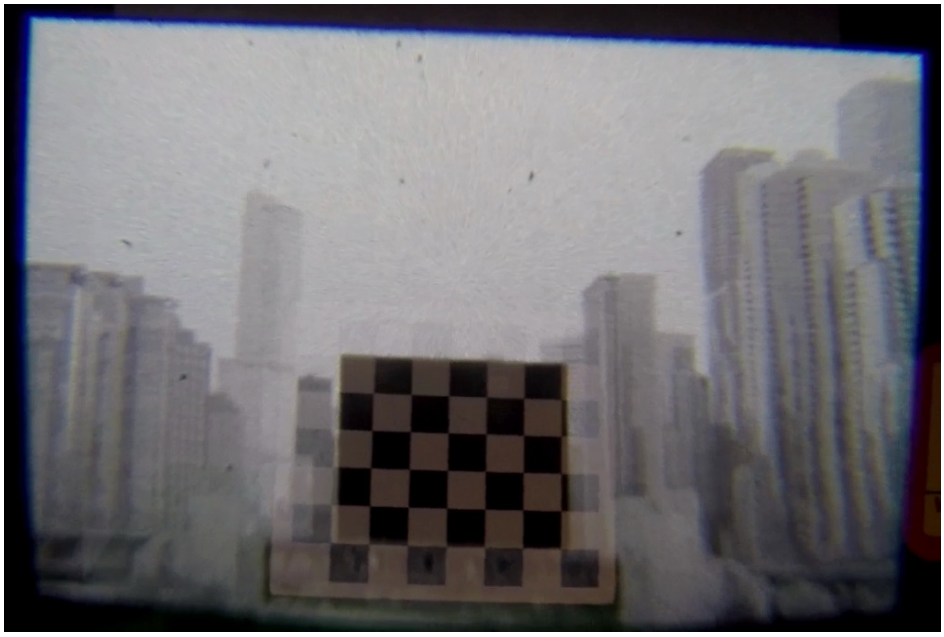


Figure 4.3: A sample frame using our latency compensation algorithm under fast panning motion (50 °/s) where the majority of the field of view is dominated by a off-kilter grayscale planar image of a city skyline, in which a small cutout region is registered to the physical checkerboard. Note that the cutout and the checkerboard interior are well aligned and details in the city photo are visible.

$$g_{\text{PR-PDM}} = \sum_{s=0}^{2^n-2} b \times m_{\text{PR-PDM}}(d, s) \tag{4.2a}$$

$$m_{\text{PR-PDM}}(d, s) = \begin{cases} 1 & \text{if } R_n(s) < d \\ 0 & \text{otherwise} \end{cases} \tag{4.2b}$$

An example sequence of simulated PR-PDM output pulses is illustrated in Figure 4.14, and a sample capture of the output pulses measured by the light sensor from Section 3.3.4.1 are illustrated in Figure 4.4.

4.3.1 Results

An example output image of this 6 bit/pixel grayscale system (*see* Section 3.3.1) is presented in Figure 4.3. The algorithm is able to show clear grayscale output, where appropriate, without any human perceptible flicker. Furthermore, it maintains low latency operation.

Somewhat unexpectedly, introducing randomness into the modulation approach also has a significant implementation benefit: rather than increasing design complexity, it actually results in a much more efficient implementation than Zheng et al.'s (2014) or classic Delta-Sigma modulation. Our approach only requires a

single memory access for a computation. The desired grayscale value is read from memory, compared with a random number, and a binary output is generated. Thus, the demand on memory bandwidth is dramatically lower, allowing us to achieve a significantly higher frame rate, at XGA output resolution, of 4340 Hz on the Virtex®-5 and 15 552 Hz on the Virtex®-7. For 6 bit grayscale, and double-buffering, we use about 30 % of the memory bandwidth and less than 15 % of the memory storage that EEM would require. Keeping our memory usage low enables us to use fast FPGA SRAM; using external DRAM would have lowered throughput and restricted the frame rate accordingly.

Our randomized approach is also simpler to implement than pure Delta-Sigma modulation. The latter requires two memory reads and one store per computation: reading the desired grayscale value, reading the accumulated value, and storing the updated accumulated value. Thus, for double-buffering, our approach requires only one-third the memory bandwidth and only two-thirds of the memory storage of Delta-Sigma.

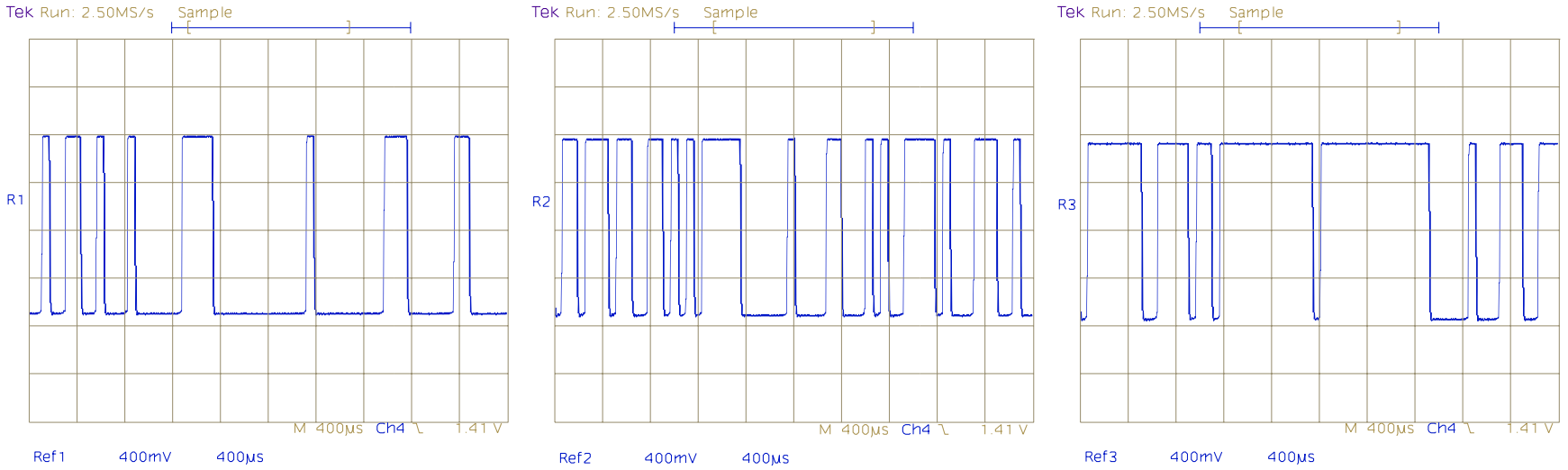
In summary, our approach of introducing randomness into modulation provides two crucial benefits: no perceptible output flicker and a much simpler hardware implementation.

4.3.1.1 Perceptual Comparison of PWM and PR-PDM

The perceptual advantage of PR-PDM over PWM can be understood in terms of the noise spectrum of the modulated light output; in short, PR-PDM moves modulation noise to higher frequencies, eliminating perceptible flicker. Referring to Figure 4.4(a), showing light intensity over time for a 25 %-gray value, we see that the light alternates between "on" and "off" and is, on average, "on" 25 % of the time. If we had continuous control over the light intensity (*i.e.*, if we could control the current delivered to the illuminator), then we could directly generate a 25 % value and the intensity value would be a straight line—a DC (Direct Current) value—at 25 % of the amplitude of the signal shown in the image. Any deviation from this DC value (the desired signal) can be thought of as noise; in the present case, this noise is due to the modulation scheme. If enough of the modulation noise is at a perceptible (low) frequency, then we will see flicker. The advantage of PR-PDM over PWM is that PR-PDM moves the noise power to higher frequencies.

To demonstrate the noise-spectral difference between these modulation schemes, we performed a simulation in LTspice[1] where a DC signal, at 25 % of full-scale, is modulated by 6 bit PR-PDM and PWM at a 16 kHz sample frequency. The simulation was carefully set up to be faithful to the PR-PDM modulation

[1] `http://www.linear.com/designtools/software/`



(a) PR-PDM 25 %

(b) PR-PDM 50 %

(c) PR-PDM 75 %

Figure 4.4: Measured 6 bit grayscale modulation patterns for PR-PDM (blue) by desired intensity level using the light sensor. A high voltage indicates the pixels are “on,” and a low voltage indicates the pixels are “off.”

scheme implemented in the actual display: the simulated PR-PDM output in Figure 4.5(a) is identical to the output in Figure 4.4(a) captured on the actual display. Figure 4.5(b) shows the same signals passed through an RC low-pass filter with a pass band of 125 Hz (chosen arbitrarily for illustrative purposes). While the two waveforms (PR-PDM and PWM) produce the same average value (about 1.25 V), it is clear that the PR-PDM output has a much smaller swing around the average, and therefore much less noise overall, when compared to PWM.

The distinction between these methods is easier to visualize in the frequency domain. Figure 4.5(c) is a plot of the spectra of the modulated, unfiltered signals shown in Figure 4.5(a). Both the PWM and PR-PDM have significant power at 254 Hz and harmonics of that frequency. Notice, however, that the power at this frequency in PWM is about 15 dB (32 times) higher than PR-PDM. It is apparent that the noise introduced by PR-PDM is contained primarily at frequencies at or above 1 kHz (and is thus far less likely to be perceived). (In the ideal amplitude modulated case, the output would only have a DC term.)

