University of Huddersfield

A Thesis Submitted to the University of Huddersfield in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

An Interest Point Based Illumination Condition Matching Approach to Photometric Registration Within Augmented Reality Worlds

by

Mr Mark Bingham BSc (Hons)

April 27, 2011

Abstract

With recent and continued increases in computing power, and advances in the field of computer graphics, realistic augmented reality environments can now offer inexpensive and powerful solutions in a whole range of training, simulation and leisure applications. One key challenge to maintaining convincing augmentation, and therefore user immersion, is ensuring consistent illumination conditions between virtual and real environments, so that objects appear to be lit by the same light sources.

This research demonstrates how real world lighting conditions can be determined from the two-dimensional view of the user. Virtual objects can then be illuminated and virtual shadows cast using these conditions. This new technique uses pairs of interest points from real objects and the shadows that they cast, viewed from a binocular perspective, to determine the position of the illuminant. This research has been initially focused on single point light sources in order to show the potential of the technique and has investigated the relationships between the many parameters of the vision system. Optimal conditions have been discovered by mapping the results of experimentally varying parameters such as FoV, camera angle and pose, image resolution, aspect ratio and illuminant distance. The technique is able to provide increased robustness where greater resolution imagery is used. Under optimal conditions it is possible to derive the position of a real world light source with low average error.

An investigation of available literature has revealed that other techniques can be inflexible, slow, or disrupt scene realism. This technique is able to locate and track a moving illuminant within an unconstrained, dynamic world without the use of artificial calibration objects that would disrupt scene realism. The technique operates in real-time as the new algorithms are of low computational complexity. This allows high framerates to be maintained within augmented reality applications. Illuminant updates occur several times a second on an average to high end desktop computer.

Future work will investigate the automatic identification and selection of pairs of interest points and the exploration of global illuminant conditions. The latter will include an analysis of more complex scenes and the consideration of multiple and varied light sources.

List of Publications

1. M. Bingham, D. Taylor, D. Gledhill, Z. Xu, *"Integration of Real and Virtual Light Sources in Augmented Reality Worlds"*, 14th International Conference on Automation and Computing. Pacilantic International, ISBN 978-0955529320, pp 75-80. Brunel University, UK, September 2008.

2. M. Bingham, D. Taylor, D. Gledhill, Z. Xu, *"Integration of Real and Virtual Light Sources in Augmented Reality Worlds"*, Computing and Engineering Annual Researchers Conference 2008. ISBN 978-1-86218-067-3, pp 28-33. Huddersfield University, UK, November 2008.

3. M. Bingham, D. Taylor, D. Gledhill, Z. Xu, *"Illuminant Condition Matching in Augmented Reality: A Multi-Vision, Interest Point Based Approach"*, Sixth International Conference on Computer Graphics, Imaging and Visualization 2009. IEEE Computer Society. ISBN 978-0-7695-3789-4/09, pp 57-61. Tianjin University, China, August 2009.

4. M. Bingham, D. Taylor, D. Gledhill, Z. Xu, *"A Multi-view Interest Point Based Approach to Photometric Realism Within Augmented Reality Systems"*, Computing and Engineering Annual Researchers Conference 2009. ISBN 978-1-86218-085-7, pp 82-87. Huddersfield University, UK, December 2009.

5. M. Bingham, D. Taylor, Z. Xu, *"An Interest Point Based Illumination Condition Matching Approach to Photometric Registration Within Augmented Reality Worlds"*, Submitted to IEEE Transactions on Visualization and Computer Graphics.

List of Acronyms

Abbreviation	Description	Definition
VR	Virtual Reality	page 3
AR	Augmented Reality	page 4
MR	Mixed Reality	page 4
NPC	Non-Player Character	page 4
HMD	Head Mounted Display	page 4
BARS	Battlefield Augmented Reality System	page 4
SDK	Software Development Kit	page 13
SIFT	Scale Invariant Feature Transform	page 52
RAM	Random Access Memory	page 17
CPU	Central Processing Unit	page 17
MIPS	Million Instructions Per Second	page 17
DSP	Digital Signal Processing	page 18
HD	High Definition	page 18
BCCD	Bootable Cluster Compact Disk	page 20
BOINK	Berkeley Open Infrastructure for Network Computing	page 20
GPU	Graphics Processing Unit	page 20
GPGPU	General Purpose Graphics Processing Unit	page 20
LCD	Liquid Crystal Display	page 21
CRT	Cathode Ray Tube	page 21
LCOS	Liquid Crystal on Silicon	page 22
GPS	Global Positioning System	page 26
IMU	Inertial Measurement Unit	page 27
PIC	Programmable Interface Controller	page 27
USB	Universal Serial Bus	page 29
CCD	Charge-Coupled Device	page 30
KLT	Kanade-Lucas-Tomasi	page 39
API	Application Programmer Interface	page 13
FAST	Features From Accelerated Segment Test	page 52
SLAM	Systematic Localization and Mapping	page 39
LS	Least Squares	page 40