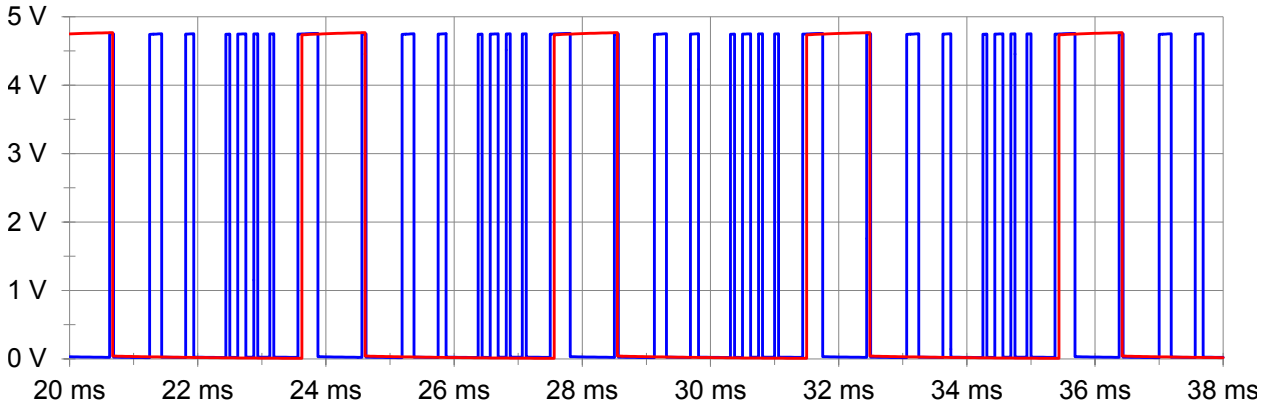
4.3.2 Limitations

The primary limitation of PR-PDM, like other PDMs, is that it is an exponential algorithm. Showing 6 bit imagery on DMDs requires 63 binary frames. On our display, this results in an integration frequency of about 247 Hz ($15\,552\,\mathrm{Hz}/63$) or interval of about 4.1 ms. Extending this to RGB color would reduce it to about 82 Hz, which is still in the realm of human flicker-fusion. However, extending from 6 bit/color to 16 bit/color would result in a worthless integration frame rate of 0.079 Hz or more than twelve seconds per sequence. Displaying HDR using DMDs requires a different modulation scheme.

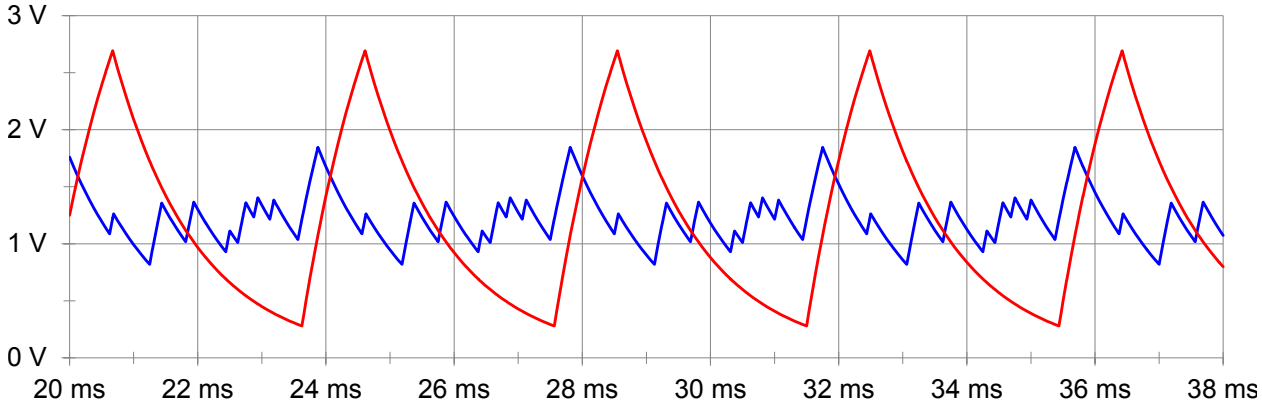
4.4 Display Scheme: Direct Digital Synthesis

Generating color HDR imagery on a DMD (or any binary SLM) at rates sufficient for low latency requires a process faster than the *exponential* methods described in Sections 2.4, 4.2, and 4.3. As an alternative, we directly control the brightness of our illuminator over a period of time via Direct Digital Synthesis (DDS) and spatially select illumination levels.

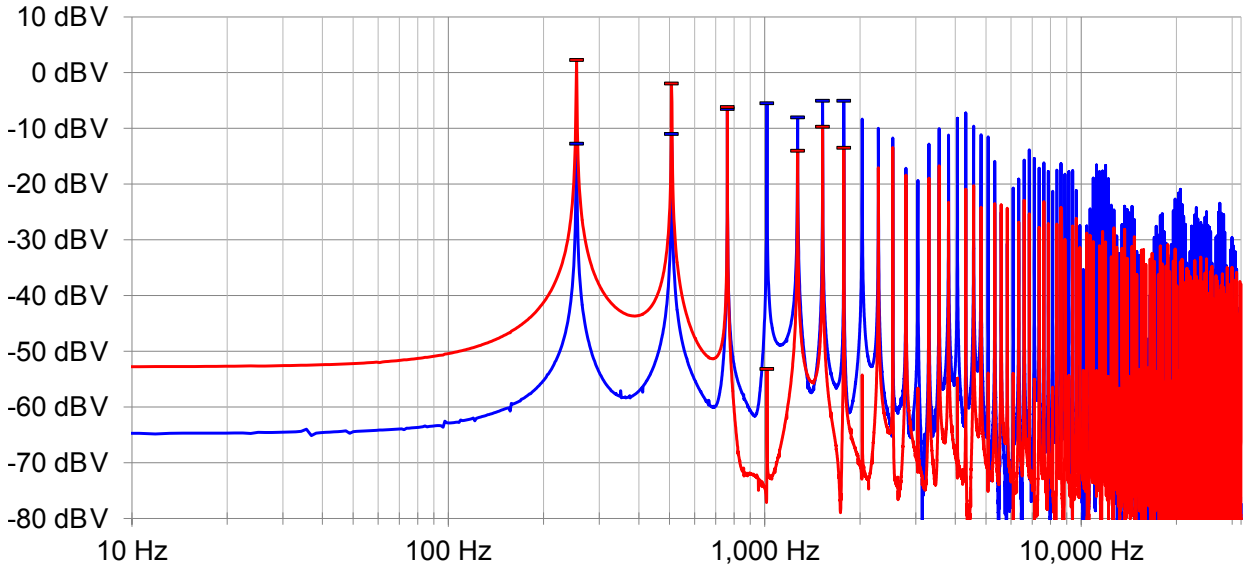
For n bits of intensity resolution, we generate n levels of illumination over a period of n steps. Each step in the sequence is twice the intensity of the previous step. A sequence of n binary images is displayed on the DMD, synchronized with the illumination so that each pixel of the display temporally selects the powers



(a) Unfiltered signals.



(b) Signals in (a) with RC low-pass filter (a pass band of 125 Hz) applied.



(c) Spectra of simulated signals in (a). For clarity, the local maximums of the first several harmonics are highlighted with dash marks of the same color.

Figure 4.5: Simulated signals and spectra of 6 bit PWM (red) and PR-PDM (blue) for targeting a constant DC signal of 25 % intensity (targeting about 1.25 V out of about 5 V).

of two corresponding to bits in its binary value. Integrating these powers of two over the n temporal steps yields the desired gray level for each pixel. Repeating the steps for each color produces a full color image with n bits per color. Expressed in the form presented in Equation (2.1) from Section 2.4, our scheme follows Equation (4.3).

$$g_{\text{DDS}} = \sum_{s=0}^{n-1} b_{\text{DDS}}(s) \times m_{\text{DDS}}(d, s) \tag{4.3}$$

Functions $b_{\text{DDS}}(s)$ and $m_{\text{DDS}}(d, s)$ can be defined as shown in Equations (4.4a) and (4.4b), where $\text{bit}(d, s)$ returns the 0-based s-th Least Significant Bit (LSB) of the binary value of d.

$$b_{\text{DDS}}(s) = 2^s \tag{4.4a}$$

$$m_{\text{DDS}}(d, s) = \text{bit}(d, s) \tag{4.4b}$$

Combining Equations (4.3), (4.4a), and (4.4b) yields Equation (4.5).

$$g_{\text{DDS}} = \sum_{s=0}^{n-1} 2^s \times \text{bit}(d, s) \tag{4.5}$$

Unlike previously presented PTM techniques, this process is linear with the number of illumination levels, which is the same as the number bits in the values of d.

The challenge of this approach is generating the n light levels at short intervals (*i.e.*, a fraction of one binary frame's duration). Typical methods for generating variable light levels with LEDs use PWM, but for 16 bit intensity generation at 16 kHz would require the LED's intensity modulator to operate at over 6 GHz, which is infeasible for high-intensity LEDs.

A key enabler for the present display is a custom high-speed, precision, digitally-controlled, DDS light module (shown at the lower-right of Figure 3.5). This module comprises a high-intensity RGB LED[2] and three independent linear current mode driver circuits controlled by 16 bit Digital-to-Analog Converters (DACs). Arbitrary intensities can be generated with turn-on and turn-off times on the order of 300 ns. We use a 10 µs pulse during each binary frame. Switching times are minimized, in part, by maintaining the LEDs in

[2]Our system used the Avago ASMT-MT00 RGB LED, produced by Broadcom: `https://www.broadcom.com/products/leds-and-displays/high-power-leds/moonstone-3w-led-emitter/asmt-mt00-00001`.

a partially forward-biased state, effectively eliminating junction capacitance. Our display targets 16 bit/color; however, to maintain color balance with slightly non-linear behaving hardware, we do not use the full extent of this dynamic range on all color channels: we achieved at least 15 bit/color on all channels. Nonetheless, this exceeds the dynamic range of current commercial standards for HDR and is of a similar order of color depth as other HDR research displays (*see* Section 2.2.5).

In order to display RGB HDR with 16 bit/color depth, our FPGA's display process iterates among each bit of each primary color, writing the bit values to the DMD and emitting a synchronized pulse of appropriate color and brightness from the light synthesizer, updated for each of the 48 binary frames. In theory, one could interleave the colors and bitplanes in any order, but we use a color sequential order due to the FPGA's memory size and speed limitations. To maintain a low latency response to the user's motion, the latency mitigation operation to realign the virtual imagery to match the physical world operates on a binary frame basis (15 302 Hz); thus the value of d varies in both time and image space. A complete data path diagram is presented in Figure 4.6; with the exception of the latency measurement tools, this diagram is a superset of the diagram in Figure 3.7.

4.4.1 Color Calibration

Producing an appropriate white balance for a display requires a color calibration procedure. Conventionally, one might use the standard 3:6:1 (RGB) ratio used in broadcast TV systems, but it is not always appropriate for LED systems. LED systems require a ratio balance that is dependent on each color component's wavelength. Furthermore, our system color calibration is also dependent on the relative current drivers in our LED board's hardware as each of these current drivers are controlled by DACs. We used a procedure based on the methods described by Hooi (2013).

Based on the dominant wavelengths of each color channel of our system's RGB LED, we first computed the CIE xy color coordinates, and then (using Hooi's procedure) we computed the D65 RGB mixing ratios. These values are summarized in Table 4.1. We used the D65 white balance color model as it is the primary standard for daylight. This ratio drives the relative contributions of each color towards a combined gray level.

To determine the relationship between a DAC code (a 16 bit value), the produced current, and the resultant intensity, we measured a sequence of increasing DAC codes for each color and used a light sensor[3]

[3]Our system used the Avago APDS-9250, produced by Broadcom: `https://www.broadcom.com/products/optical-sensors/ambient-light-photo-sensors/apds-9250`

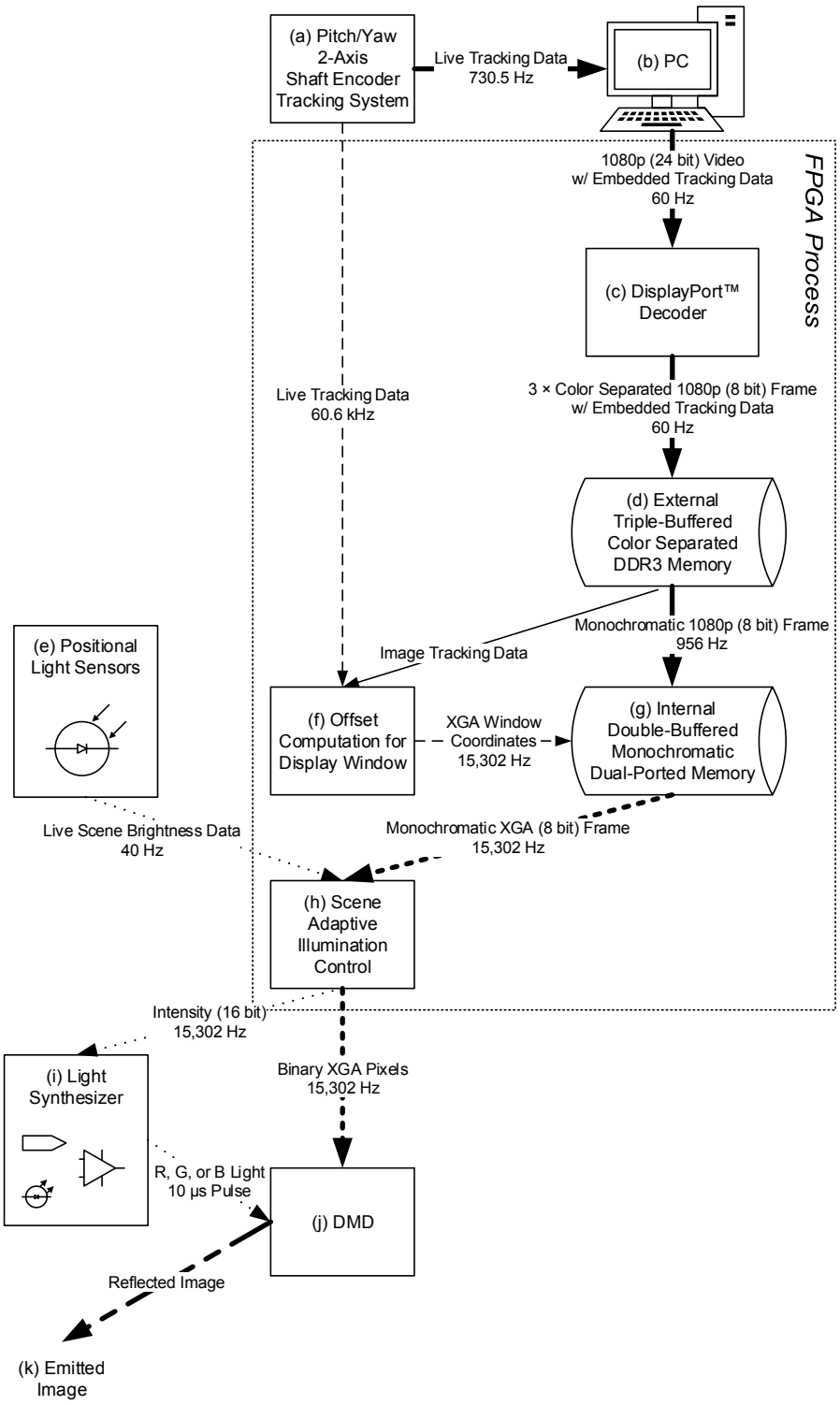


Figure 4.6: Implemented data path diagram for the 16 bit/color system. Dashed lines indicate the tracking data pipeline: (a), (f), (g), (h), (j), (k). Thick lines indicate the video content pipeline: (a), (b), (c), (d), (g), (h), (j), (k). Lastly, dotted lines indicate the illumination pipeline: (e), (h), (i), (j), (k). Segments of composite features (*e.g.*, thick dashed) indicate overlap (*e.g.*, video and tracking) in the pipelines.

Table 4.1: Summary of color mixing parameters for the documented dominant wavelengths of the 16 bit/color system's LEDs.

Color	*Wavelength (nm)*	*CIE Color*		*Color Mix (D65)*		
		x	*y*	*Computed*	*Normalized (Green)*	*Normalized (1)*
Blue	460	0.1440	0.0297	0.1145	0.0554	0.0373
Green	525	0.1142	0.8262	2.0658	1	0.6738
Red	625	0.7006	0.2993	0.8855	0.4286	0.2888

to measure each intensity. This light sensor provided an Ambient Light Sensing (ALS) value, which consists of an overall brightness value similar to human perceived brightness of color. To produce the intensities for capture, the system ran the normal modulation procedure with the following modifications: only one color channel was active at a time, the specified DAC code was used for all 16 bitplanes of that color channel, and the binary image consisted of all "on" values; all other colors were skipped. To speed up the data collection process, we collected a subset of all possible DAC codes, with smaller step sizes at the dim end and larger step sizes at the bright end. The sensor was shielded from ambient room lighting to measure the display's light path exclusively. A sample graph of collected calibration data is presented in Figure 4.7.

Lastly, based on the color mixing values and the collected data, we determined which color limited the maximum supported intensity and computed all of the bitplane's DAC codes. Each color channel's contribution to a possible maximum intensity is determined by the ratio of the largest measured ALS value for that color divided by the normalized color mix value (the last column in Table 4.1). In our system's case, red was the limiting color; as a result, its ratio was used for the target ALS combined value for the brightest bitplane, and its maximal DAC code was the maximal 16 bit value: 65 535. Each subsequent dimmer bitplane used a target ALS value that was half of the previous bitplane. Each DAC code was computed by linearly interpolating between the two nearest measured ALS-DAC code correspondences. A summary of the computed DAC codes for our system are listed in Table 4.2. Our color calibration produces 16 bit/color HDR brightness values with white balance. Despite the blue color channel requiring a target ALS value of zero for the dimmest bitplane, all DAC codes within a color channel are unique.

4.4.2 Results

To demonstrate that our system meets our objectives—HDR color display and low latency—we created test setups that allowed us to use standard cameras (*i.e.*, 8 bit/color) to capture what a user would see. We

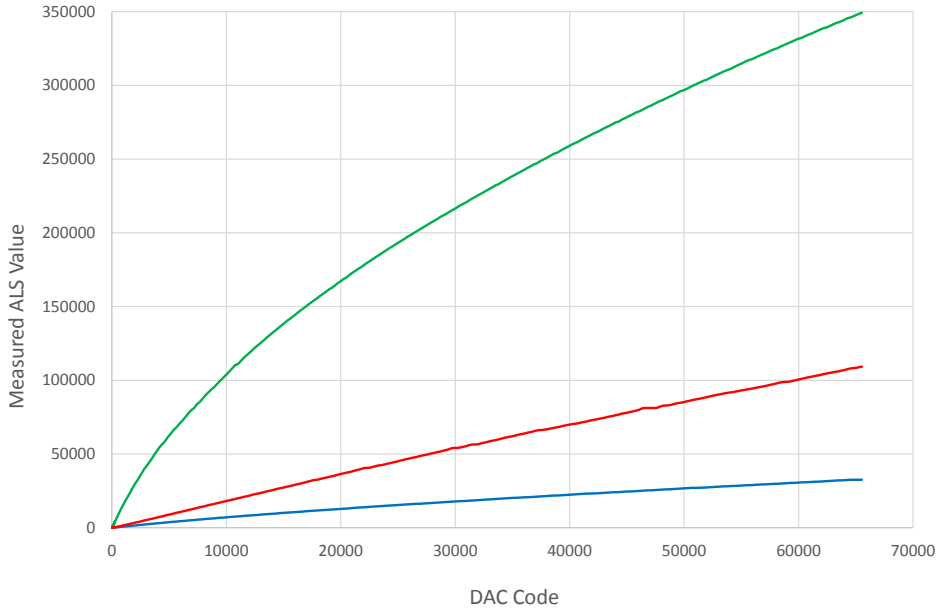


Figure 4.7: Per-color measured light sensor data for each 16 bit DAC code. Measured ALS values are raw 20 bit count values which map to a unit of power per area. The sensor varied in sensitivity to color (highest to lowest): green, red, and blue.