Abbreviation	Description	Definition
UKF	Unscented Kalman Filter	page 40
HDR	High Dynamic Range	page 46
SUSAN	Smallest Univalue Segment Assimilating Nucleus	page 52
CMOS	Complimentary Metal-Oxide Semiconductor	page 28
VDU	Visual Display Unit	page 28
MRT	Mixed Reality Toolkit	page 31
SLHT	Straight Line Hough Transform	page 54
UML	Unified Modeling Language	page 91
PCA	Principle Component Analysis	page 53
RGB	Red-Green-Blue	page 62
BGR	Blue-Green-Red	page 62
CL	Correspondence Line	page 67
DOF	Degrees of Freedom	page 40
FoV	Field of View	page 116

List of Figures

1.1 Head Mounted Display Scene[31] 4
1.2 Battlefield Augmented Reality System[45] 5
1.3 Tinmith Augmented Reality System[77] 6
1.4 Incorrect Photometric Registration 7
1.5 Unnatural Photometric Calibration Sphere [63] 8

2.1 Eye of Judgement Game Setup [43] 12
2.2 Eye of Judgement Game Augmentation[43] 12
2.3 AR Car-Finder iPhone Application[66] 13
2.4 AR Bar Guide iPhone Application[66] 14
2.5 AR AcrossAir iPhone Application[66] 15
2.6 AR Fire Fighter 360 iPhone Application[66] 15
2.7 Marker Based Augmentations 16
2.8 An Augmented Scene 17
2.9 AR Scene Lighting 18
2.10 Multi-Pass Rendering 18
2.11 AR Hardware[80] 19
2.12 Human Visual Fields 21
2.13 Classification of Stereoscopic Displays 32
2.14 GPS Navigation [20] 33
2.15 GPS Component [20] 33
2.16 AR Base Hardware Components 34
2.17 Interlace Problem 34
2.18 ARTag Magic Lens[4] 35
2.19 ARTag Magic Mirror[4] 35

3.1 SIFT Extracted Features [29] 41
3.2 The KLT Feature Tracker [38] 42
3.3 Object Reflectivity 43
3.4 Photometric Calibration Using Specular Highlights[107] 47

3.5 Calibration Spheres[22] 48
3.6 Canny Edge Detection 51

4.1 Synthetic Input Imagery 58
4.2 Augmented Reality Process 58
4.3 Technique Process 59
4.4 Suitable Camera View A 61
4.5 Suitable Camera View B 62
4.6 Noisy Input Data 64
4.7 Gaussian Smoothing / Noise Reduction 65
4.8 Input Canny Edge Detection 65
4.9 Performing SIFT on Input Using Matlab 66
4.10 Shadow and Object Segmentation and Classification 67
4.11 Interest Point Correspondence 68
4.12 Forward and Backward 2D Intersections 69
4.13 Camera Pose Estimation 70
4.14 Forward Perspective Projection Transform 71
4.15 Converting to Clip Coordinates 72
4.16 Point Projection: Top View 74
4.17 Point Projection: Side View 74
4.18 Backward Perspective Projection Transform 75
4.19 Reverse Projection Cam A & Cam B 75
4.20 Illuminant Rays 76
4.21 Closest Points of Approach Lines 78
4.22 Virtual Shadowing Using Real Illuminant 79
4.23 Correspondence Line Intersection 80
4.24 Interesting Image Features[61] 80
4.25 Tracking Pseudo-Illuminant 81

5.1 UML Class Diagram: Detection Framework 92
5.2 2D Illuminant Detection 97
5.3 Fully Registered Augmentation 98
5.4 Tracking Simulation 100
5.5 Tracking Configuration and Information 101
5.6 Directly Opposing Camera Problem 102
5.7 Parallel Line Problem 103
5.8 Same Image Problem 104
5.9 Same Line Problem 105

6.1 MathCAD Initial Results 108
6.2 Sample Input Data 1 108
6.3 Sample Input Data 2 109

6.4 Sample Input Data 3 110
6.5 Visual Input (45°Δ) 110
6.6 Visual Input (90°Δ) 111
6.7 Example output 113
6.8 Angle Difference vs Error (320x240) 114
6.9 Angle Difference vs Error (640x480) 115
6.10 Angle Difference vs Error (1024x768) 115
6.11 Angle Difference vs Error (2048x1536) 116
6.12 Angle Difference vs Error (6400x4800) 116
6.13 Angle Difference vs Error (1920x1080) 117
6.14 Angle Difference vs Error (7680x4800) 117
6.15 Angle Difference vs Error (Combined) 118
6.16 FoV vs Error (640x480) 119
6.17 FoV vs Error (7680x4800) 119
6.18 FoV CamA, FoV CamB vs Error (640x480) 120
6.19 FoV CamA, FoV CamB vs Error (7680x4800) 120
6.20 Angle Difference, FoV vs Error (640x480) 121
6.21 Angle Difference, FoV vs Error (7680x4800) 121
6.22 Variable Resolution Both Cameras (1.333 aspect) 122
6.23 Variable Resolution Both Cameras (1.777 aspect) 122
6.24 Variable Resolution Both Cameras (1.6 aspect) 123
6.25 Artificially Inducing Error 123
6.26 Induced IP Err. vs Overall Err.: Single Cam (640x480) 124
6.27 Induced IP Err. vs Overall Err.: Both Cam (640x480) 125
6.28 Illum. Dist., Ind. IP Err. X vs Overall Err.: Single Cam (640x480) . . 126
6.29 Illum. Dist., Ind. IP Err. X vs Overall Err.: Both Cam (640x480) . . 127
6.30 Illum. Dist., Ind. IP Err. X vs Overall Err.: Both Cam (7680x4800) . 127
6.31 Illum. Dist., Ind. IP Err. Y vs Overall Err.: Single Cam (640x480) . . 128
6.32 Illum. Dist., Ind. IP Err. Y vs Overall Err.: Single Cam (7680x4800) 128
6.33 Illum. Dist., Ind. IP Err. Y vs Overall Err.: Both Cam (640x480) . . 129
6.34 Illum. Dist., Ind. IP Err. Y vs Overall Err.: Both Cam (7680x4800) . 129
6.35 0° Angle Difference Visualization 131
6.36 90° Angle Difference Visualization 132
6.37 110° Angle Difference Visualization 132
6.38 160° Angle Difference Visualization 133
6.39 270° Angle Difference Visualization 133