Table 4.2: Computed color calibrated 16 bit/color DAC codes using ALS data in Figure 4.7 and mixing values in Table 4.1.

Bitplane	*Target ALS Sum*	*Per-Color ALS*			*Target DAC Code*		
		Blue	*Green*	*Red*	*Blue*	*Green*	*Red*
15	378 243	14 123	254 870	109 250	22 555	39 024	65 535
14	189 122	7062	127 435	54 625	10 020	13 323	30 549
13	94 561	3531	63 718	27 313	4695	5133	15 029
12	47 280	1765	31 859	13 656	2298	2199	7589
11	23 640	883	15 929	6828	1171	1018	3891
10	11 820	441	7965	3414	623	499	2025
9	5910	221	3982	1707	344	258	1074
8	2955	110	1991	854	200	141	582
7	1478	55	996	427	123	84	323
6	739	28	498	213	78	51	189
5	369	14	249	107	53	37	115
4	185	7	124	53	36	25	74
3	92	3	62	27	25	22	49
2	46	2	31	13	17	18	35
1	23	1	16	7	9	16	26
0	12	0	8	3	8	13	20

Figure 4.8: Panorama of demonstration setup used with the 16 bit/color system.

Figure 4.9: A raw GPU screenshot used as input for testing DDS.

designed a physical environment, as shown in Figure 4.8, in which we could create areas of both bright light and dark shadow using a desk lamp (250 W-equivalent LED light bulb), two stand lights (each with two or three 100 W incandescent bulbs, two bright flashlights, and the ceiling florescent lights. Lighting conditions varied among demonstrations.

We augmented the user's view of a real desk (about 3 m from the user) with a static pair of colorful teapots, positioned so that they exist in scene areas with different lighting conditions. Figure 4.9 presents an example GPU screenshot of this virtual scene.

As a proxy for a user's eye, we used a Point Grey CM3-U3-13Y3C[4] (1280 × 1024, global shutter) camera with a Kowa LM6NCM[5] (1/2 in, 6 mm, F/1.2) C-mount lens. The camera was rigidly mounted to the HMD apparatus (*see* Figure 3.5, lower left). Filming with a standard camera would result in flicker (not perceived by human viewers) due to capturing variable numbers of Most Significant Bit (MSB) intensity binary frames in each camera frame. To avoid this we triggered the camera in phase with the current color synthesis sequence, and we restricted the shutter duration to an integer multiple of the synthesis sequence duration.

For this demonstration, we optimized the camera exposure for the shadow half of the scene by fully opening the aperture; an image of this is shown in Figure 4.10(a). The camera recorded video at 22.77 Hz. To show the maximal range of the display without losing detail, we disabled brightness adaptation by disabling the system's light sensors. We also locked the display's 16 bit brightness scaling parameters, setting these to utilize both the upper and lower 8 bit subranges in each half of the display; referring to Equation (4.6) in Section 4.5, we set the scale factor value of $l_{\text{sensed}}/l_{\max}$ to 1 for the left half and to $1/2^8$ for the right half. Since it is not normally possible to capture the full range of the display with a standard camera, we physically placed a neutral density filter in front of the camera so that it covered the bright (left) half of the scene and the whole image; these are shown in Figures 4.10(b) and 4.10(c) respectively. Note that teapots' details remain visible in both halves of the scene.

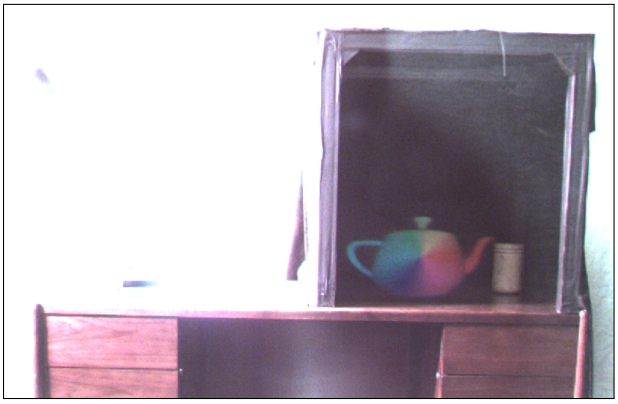
4.4.3 Limitations

As a result of a limited amount of fast, chip-internal SRAM, this implementation presents imagery in a color sequential manner. While a given output frame has an average motion-to-photon latency of 124 µs with regard to its position (*see* Section 3.3.5), changes pertaining to the appearance or color of the virtual object is much longer. For this RGB 16 bit/color implementation, the integration duration is 3.1 ms; although, it is still faster than that of the PR-PDM 6 bit/pixel version (*see* Section 4.3.2).

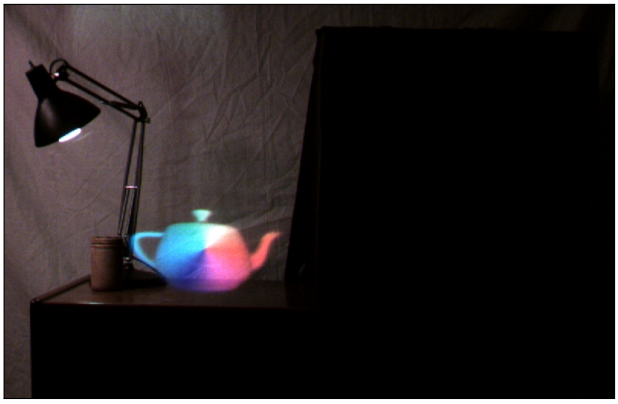
The current implementation is also limited to scaling 8 bit/color input to 16 bit/color output. Some changes would be required to support processing HDR input and increasing the output brightness resolution further to match the real world, though some of these problems may have relatively straightforward solutions (*see* Section 5.2.4).

[4] `https://www.ptgrey.com/chameleon3-13-mp-color-usb3-vision-on-semi-python1300-3`

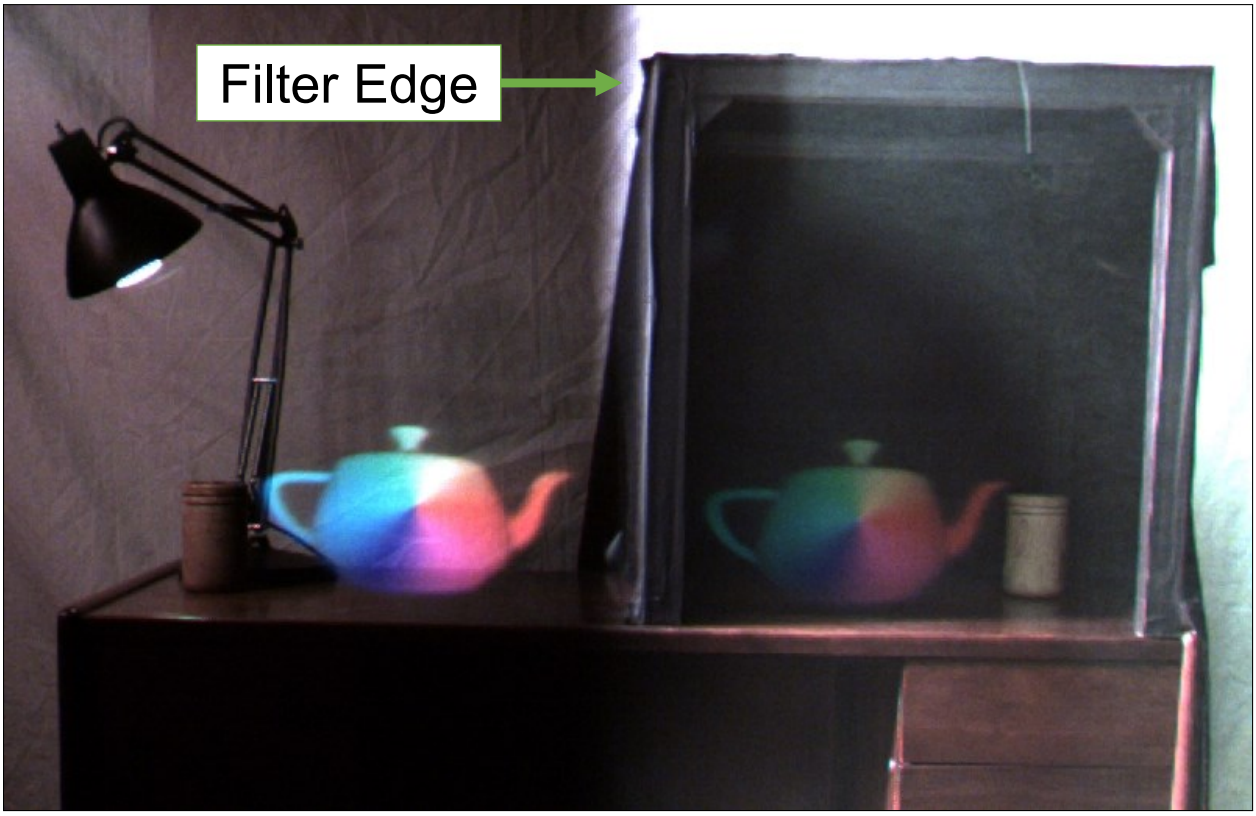
[5] `https://lenses.kowa-usa.com/wide-angle-megapixel-ncm-series/409-lm6ncm.html`



(a) Neutral density filter absent.



(b) Neutral density filter fully blocks camera lens.



(c) Neutral density filter blocks left-half only.

Figure 4.10: An HDR pair of colored virtual teapots (*see* Figure 4.9 for example input imagery) as recorded through the display. The teapot on the left is displayed 256 times brighter than the one on the right. To capture both simultaneously in a standard 8 bit/color camera, we optimized the exposure for the right/darker teapot (without filter). To demonstrate the HDR display output, a neutral density filter with 98 % attenuation is placed in front of the camera.