B.1 OpenGL Lighting Model 153
B.2 Linear Interpolation of Vertex Properties Within a Polygon 155

C.1 Shadow Point Projection 158

List of Tables

2.1 Comparison of GPS Components 28
2.2 Comparison of Compass Components 29
2.3 Comparison of Gyro Components 29
2.4 Comparison of Accelerometer Components 29
2.5 Comparison of Camera Input Devices 30

3.1 Comparison of significant photometric registration techniques 49
3.2 Box Blur Convolution Kernel 50
3.3 Gaussian Convolution Kernel 50

4.1 Memory Requirements of Grayscale Frame 63
4.2 Memory Requirements of Colour Frame 63

6.1 Results: Variant Resolution (45°Δ) 111
6.2 Results: Variant Resolution (90°Δ) 112

C.1 Comparison of 3D Shadow Emulation Techniques 160

Contents

Abstract **i**

List of Publications **iii**

List of Acronyms **v**

List of Figures **vii**

List of Tables **x**

Contents **xi**

1 Introduction **3**

1.1 Motivation 3

1.2 Aims 8

1.3 Thesis Outline 9

2 Background **11**

2.1 Hardware 15

2.1.1 Processing 16

2.1.2 Displays 20

2.1.3 Sensors 26

2.1.4 Video Input Devices 29

2.2 AR SDKs 30

3 Related Work **37**

3.1 World Registration 37

3.1.1 World Alignment & Geometric Registration 38

3.1.2 Illuminant Detection & Photometric Registration 42

3.2 Image Processing 49

3.2.1 Image Preparation 49

3.2.2 Feature extraction 50
3.2.3 Correspondence detection 53
3.3 Image Segmentation 53
3.3.1 Shadow Segmentation 54
3.3.2 Object Segmentation 54
3.4 Summary 55

4 Illuminant Tracking Technique 57
4.1 Technique Overview 57
4.2 Photometric Registration 60
4.3 Error Mitigation Strategies 79

5 Implementation 83
5.1 Mathematical Model 83
5.2 Detection Framework 91
5.3 2D Detection Prototype 96
5.4 3D Detection Prototype 96
5.5 Tracking Simulation 99

6 Evaluation 107
6.1 Technique Verification 107
6.2 Optimal Configuration and Operating Conditions 113
6.3 Susceptibility to Pixel Error 123
6.4 Technique Performance 126

7 Conclusion and Future Work 135
7.1 Conclusion 135
7.2 Contributions 138
7.3 Future work 139

Bibliography 141

A Camera Calibration 151

B Lighting & Shading Calculation 153

C Further Illumination Integration 157

Copyright Notice 161

Acknowledgements

First and foremost I would like to thank my friends and family for all their support throughout this venture. I would like to thank the School of Computing and Engineering at the University of Huddersfield for the opportunity to undertake this research. I would like to thank Prof. D. Taylor for guiding me in his role as Director of Studies and also Dr. Z. Xu as secondary supervisor. I would also like to thank the staff within the department that have helped and supported me during my time at the University as both an undergraduate student and postgraduate researcher.

Chapter 1

Introduction

1.1 Motivation

Virtual reality (VR) is the computer graphics concept in which a user is able to interact with a virtual, or computer generated environment. Artificial environments may be displayed in a number of ways including via computer or projector screen, or specialist head-worn visual display equipment. Stereoscopic displays that have two independent views, one for each eye, can be used to further enhance the virtual world. Such devices provide a 3D experience. They take advantage of the way in which our brains interpret the data from each eye in order to perceive the world around us. By showing each eye the same world from slightly different angles the illusion of a three dimensional world is created. Users of VR simulations usually interact with the environment via standard human interface devices such as the keyboard, mouse, or joystick.

Virtual reality has a number of proven areas of application, which include VR training, simulation and gaming. The main benefit of VR is the ability to totally immerse the user in an environment that would be otherwise inconvenient, or even potentially dangerous. For example, training firemen to put out fires within hazardous environments, or training military personnel to defuse explosives. A number of VR systems have been presented that take into account senses other than the visual. Virtual sounds, smells and touch have been experimented with to various degrees. Some virtual reality implementations have attempted to create simulated smells, others have endeavored to simulate the surface texture of virtual objects that are touched by the user.