4.5 Scene-Aware Spatial Brightness Control

The PC in our system generates 8 bit/color imagery, without knowledge of the sensed light conditions. To display it for the physical HDR scene, we must scale it appropriately. If part of the physical scene, as viewed from the user's perspective, were as bright as the display maximally supports, then the output content needs to be as bright; if the scene is locally dim, then the output must match it there as well. If the display were too bright relative to the background, then it could dazzle (temporarily blind) the user in those regions; if the display were too dim, then user may not be able to discern the virtual imagery.

Supposing a single sensed ambient light value represented the entire visible scene, we could produce the 16 bit HDR desired value (d_{16}) using the 8 bit standard range value (d_8) and a scale factor based on the sensed and maximum supported light values (l_{sensed} and l_{max}); this is shown in Equation (4.6).

$$d_{16} = 2^8 \times \frac{l_{\text{sensed}}}{l_{\text{max}}} \times d_8 \tag{4.6}$$

For example, if the sensed brightness of the scene were as high as the display is capable of producing ($l_{\text{sensed}} = l_{\text{max}}$), then the 8 bit input data would occupy the eight MSBs of the 16 bit output. If, instead, the scene were half as bright as maximally supported, then the output data would be half of the maximally bright value.

In order to provide a more localized HDR match, we could replace l_{sensed} with a viewpoint (image-space) representation of the sensed light, $l_{\text{sensed}}(x, y)$, for each pixel. This requires measuring the lighting conditions of the physical scene with at least as much range as the display's output. We were unable to obtain a camera with the necessary range of sensitivity, so we used live measurements from a horizontal array of four light sensors[6], which is pictured at the top right of Figure 3.5 and diagrammed as Figure 4.6(e). The physical placement of the sensor array behind a baffle restricted each to seeing approximately one-quarter of the display's horizontal field-of-view. We currently use a piece-wise linear function (using 5 segments) and a exponential-decay averaging operation to interpolate sensor measurements horizontally in space and to smooth them in time. Each sensor updates its measurement at 40 Hz. In order to provide smooth transitions in time, we perform interpolation updates much faster (1514 Hz, limited by the FPGA's Central Processing Unit [CPU]). While small measured differences in brightness between physical regions are not perceptually

[6]Our system used the Avago APDS-9250, produced by Broadcom: `https://www.broadcom.com/products/optical-sensors/ambient-light-photo-sensors/apds-9250`

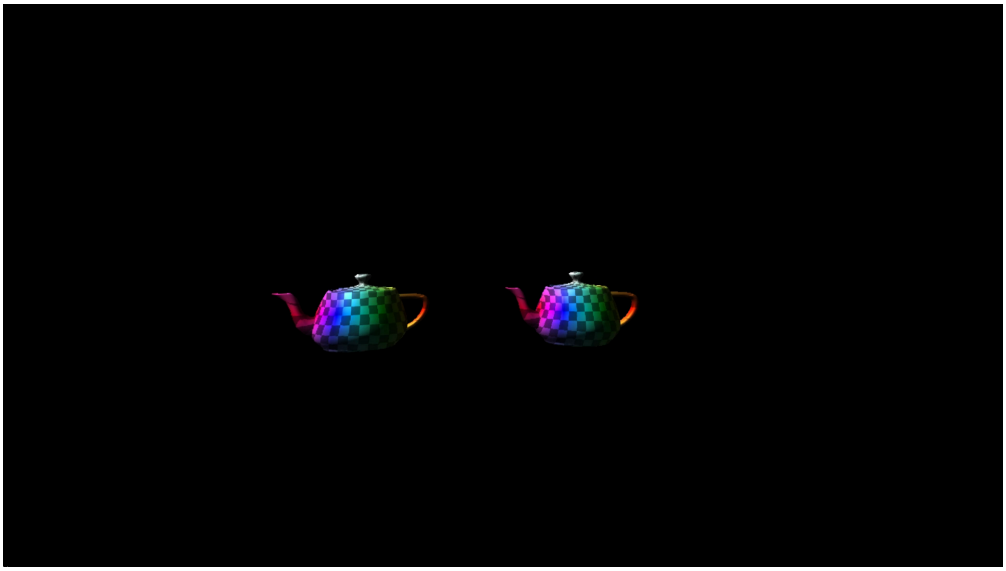


Figure 4.11: A raw GPU screenshot used as input to the illumination algorithm.

disturbing, significant variation in brightness occasionally produces human-perceptible discontinuities in the output along the sensor boundaries. In general, these issues were minor.

4.5.1 Results

We used the same physical setup as presented in Section 4.4.2 and Figure 4.8, but instead augmented the user's view with a pair of colorful 3D, rotating, Phong-shaded (Phong, 1975) teapots (*see* Figure 4.11), positioned so that they exist in scene areas with different lighting conditions. The camera recorded video at 45.45 Hz with a locked shutter duration. Note that the GPU lighting model is static: the GPU rendering is not modified by the real-time lighting measurements.

To demonstrate dynamic brightness under changing lighting conditions, we varied the lighting across the physical scene while recording. Our system's light sensors detect the changing lighting conditions, and the system responds by modifying the displayed brightness of the virtual scene elements. For recording, the user's proxy camera's exposure was balanced to minimize oversaturating during the brightest observed condition in the demonstration. Since the recording camera did not support HDR, when illumination is very low, the teapots can dim to invisibility.

We used two different lighting scenarios to test the system: we moved a pair of bright flashlight beams around the scene (*see* Figure 4.12), and we manipulated the head of a bright desk lamp (*see* Figure 4.13). During the first condition, we also moved the display left and right along the yaw axis; though during the

second condition, the display remained static in pose. Due to the limited sensitivity of the light sensors, we needed to apply an additional scale factor on the incoming measurements in order to produce a commensurate response to the lighting conditions. For example, in the flashlight condition, the values detected by the 16 bit light sensors, with an analog gain of 18 (the supported maximum), ranged between 0 and 4. In order to produce human detectable changes in brightness, we applied an additional digital gain factor of 140 to the reported measurements.

With these examples and scale factors, we observed that the teapots rapidly and detectably changed in brightness. In the flashlight case, for example, the teapot nearest to the dimmer flashlight would be dimmer than the the teapot nearest to the brighter flashlight, and either teapot would disappear in the absence of a nearby light. Similarly, in the desk lamp case, the left teapot (closer to the lamp) was always brighter than the right one. Additionally, as the desk lamp was much brighter than the flashlights, and it cast more light onto the background, which could reflect onto the light sensors; these teapots were brighter than the flashlight case. The desk lamp example illustrated the greatest range of these tests, as the brightness of the teapots ranged between nearly invisible and oversaturated when viewed by the 8 bit camera; however, these examples did not fully use the entire 16 bit/color range and were not intended to capture full use of the range.

4.5.2 Limitations

The primary limitations of the current implementation relate to the light sensors' sensitivity and the manner in which their data is used.

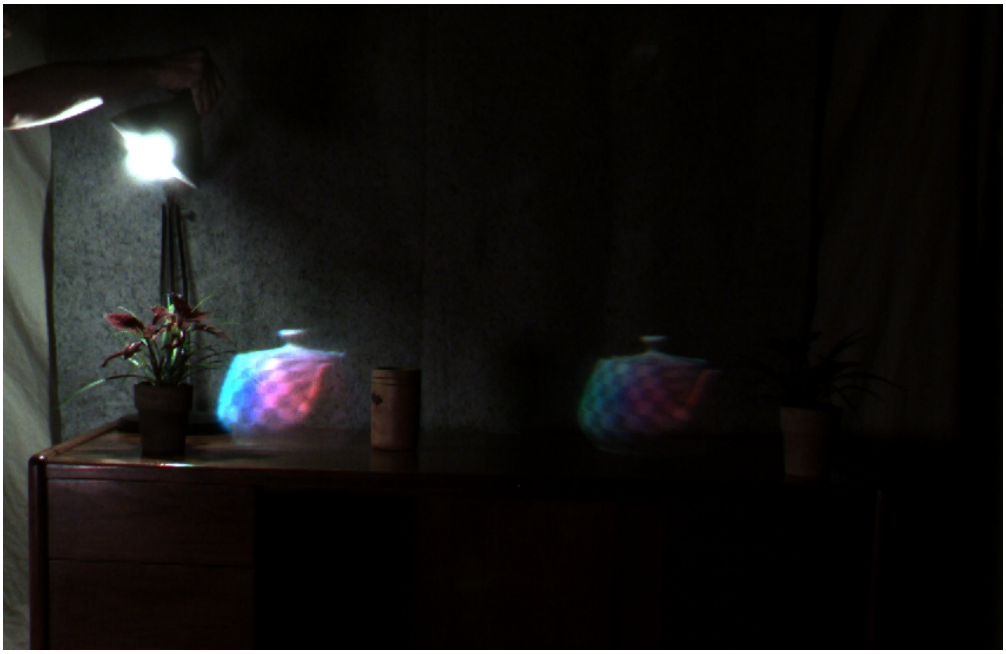
In the lab conditions, with maximal analog gain, we were unable to saturate the reported values. In fact, shining our display onto the sensors at close range (less than a centimeter from the lens) during calibration did not result in saturation. As a result, our sensitivity to room lighting conditions was very limited; the 16 bit sensors reported values less than a couple hundred. In order to produce visible output based on comparing with the calibration data required applying both analog and digital gain factors. This digital gain factor currently requires manual tuning based on the lighting condition and recording camera settings. When a human user directly observes the display instead, the sensitivity of the gain factor's value is lessened. Using different optics, which could concentrate the frustum's light onto the sensor, or using different sensors with higher sensitivity might have improved the situation but were unavailable at the time of testing.

The limited spatial resolution of the light sensor array occasionally produced artifacts in the output. As pictured in Figure 3.5, a baffle ensures that, up to a specified distance (3 m), each light sensor observes a

(a) The right teapot (closer to the brighter flashlight) appears brighter than its left counterpart.