It is believed that virtual reality technologies will eventually become so detailed that real and virtual environments will be indeterminable from each other. Such high VR realism requires extreme design effort, and the associated development costs are high. Despite the obvious benefits of virtual reality, the paradigm is massively constrained. The process of creating and developing high quality virtual worlds is time consuming and costly. A virtual world that is both large and of high detail would be very expensive to create; therefore a tradeoff exists between the size of the world and its level of detail. The scope of the environment is constrained by the available development resources. One solution to the problem above is to combine virtual reality with actual reality.

Augmented reality (AR), also known as mixed reality (MR) is a relatively new paradigm. The concept provides an alternative way of producing *virtual* environments. AR involves the augmentation of a virtual agent over a real-world environment. The virtual agent may be an artificial object or non-player character (NPC). This agent is overlayed over video footage of the actual environment and the composite image is displayed to the user. Figure 1.1 shows an example AR scene in which a city is visualized on a standard desktop surface. Here head mounted displays (HMD) devices with built in cameras are being used.

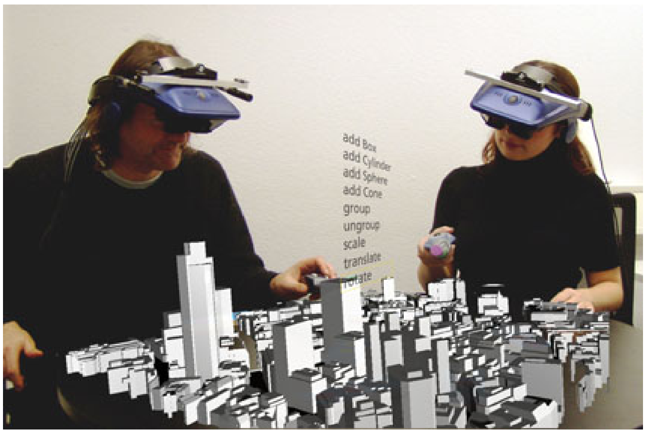


Figure 1.1: Head Mounted Display Scene[31]

AR has many areas of application, and in recent years the field has begun to receive interest from a number of sectors such as manufacturing, military, medical and the computer games industry. Many systems virtually annotate the real world, to detect and explain real world entities to the user. One example is the Battlefield Augmented Reality System (BARS) developed by the US Naval Research Facility[45]. BARS is a wearable device that attempts to gather intelligence from, and provide real-time information on, a soldier's surroundings using augmented reality. The BARS system consists of a GPS unit an antenna, a wireless network receiver, wearable computing, inertial sensors and a see through HMD. Figure 1.2 shows the BARS system as worn.

AR gaming applications such as ARQuake have been developed that allow users to interact with virtual enemies in their own every day environment. Piekarski[77] presents an AR system known as the Tinmith implementation which

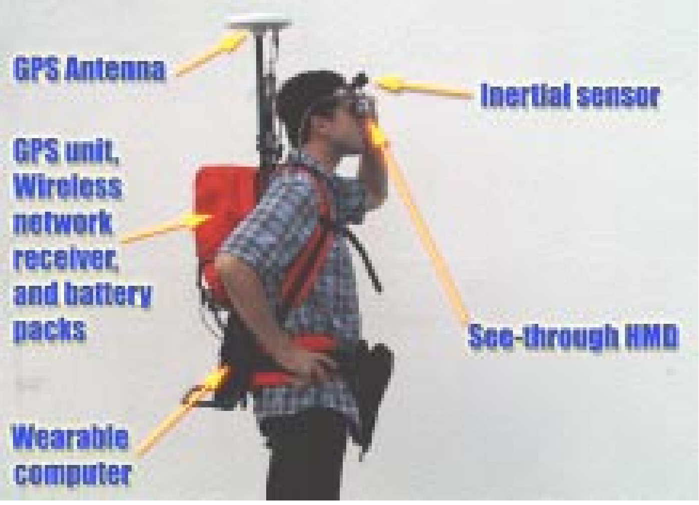


Figure 1.2: Battlefield Augmented Reality System[45]

allows the user to construct AR outdoor structures via visually tracked hand movements. The Tinmith system has since been adapted for a number of applications including medical, security, entertainment, navigation, shopping, maintenance and has military potential. The equipment is similar to that of BARS as shown in figure 1.3.

Realism is important when augmenting reality. Lack of physical and sensory immersion is essentially a failure by the system to communicate with the user, and the believability of the augmentation will be reduced. The geometric alignment between real and virtual worlds must be accurate in order to achieve augmentation realism. The process of obtaining such alignment is known as geometric registration. Geometric registration has been well researched and is both efficient and sufficiently accurate for augmented reality use, providing that certain conditions are met. A number of approaches may be used that make use of either sensor data, visual cues or a hybrid combination of both in order to align worlds. Such techniques are explained in section 3.1.1. When illumination conditions are not consistent between real and virtual worlds the illusion of realistic augmentation is destroyed as shown in figure 1.4 where the virtual object is casting shadows in a different direction to the real objects.

The process of establishing illumination consistency between the worlds is called photometric registration. In augmented reality it involves detecting the characteristics of any illuminant affecting the real component of the scene. This includes position, orientation, type and colour. Researchers have explored a number of photometric registration methods which attempt to estimate real-world illumination conditions by gathering various metrics from the real scene. The resultant data is used to illuminate the artificial component. Establishing

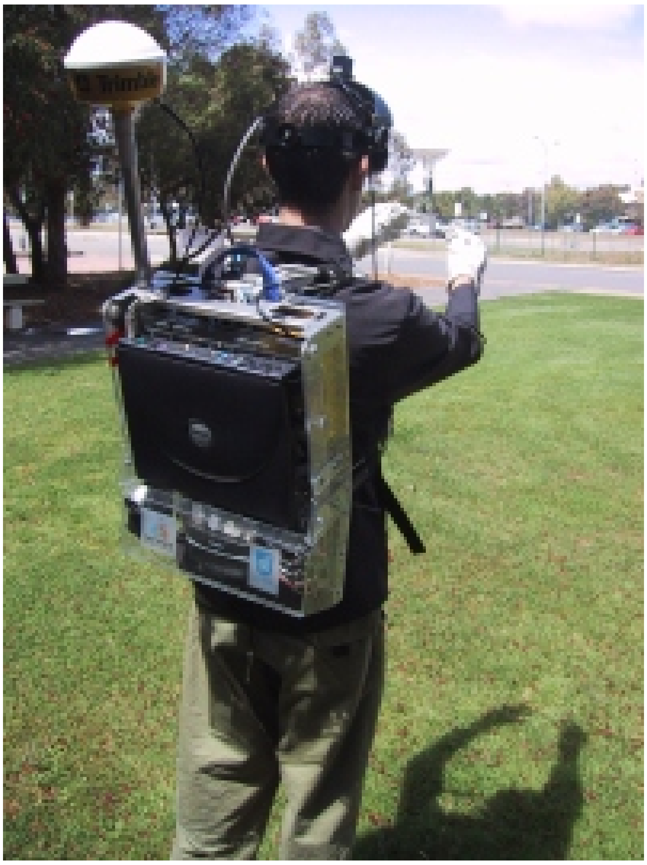


Figure 1.3: Tinmith Augmented Reality System[77]

correctly illuminated virtual components is important if the virtual scene is to appear at all integrated with the real scene. An overview of current photometric registration techniques is given in section 3.1.2.

Although there are many contributing factors, the realism of an augmented reality scene is massively dependant on sound geometric and photometric registration. Despite recent advances in virtual and augmented reality systems, the believable integration of real and virtual components is still a challenge. The geometric problem has mostly been addressed by use of either fiducial markers or interest point based approaches; however photometric registration is still problematic. This is primarily due to the unpredictable nature and complexities of the real world, especially when dealing with natural environments.

Existing photometric registration techniques have limitations that include:

- High computational complexity
- Need to pre-calibrate the scene

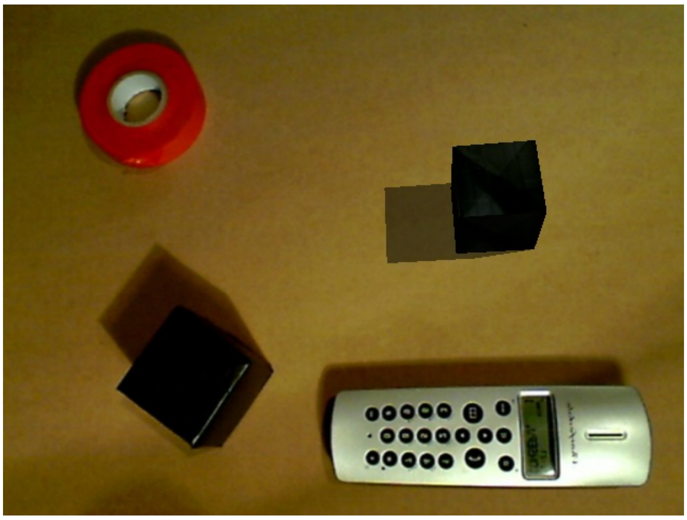


Figure 1.4: Incorrect Photometric Registration

- Requirement of persistent in-scene calibration object
- Constrained operational environment

Computationally complex techniques require copious CPU clock cycles in order to perform the necessary calculations. Any lag time induced by such a requirement would cause desynchronization between the two worlds, ultimately reducing realism and would therefore not be suitable for augmented reality application. If it took too long to recalculate an illuminant position then the virtual lighting conditions would not immediately match the real conditions. Or worse, the calculation may be so computationally demanding that it reduces the output framerate, causing the scene to jitter or even freeze. Techniques that require pre-calibration are often less computationally intense, however they make the assumption that lighting conditions and camera position are fixed and never change.

A number of techniques make significant assumptions and only operate under certain constraining conditions. For example, the system may only function in a room of known geometry where the light sources have been manually positioned in the virtual scene. Techniques that require constant calibration at runtime require some form of calibration object. These objects are unnatural in appearance and therefore destroy the believability of the scene in the same way that the deployment of fiducial marker would when considering the geometric registration of a scene. Figure 1.5 shows both a calibration object and fiducial marker.