(b) The left teapot (closer to the sole dimmer flashlight) appears brighter than its right counterpart. The right teapot is nearly invisible.

Figure 4.12: The augmented scene illuminated by bright flashlights (*see* Figure 4.11 for example input imagery). The camera exposure is optimized for the brightest possible condition without over saturating. Note the variation in brightness of the teapots among placement and lighting condition.



(a) Desk lamp tilted left. Note that the left teapot (closer to the lamp) appears significantly brighter than its more distant counterpart.

(b) Desk lamp tilted right. Note that the left teapot (closer to the lamp) is slightly brighter than its more distant counterpart.

Figure 4.13: The augmented scene illuminated by a desk lamp (*see* Figure 4.11 for example input imagery). The camera exposure is optimized for the brightest possible condition without over saturating. Note the variation in brightness of the teapots among placement and lighting condition.

different and display-matched frustum of the world. For a given sensor, only light emitted or reflecting within that frustum will be detected. However, in cases where there is very large variation in average brightness detected between neighboring sensors, the spatial interpolation of brightness scaling between those regions produces a linear ramp between two corners. When this ramp is sufficiently steep, is due to a hard edge in brightness in the real-world, and crosses through a virtual object, the effect is very noticeable. Higher sensor spatial resolution (especially if matched to the display's spatial resolution) could resolve the location of the edge, but it remains to be seen if a hard edge in the virtual object would be better. Section 5.2.5 discusses other possible extensions that could resolve these issues.

4.6 Summary

A comparison of the requirements and resultant behavior of the aforementioned modulation schemes are presented in Tables 4.3 and 4.4, respectively. Of all the algorithms presented and considered, PR-PDM and DDS require the least amount of FPGA memory resources, and DDS requires the least FPGA computation resources. DDS is also the only algorithm suitable for HDR operation with modulated displays as it requires the lowest order of steps presented: linear. A visual comparison illustrating simulated behavior of these modulation schemes is aggregated in Figure 4.14.

Table 4.3: Comparison of modulation technique requirements.

Method	*Requirements per Binary Pixel*						
	Memory Operations[a]			*Math Operations*[b]			
	Desired	*Accumulation*	*History*	*Comparator*	*Add/Sub.*	*Counter*	*LFSR*
PWM	Read			Yes		Yes	
EEM	Read	Read/Write	Read/Write	Yes	Yes, 2		
PDM (Delta-Sigma)	Read	Read/Write		Yes	Yes, 2		
PR-PDM	Read			Yes			Yes
DDS	Read					Yes	

[a] Fewer is better (higher display frequency)

[b] Fewer is better (lower FPGA logic requirements)

Table 4.4: Comparison of modulation technique behavior.

Method	*Major Oscillation Frequency*[c]	*Response to Input*[d]	*Modulation Steps*[e]
PWM	Low	Fast	Exponential
EEM	Low	Slow	Exponential
PDM (Delta-Sigma)	High	Fast	Exponential
PR-PDM	High	Fast	Exponential
DDS	High	Fast	Linear

[c] Higher is better (less perceptually noticeable)

[d] Faster is better (less latency)

[e] Slower to increase is better (shorter integration interval)

Figure 4.14: Sample comparison of simulated 4 bit grayscale modulation schemes by method and desired intensity level. Each chart shows the output intensity (vertical axis) associated with 48 consecutive binary frames (horizontal axis). Output intensities are normalized so that each scheme produces the same aggregate intensity. The value d indicates the desired integrated intensity in the range $[0, 15]$. Color indicates that the pixel is on with the illuminator using the specified intensity; white indicates that the pixel is off. The numbers on the horizontal axes indicate the first step of each integration cycle. Note that for PWM (blue), Delta-Sigma (red), and PR-PDM (green) modulations, the graphs show 4 complete integration cycles (each of 15 steps), while for DDS (purple), 15 complete integration cycles are present (each of 4 steps).

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 Summary

This dissertation was motivated by the problems with registration and appearance matching in AR. More specifically it was motivated to do the following:

1. to improve temporal registration, which breaks down most often as a result of latency;
2. to provide low latency HDR, in an effort to mimic the real world;
3. and to provide scene aware brightness, in an effort to match the real world without blinding the user or hiding the virtual imagery.

The results of the presented work have demonstrated these abilities through proof of concept implementations.

Chapter 3 presented the render cascade pipeline: a series of steps, each faster and more frequent than the prior and each applying simpler PRWs, each of which reduce the effects of latency in the previous result. PRWs propose that as long as the intended operation is small enough, then errors resulting from the change made are less disturbing than not attempting to fix the latency. Furthermore, in order to reduce the latency following the end of the cascade, the final step(s) of PRW should be performed in the display hardware, thus making it also a post-transmission warp (referring to the connection between GPUs and displays). Essentially the overall rendering algorithm is split between the GPU and the display processor. The chapter also presented a subset implementation of a proposed render cascade pipeline in FPGA hardware on a DMD, which supported an average motion-to-photon latency of about 80 µs or 124 µs, depending on the modulation method used. When compared with spatial and temporal thresholds for head turning latency detection (*see* Section 2.2.3) for VR systems, our AR system either meets those requirements or provides support at the faster end of threshold range.

Table 5.1: Comparison of implementation performance. All frequencies are in Hz.

Implementation	*Virtex®-5 XGA Freq.*		*Virtex®-7 XGA Freq.*	
	Binary	*Integration*	*Binary*	*Integration*
EEM: 6 bit/pixel	964	15.3		
PR-PDM: 6 bit/pixel	4340	68.9	15 552	246.9
DDS: 16 bit/color (RGB)			15 302	318.8

Chapter 4 presented several implementations of modulation methods, most notably PR-PDM and DDS (and one less effective one: EEM). A summary of the performance of these methods on prototype hardware is listed in Table 5.1. Both PR-PDM and DDS require little computation hardware to execute, DDS even less so. These lightweight modulation schemes enabled taking advantage of the aforementioned in-display PRW. However, DDS provides the greatest benefit by being linear-time with regard to the color bit depth of the imagery; this enabled generating HDR output, in an effort to both mimic and match the real world brightness with virtual imagery in real time, with low latency.

5.2 Future Work

This work has presented a system with very low display latency, and certain steps were taken to simplify the situation. A more general use case would introduce many additional challenges. Despite these challenges, the basic design of our display system would continue to benefit from these potential additions.

5.2.1 Inertia

In our development system, we use several commercial development kits (namely the TI DLP® Discovery™ 4100 Kit for projection and the HTG-777 FPGA board for processing) and some custom components (a custom interconnect board, conversion optics, and a light synthesis board). The use of these kits adds both constraints and bulk to the design; mostly these resulted from a lack of high-speed serial links on the DMD chip and controller board used, requiring many very wide parallel links instead, which needed large inflexible cables. This resulted in a heavy head-mounted display (more than 3 kg).

However, if one were to design a custom DMD-based HMD using our technique from scratch, then several options to reduce the weight (and also the inertia) would be available. TI is already designing new, small, low-power DMDs for head-worn displays (Bhakta et al., 2014). Using a smaller light source and

replacing the lens and prism or beam-splitter with a waveguide for optics (like most commercial OST-AR systems) would also be less bulky. These and specialized (rather than FPGA) control chips with high-speed serial links (at least 13 Gbit/s for the current resolution and binary frame rate) would reduce the amount of hardware that needs to go on the head, and move the display processing logic into a backpack or off-person location. Additionally, for mass production, the processing hardware could be optimized further for minimal size and power or produced as Application Specific Integrated Circuits (ASICs) or SoCs rather than general-purpose (reprogrammable) FPGAs.

5.2.2 Latency and General Pose Tracking

Tracking accuracy and latency from more general purpose trackers represent a more significant problem for OST-AR displays. Our technique is an open-loop process and only performs dynamic mitigation due to latency from initial rendering to scanout to presentation. Errors due to miscalibration or late reports directly from the tracking system to the display processor would both degrade the experience by misaligning the virtual imagery or causing it to lag behind the physical world. However, in the presence of a tracking system which reports frequently, though with significant delay, we could continue to reduce the perceived latency to the sum of that tracking latency and about 80 µs or 124 µs of display latency, which would be less than a standard display system where the latency also includes CPU and GPU processing, buffering, and transmission and display buffering, processing, and outputting. Tracking systems with low update rates, such as those using normal cameras, would reduce the effectiveness of the display, as the virtual imagery only moves with new tracker poses. AR systems using tracking systems with low latency and high report rates would gain the most benefit from our display processing system.

5.2.3 Render Cascade Pipeline

In the prototype systems, we have demonstrated a modified rendering pipeline that splits the rendering effort between the standard PC and the display hardware; some graphics transformations occur post-transmission. We explored a simple 2D image-space translation operation since our tracking system was limited to two rotational axes. Our splitting operation worked as we could provide tracking data to both the GPU and display processor.

Section 3.2 proposes that an extended pipeline with more intermediate transformations could further improve the situation. The improvement is predicated on the assumptions of PRWs: as long as the change

is small, then the latency compensation method should realign the imagery with less noticeable artifacts than not performing the operation. Assuming that a sufficiently high-frequency, low latency, unrestricted, 6-DOF (x, y, z, roll, pitch, yaw) tracking system existed, then one could explore these options. Operations like 2D translation are relatively simple as they maintain a 1-to-1 mapping between input and output pixels. Performing more general warps in the display—including rotation, skew, perspective, homography, or distortion—would significantly complicate the processing. If performed without interpolation, a skew operation could also maintain a 1-to-1 mapping, and a small 2D rotation can be approximated by one or two skew operations. Small perspective changes or other 3D transforms of multiple objects could be approximated by compositing a set of 1-to-1 mappable operations. However, many of the other listed operations will break the simple 1-to-1 mapping; implementing these on FPGA logic essentially reimplements a limited GPU. Xilinx provides a sample library for performing radial distortion correction (Xilinx, 2015), and third parties provide libraries for GPU-like modules[1]; however, each of these samples are limited to one pixel/clock cycle. We perform the translation on display since the GPU could not provide pixel data at the necessary rate. When considering extending the render cascade pipeline, selecting the transition point between GPU and FPGA/ASIC logic becomes a balance between processing power and transmission bandwidth.