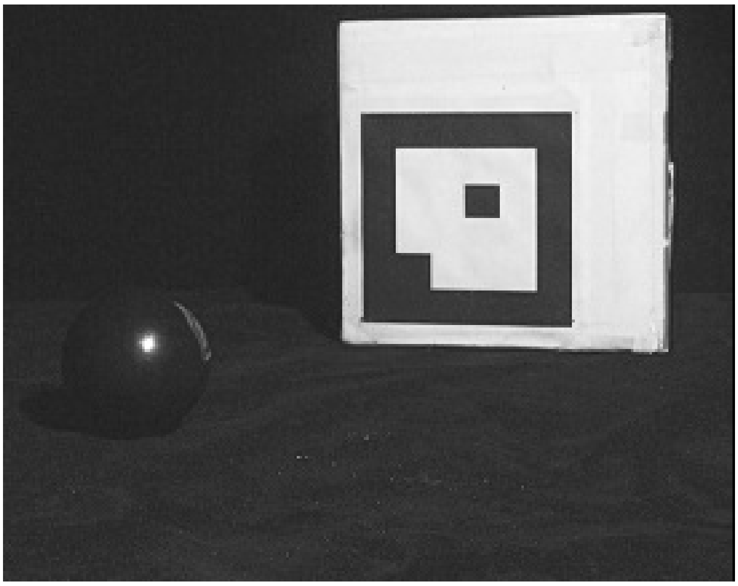


Figure 1.5: Unnatural Photometric Calibration Sphere [63]

1.2 Aims

Current state-of-the-art AR systems are able to augment reality with correct world alignment, and can do so in real-time. However, these augmentations do not completely appear to fit in with the real component of the scene, affecting scene realism and believability. Therefore it is apparent that the realism problem is yet to be addressed. This project attempts to improve augmented reality realism by performing photometric registration.

The proposed technique aims to do this by detecting the illuminant that most effects the real scene, with a focus on single illuminant environments. It aims to use natural scene features, within two input images, to estimate the illuminant position. Each input image should be an image of the real scene viewed from a uniquely different angle. The result is a set of 3D coordinates that represent the position of the real world illuminant. The technical aspects of the proposed technique are explained in chapter 4.

The method aims to be less computationally intense than other photometric registration techniques and avoids the use of unnatural features, as not to disturb the scene by inserting artificial calibration objects. The natural scene features the technique is to use are interest points that are caused by shadow and object entities within the image. It aims to be robust to movable scene geometry, moveable cameras and track dynamic illuminants, provided that the object and shadow regions are not severely occluded. It is to operate in real-time, continuously calibrating scene photometrics with the data available. If the geometric registration technique implemented does not rely on the presence of artificial components such as fiducial markers then no disruption to the real scene is required throughout the entire AR process. Investigations show that the proposed technique provides sufficient accuracy for augmented reality application even in unconstrained environments.

The project does not attempt to consider complex scene lighting, such as multiple light sources, reflected light, refracted light, or different types of light source.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 discusses the background and objectives of the project. Chapter 3 discusses other research and techniques related to the main areas of the research project. A new photometric registration technique is proposed in chapter 4, where error mitigation strategies are discussed in section 4.3. The software framework, prototypes and simulation applications are introduced in chapter 5. The project is evaluated and compared against other techniques in chapter 6. Chapter 7 discusses further work and concludes the thesis. Additional supporting information is provided in a number of appendices. Appendix A explains the concept of camera calibration, which is used by many of the techniques discussed within this thesis. Appendix B explains lighting and shading calculations that enable virtual illumination once the worlds are correctly registered and scene illuminants are detected. Appendix C discusses shadowing and relighting techniques that are able to improve scene realism using the detected illuminant coordinates.

Chapter 2

Background

Virtual reality and VR systems have been innovated since the first virtual reality system, The Sword of Damocles, was developed in 1968[71]. More recently, computer mediated reality*, including augmented and subtracted realities has become a focus area for cutting edge research. Uses for such systems are constantly being found and are gradually becoming ubiquitous. A number of augmented reality software applications have become available to buy over the last few years. One such application includes the Eye of Judgement computer game for the Sony Playstation 2 games platform, shown in figure 2.1. The Eye of Judgement plays like any other card game, except actions are played out as augmentations to a webcam feed. The geometric registration used by the game is simple, and artificial markers are printed onto the cards to assist. These markers are shown in figure 2.2.

The levels of processing power currently available allow markerless geometric registration techniques to operate in realtime. Such registration techniques are now well enough refined as to operate with relatively low complexity. This has led to the development of augmented reality applications for mobile devices. Many such applications are currently available for the Apple iPhone. Figure 2.6 shows a screen-shot of the AR FireFighter 360 iPhone application that is currently available for purchase through the Apple application store. This application is a fire fighting game and simulation, where virtual fires are augmented into the view of the user. Mobile implementations such as AR navigational systems have become popular and a mobile AR browser has recently been released that integrates with the real world via a mobile phone camera to achieve 3D annotated navigation. It is integrated with mapping data from local companies which includes information on properties and big name brands. This data is represented in an augmented

*A subset of virtual reality

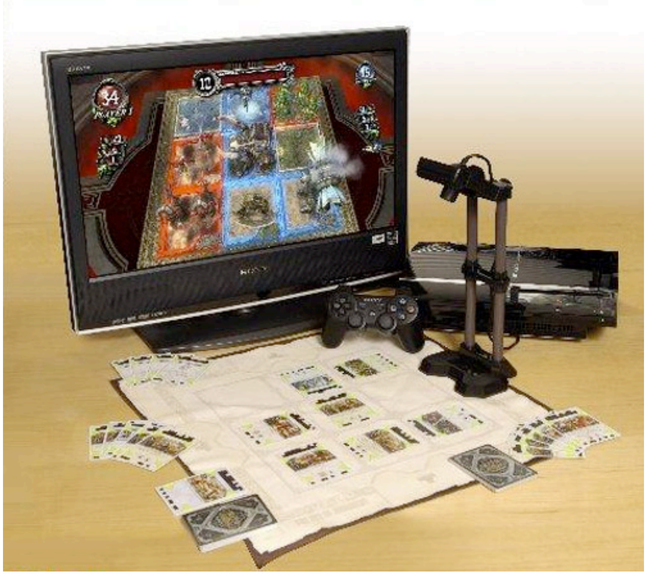


Figure 2.1: Eye of Judgement Game Setup [43]

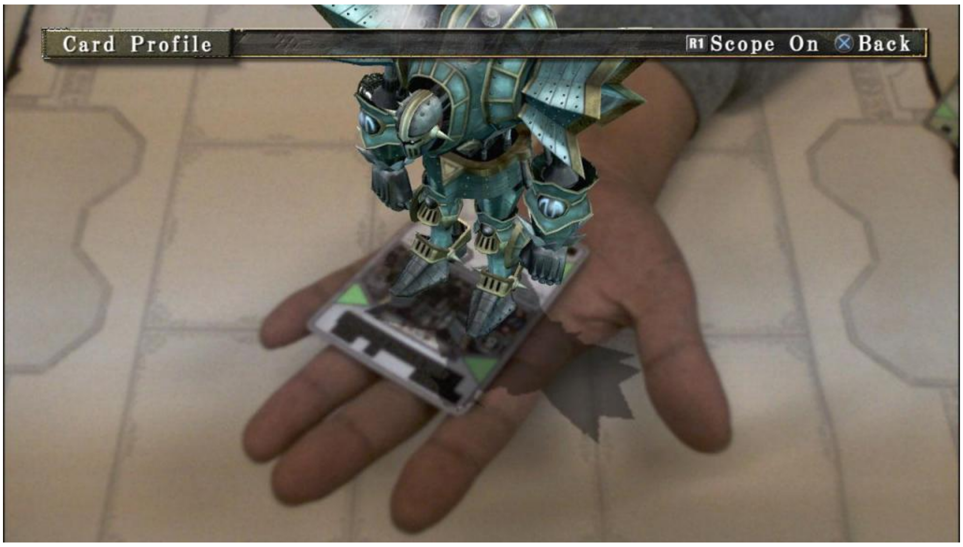


Figure 2.2: Eye of Judgement Game Augmentation[43]

reality view as can be seen in figure 2.5.

Augmented reality systems have been used within a variety of different application areas including:

- Entertainment
- Exercise

- Training
- Education
- Psychological rehabilitation
- Motion capture
- Real-time virtual actor input
- Fully immersive gaming

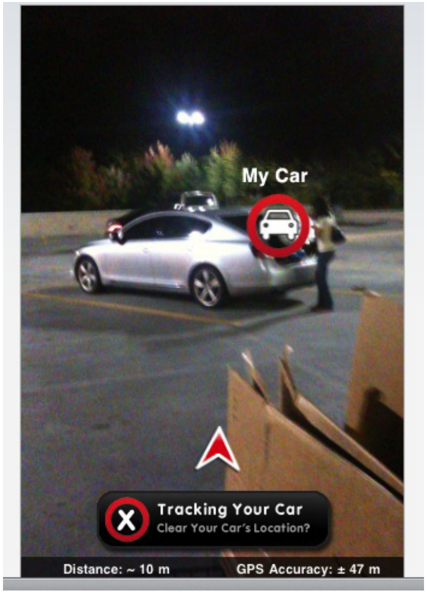
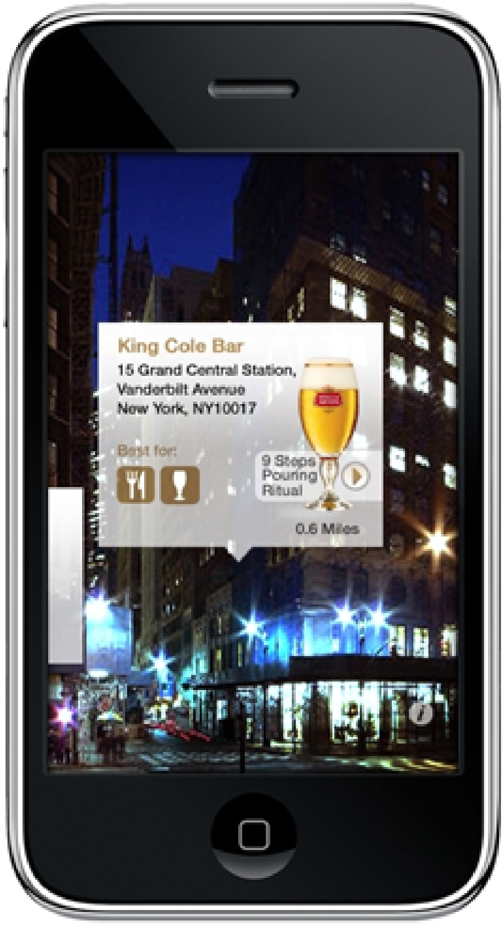