It remains to be seen, however, for a general low latency tracking system, how many transforms, in what order, and where to apply them will be necessary. Once the technology is available, one could perform perceptual user studies to explore it. The operations might be determined on a case-by-case basis with no "one size fits all" solution.

5.2.4 High(-er) Dynamic Range

As configured and calibrated, our system was able to display virtual objects in a background of a maximum of about 800 lx to 1000 lx while remaining visible. Given the sample scene in Figure 1.2, which featured a window to a sunny exterior, we would only have been able to keep virtual objects visible away from the window. In order to fully support details in all parts of that scene or the outside world, we would need both a brighter LED illuminator and more than 16 bit/color resolution.

In the prototype HDR system, the illuminating LEDs are turned on for a constant duration for each pulse, currently 10 µs for each of the 48 binary frames required to generate a complete color image. This timing

[1]For example, `https://www.xilinx.com/products/intellectual-property/1-8dyf-2248.html` or `https://www.xilinx.com/products/intellectual-property/1-1p022m.html`

was chosen because 10 µs is the maximum time during which the micromirrors are guaranteed to be in a stable state between frames, and so this is the brightest output we can produce. The illuminator module itself, however, is capable of turning on and off in under 1 µs. With suitable control signals to the LEDs, we could turn on the illuminator for shorter iterations and produce a higher bit depth for darker intensities. Given the rise and fall times of our current LED (less than 620 ns), we could extend the color depth to up to 20 bit/color resolution without changing the 16 bit DACs. Additional color resolution would require also changing the resolution of the DAC.

5.2.5 Scene Brightness Detection

Ambient scene brightness is currently measured with an array of four sensors, each covering a horizontal quarter of the scene. We use a relatively unsophisticated interpolation function to smooth measured brightness at the boundaries between sensors and we have occasionally observed artifacts occurring along the edges between the horizontal area assigned to the different sensors. There is work to be done, including possible user studies, to identify an appropriate, artifact free blending algorithm for very low spatial resolution sensors of this kind. As noted in Section 4.5.2, a straightforward hardware improvement is to build a higher resolution (*e.g.*, 2D) array of light sensors and use sensors with higher sampling rates.

Additional improvements could improve the alignment of the sensor array's view with the display's view. Using a beam splitter to place the light sensor module in the optical path would allow it to work at any distance from the scene, removing a distance limitation of our current setup. A common position for the sensor would be below the existing splitter; in this way the user continues to see the virtual image and real world, but the sensor sees the real world and the virtual image, the latter of which would influence the measured values. However, the light emitted by the display occurs as short pulses, currently with a duty cycle of about 15 %. If the exposure of the light sensors were modified to aggregate only the other 85 % of the time, then it would effectively ignore the display's emitted brightness and only observe the physical world.

An alternative to an optically aligned light sensor array would be a second optically aligned DMD. Instead of placing an LED at the typical location for illuminating a DMD, one could place a single light sensor. By carefully synchronizing the sensor-DMD's pixels with the emitter-DMD or by using a very fast light sensor (where samples during light emission could be ignored), this light sensor could collect aggregate brightness over the regions selected by the sensor-DMD's pixels. In this way, the second DMD becomes a synchronized single-pixel camera. If the sensor-DMD's pixels were matched to the filled shape of a virtual

object, then the collected light would directly represent the aggregate brightness behind that object. To handle larger objects, multiple objects, or structure within objects, however, a different pattern would be needed. Compressive sampling techniques (Duarte et al., 2008; Koppelhuber and Bimber, 2017) could help construct a full resolution brightness map with this single-pixel camera at high rates.

Lastly, if we were able to capture full scene illumination parameters (*e.g.*, a light field or an enumeration of all the physical light sources and their properties), we could further improve the appearance of the augmenting elements by relighting them to match the lighting characteristics of the real scene.

5.2.6 Video Latency

One component of the overall video latency is the result of having to use external DRAM for the triple buffering of the three streams of single color images (*see* Figure 4.6(d)). The current 16 bit/color system's Virtex®-7 FPGA has 54.2 Mbit of SRAM and we use most of that memory to store two 2048 $\times$ 1080 8 bit/pixel color planes (about 17.7 Mbit each). Each 8 bit/color RGB image at 2048 $\times$ 1080 resolution requires about 53.1 Mbit. A future implementation of the display controller could use a custom ASIC or larger FPGA, which could include enough SRAM to eliminate the time penalty of reading and writing to slower off-chip DRAM. For example, the Virtex® UltraScale™ XCVU190 FPGA[2] provides about 139.3 Mbit of SRAM.

Alternatively, the availability of very fast external memory could resolve this issue. Our 16 bit/color system requires about 2.99 Gbit/s of write performance (8 bit/color RGB images at 1080p resolution at 60 Hz) simultaneously with 96.3 Gbit/s read performance (8 bit/pixel color planes at XGA resolution at 15 302 Hz).

5.2.7 Human Perception Studies

The demonstrated prototype presents a system with particular latency characteristics; however, it is an open question on how much of this effort is needed. Analytically, one could determine numerically that given a particular motion profile and display specification that a particular latency will result in a one-pixel offset error, but is that error noticeable? Prior studies on human perception in virtual environments have often dealt with systems with much greater latencies than our prototype system or used carefully controlled motion to simulate particular latencies (*e.g.*, Jerald, 2009). Other studies have use mechanical setups linked to the

[2] `https://www.xilinx.com/products/silicon-devices/fpga/virtex-ultrascale.html`

user's change perspective to simulate proportionally different motions (*e.g.*, Wallach, 1987). Nonetheless, a common theme to these studies is the use of "just noticeable differences;" as long as the system is capable of representing reality or achieving a golden baseline, one can query users if another option is perceptually different. With this type of method, the perceptual threshold can be determined for a given situation.

With modifications to the current system, one could dynamically modify the operational latency of the system by delaying tracking data in the display hardware. However, varying this parameter alone may assume that the current system is sufficient. Furthermore, the physical setup of the system and environment may introduce complicating factors (*e.g.*, inertia, brightness, color, spatial structure, contrast, size, motion path, periodicity). Future users of this type of system must also take care when designing the study to provide an appropriate baseline for comparison, which may be plain reality itself or a Spatial Augmented Reality (SAR) (simulated HMD) solution.

5.2.8 Alternative Spatial Light Modulators

While the techniques presented have been applied to DMDs, some may be applicable to other SLMs as well. Organic Light-Emitting Diode (OLED) displays are also configurable in a binary mode. Greer III (2015) presented a system for low latency display using a binary mode OLED display driven at 1700 Hz using a technique combining PDM and PWM. PR-PDM could also be applicable to this type of display as well, though with 6 bit/pixel grayscale resolution, integration rate would be about 27 Hz, which may be perceptually visible. If faster OLEDs become available, on the order of DMDs, then the presented techniques would have greater use.

5.3 Conclusions

I initially started working on the Low Latency Display project from a SAR perspective. I had worked on a multi-projector blending SAR system (Lincoln et al., 2011), which involved using multiple projectors, spaced around a white physical object, to provide a new virtual appearance. The system tracked the object and updated the imagery to display on the projectors accordingly. Unfortunately, at the time, the latency in the PC, GPU, and projectors meant that whenever the user moved the object at a decent speed, the imagery would lag. Even worse, because multiple projectors were involved, the imagery would break apart and rejoin rather than simply diverging. When I heard about what Feng Zheng was working on, which was later published as Zheng

et al. (2014), I suspected it would work for SAR. While I instead chose to refocus on OST-AR displays, I still suspect that it could.

The work presented here was focused on DMDs, though it should still be appropriate for other binary SLM-based displays. Furthermore, while I focused on OST-AR displays, VR displays also benefit from low latency and HDR. VR has less stringent requirements for latency than OST-AR, so the performance presented here should be quite sufficient.

As noted in Section 5.2, there is still substantial work to do before a full system based on this work can be deployed. The biggest complications relate to the weight of the equipment and the lack of a low latency, high precision 6-DOF tracking system. Furthermore, this prototype system focused on latency and brightness compensation; as noted previously, these alone may be insufficient. Nonetheless, things can get especially exciting in the future with more study and development. The benefits have been shown; now the gauntlet has been thrown down for solving the next challenges with producing compelling AR systems.

BIBLIOGRAPHY

Antonov, M. (2015). Asynchronous timewarp examined. `https://developer3.oculus.com/blog/asynchronous-timewarp-examined/`.

Aziz, P., Sorensen, H., and van der Spiegel, J. (1996). An overview of sigma-delta converters. *Signal Processing Magazine, IEEE*, 13(1):61–84.

Azuma, R. T. (1995). *Predictive Tracking for Augmented Reality*. PhD thesis, University of North Carolina at Chapel Hill.

Bailey, R. E., Arthur III, J. J., and Williams, S. P. (2004). Latency requirements for head-worn display S/EVS applications. In *Proc. SPIE*, volume 5424, pages 98–109.

Bhakta, V. R., Richuso, J., and Jain, A. (2014). DLP® technology for near eye display. White Paper DLPA051, Texas Instruments.

Bishop, G., Fuchs, H., McMillan, L., and Zagier, E. J. S. (1994). Frameless rendering: Double buffering considered harmful. In *Proc. of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 175–176, New York, NY, USA. ACM.

Borer, T. and Cotton, A. (2015). A "display independent" high dynamic range television system. White Paper WHP 309, BBC.

Bradski, G. (2000). *Dr. Dobb's Journal of Software Tools*.

Bresenham, J. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30.

Brooks, F. P. (1999). What's real about virtual reality? *IEEE Computer Graphics and Applications*, 19(6):16–27.

Buker, T. J., Vincenzi, D. A., and Deaton, J. E. (2012). The effect of apparent latency on simulator sickness while using a see-through helmet-mounted display: Reducing apparent latency with predictive compensation. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 54(2):235–249.

Daly, S., Kunkel, T., Sun, X., Farrell, S., and Crum, P. (2013). Preference limits of the visual dynamic range for ultra high quality and aesthetic conveyance. In *Proc. SPIE*, volume 8651, pages 86510J–86510J–11.

Dayal, A., Woolley, C., Watson, B., and Luebke, D. (2005). Adaptive frameless rendering. In *Proc. Eurographics Conference on Rendering Techniques*, pages 265–275.

Didier, J., Roussel, D., and Mallem, M. (2005). A time delay compensation method improving registration for augmented reality. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3384–3389.

Dolby (2016). Dolby Vision™ for the home. White paper, Dolby.

Duarte, M. F., Davenport, M. A., Takbar, D., Laska, J. N., Sun, T., Kelly, K. F., and Baraniuk, R. G. (2008). Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91.

Friston, S., Steed, A., Tilbury, S., and Gaydadjiev, G. (2016). Construction and evaluation of an ultra low latency frameless renderer for VR. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1377–1386.

Greer, T., Spjut, J., Luebke, D., and Whitted, T. (2016). 8-3: Hybrid modulation for near zero display latency. *SID Symposium Digest of Technical Papers*, 47(1):76–78.

Greer III, T. H. (2015). Low-latency display. US Patent App. 14/940,067.

Holloway, R. L. (1995). *Registration Errors in Augmented Reality Systems*. PhD thesis, University of North Carolina at Chapel Hill.

Holloway, R. L. (1997). Registration error analysis for augmented reality. *Presence*, 6(4):413–432.

Hooi, L. B. (2013). Understand RGB LED mixing ratios to realize optimal color in signs and displays. *LEDs Magazine*, 10(4).

Hornbeck, L. J. (1997). Digital light processing™ for high-brightness, high-resolution applications. In *Proc. SPIE, Projection Displays III*, volume 3013, pages 27–40.

Itoh, Y., Orlosky, J., Huber, M., Kiyokawa, K., and Klinker, G. (2016). OST Rift: Temporally consistent augmented reality with a consumer optical see-through head-mounted display. In *2016 IEEE Virtual Reality (VR)*, pages 189–190.

Jang, J. H., Kwon, M., Tjandranegara, E., Lee, K., and Jung, B. (2009). A digital driving technique for an 8b QVGA AMOLED display using $\Delta\Sigma$ modulation. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 270–271,271a.

Jerald, J. and Whitton, M. (2009). Relating scene-motion thresholds to latency thresholds for head-mounted displays. In *Proc. IEEE Virtual Reality*, pages 211–218.

Jerald, J. J. (2009). *Scene-Motion- and Latency-Perception Thresholds for Head-Mounted Displays*. PhD thesis, University of North Carolina at Chapel Hill.

Jones, A., Lang, M., Fyffe, G., Yu, X., Busch, J., McDowall, I., Bolas, M., and Debevec, P. (2009). Achieving eye contact in a one-to-many 3D video teleconferencing system. *ACM Trans. Graph.*

Kajiya, J. T. and Torborg, J. (1996). Talisman: Commodity realtime 3D graphics for the PC. Association for Computing Machinery, Inc.

Koeter, J. (1996). What's an LFSR. Technical Report SCTA036A, Texas Instruments.

Koppelhuber, A. and Bimber, O. (2017). Computational imaging, relighting and depth sensing using flexible thin-film sensors. *Opt. Express*, 25(3):2694–2702.

Lincoln, P., Blate, A., Singh, M., State, A., Whitton, M., Whitted, T., and Fuchs, H. (2017). Scene-adaptive high dynamic range display for low latency augmented reality. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, New York, NY, USA. ACM.

Lincoln, P., Blate, A., Singh, M., Whitted, T., State, A., Lastra, A., and Fuchs, H. (2016). From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1367–1376.

Lincoln, P., Welch, G., and Fuchs, H. (2011). Continual surface-based multi-projector blending for moving objects. In *2011 IEEE Virtual Reality Conference*, pages 115–118.

List, U. H. (1983). Nonlinear prediction of head movements for helmet-mounted displays. Technical Report AFHRL-TP-83-45, Air Force Human Resources Laboratory, Air Force Systems Command, Brooks Air Force Base, Texas, US. Final technical paper.

Livingston, M. and Ai, Z. (2008). The effect of registration error on tracking distant augmented objects. In *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 77–86.

Lynley, M. (2016). With 500m downloads, Pokémon Go is coming to the Apple Watch. `https://techcrunch.com/2016/09/07/pokemon-go-the-hottest-game-on-the-planet-is-coming-to-the-apple-watch/`.

Maimone, A. S. (2015). *Computational See-Through Near-Eye Displays*. PhD thesis, University of North Carolina at Chapel Hill.

Mann, S., Lo, R. C. H., Ovtcharov, K., Gu, S., Dai, D., Ngan, C., and Ai, T. (2012). Realtime HDR (high dynamic range) video for eyetap wearable computers, FPGA-based seeing aids, and glasseyes (EyeTaps). In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6.

Mark, W. R. (1999). *Post-Rendering 3D Image Warping: Visiblity, Reconstruction, and Performance for Depth-Image Warping*. PhD thesis, University of North Carolina at Chapel Hill.

Mark, W. R., McMillan, L., and Bishop, G. (1997). Post-rendering 3d warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 7–16, New York, NY, USA. ACM.

McDowall, I. and Bolas, M. (2005). Fast light for display, sensing and control applications. In *IEEE VR Workshop on Emerging Display Technologies*.

McMillan, L. (1997). *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, University of North Carolina at Chapel Hill.

McMillan, L. and Bishop, G. (1995). Head-tracked stereoscopic display using image warping. In *Proc. SPIE*, volume 2409, pages 21–30.

Microsoft (2016). Microsoft HoloLens | Official site. `https://www.microsoft.com/microsoft-hololens/`.

Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1995). Augmented reality: a class of displays on the reality-virtuality continuum. volume 2351, pages 282–292.

Mine, M. and Bishop, G. (1995). Just-in-time pixels. Technical report, Chapel Hill, NC, USA.

Oculus VR (2013). Blog — building a sensor for low latency virtual reality. `http://www.oculusvr.com/blog/building-a-sensor-for-low-latency-vr/`.

Pasman, W., van der Schaaf, A., Lagendijk, R., and Jansen, F. (1999). Low latency rendering for mobile augmented reality. In *Proceedings of the ASCI*, volume 99, pages 15–17.

Pausch, R., Crea, T., and Conway, M. (1992). A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. *Presence: Teleoperators and Virtual Environments*, 1(3):344–363.

Pavlovych, A. and Stuerzlinger, W. (2005). A high-dynamic range projection system. *Proc. SPIE*, 5969:59692O–59692O–8.

Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317.

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. (1998). The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proc. ACM SIGGRAPH*, pages 179–188.

Regan, M. and Pose, R. (1994). Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 155–162, New York, NY, USA. ACM.

SAE (2001). Transport category airplane head up display (HUD) systems. Technical Report ARP5228, SAE International.

Seetzen, H., Whitehead, L. A., and Ward, G. (2003). 54.2: A high dynamic range display using low and high resolution modulators. *SID Symposium Digest of Technical Papers*, 34(1):1450–1453.

Smit, F., van Liere, R., and Froehlich, B. (2010). A programmable display layer for virtual reality system architectures. *Visualization and Computer Graphics, IEEE Transactions on*, 16(1):28–42.

Smit, F. A., van Liere, R., and Fröhlich, B. (2008). An image-warping VR-architecture: Design, implementation and applications. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, pages 115–122, New York, NY, USA. ACM.

So, R. H.-Y. (1997). Lag compensation by image deflection and prediction: A review on the potential benefits to virtual training applications for manufacturing industry. *Advances in Human Factors/Ergonomics*, 21(B):997–1000.

Technicolor (2016). Technicolor HDR. `http://www.technicolor.com/en/solutions-services/technology/technology-licensing/hdr-technologies/technicolor-hdr`.

Texas Instruments (2013a). DLP® 0.7 XGA 2xLVDS Type A DMD. `http://www.ti.com/lit/ds/symlink/dlp7000.pdf`.

Texas Instruments (2013b). DLP® Discovery™ 4100 chipset data sheet. `http://www.ti.com/lit/er/dlpu008a/dlpu008a.pdf`.

UHD Alliance (2016). UHD Alliance defines premium home entertainment experience. `http://www.uhdalliance.org/uhd-alliance-press-releasejanuary-4-2016/`. Press Release.

Wallach, H. (1987). Perceiving a stable environment when one moves. *Annual review of psychology*, 38(1):1–29.

Walter, B., Drettakis, G., and Parker, S. (1999). Interactive rendering using the render cache. In *Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, volume 10, pages 235–246, New York, NY. Springer-Verlag/Wien.

Wetzstein, G., Heidrich, W., and Luebke, D. (2010). Optical Image Processing Using Light Modulation Displays. *Computer Graphics Forum*, 29:1934–1944.

Xilinx (2015). *Vivado Design Suite User Guide: High-Level Synthesis*. Xilinx, 2015.1 edition. UG902.

Yao, R., Heath, T., Davies, A., Forsyth, T., Mitchell, N., and Hoberman, P. (2014). Oculus VR best practices guide v 0.008. http://static.oculusvr.com/sdk-downloads/documents/OculusBestPractices.pdf.

Zheng, F., Whitted, T., Lastra, A., Lincoln, P., State, A., Maimone, A., and Fuchs, H. (2014). Minimizing latency for augmented reality displays: Frames considered harmful. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 195–200.