Figure 2.3: AR Car-Finder iPhone Application[66]

Despite recent demand, the technology that is integrated into AR applications and software development kits (SDK) is yet to consider realistic scene integration. Infact, even the most popular application programming interface (API) packages, including ARToolkit, ARTag and the FLARToolkit fail to include environment matching functionality beyond that of world alignment. The problem of geometric registration has now been solved and research is moving towards improving scene realism. Many applications, such as mapping and navigation, do not require advanced realism. Applications that do include gaming, simulation and training. The current state-of-the-art can provide illuminant matching, however the techniques to do so consume much computation time, constrain the operational environment or require artificial objects or markers. Significant current and past research in this area is discussed in chapter 3. Figure 2.9 shows multiple instances of the same augmented scene under different illumination conditions.



King Cole Bar

15 Grand Central Station,
Vanderbilt Avenue
New York, NY10017

Figure 2.4: AR Bar Guide iPhone Application[66]

The figure illustrates how lighting can affect how augmentations integrate with the real environment.

Beyond geometric registration, realism can be further improved by use of multi-pass rendering techniques such as used when adding shadows and reflections to a scene. Multiple executions of the render pipeline allow for the production of more complex and realistic scenes, whereas a single pass is sufficient for simple scenes. A simple AR scene, such as seen in annotative or navigational systems can be generated in just one pass using a *full viewport textured quad.* Frames from the video feed are sequentially mapped to the background texture and the geometrically registered virtual scene is drawn in the foreground. It is possible to display scenes that contain shadowed or reflective elements by making use of multi-pass rendering. Figure 2.10 diagrammatically shows the multi-pass concept.

The remaining sections of this chapter aim to provide context to problems

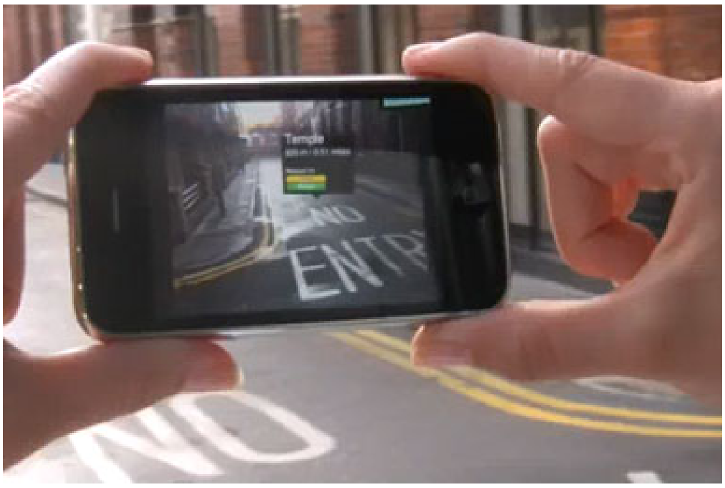


Figure 2.5: AR AcrossAir iPhone Application[66]

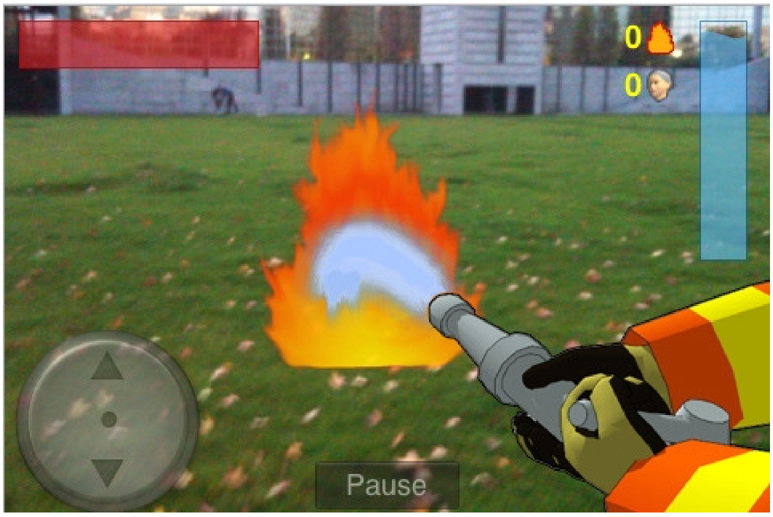


Figure 2.6: AR Fire Fighter 360 iPhone Application[66]

faced by current augmented reality systems. It is clear that AR implementations are constrained by the same hardware and software that enables them. The following discussion communicates such constraints and provides indication as to why robust software methods, such as proposed within this thesis, are important.

2.1 Hardware

AR systems achieve augmentation through a number of processes and typical AR software requires the minimum of four components, *image acquisition*, *registra-*

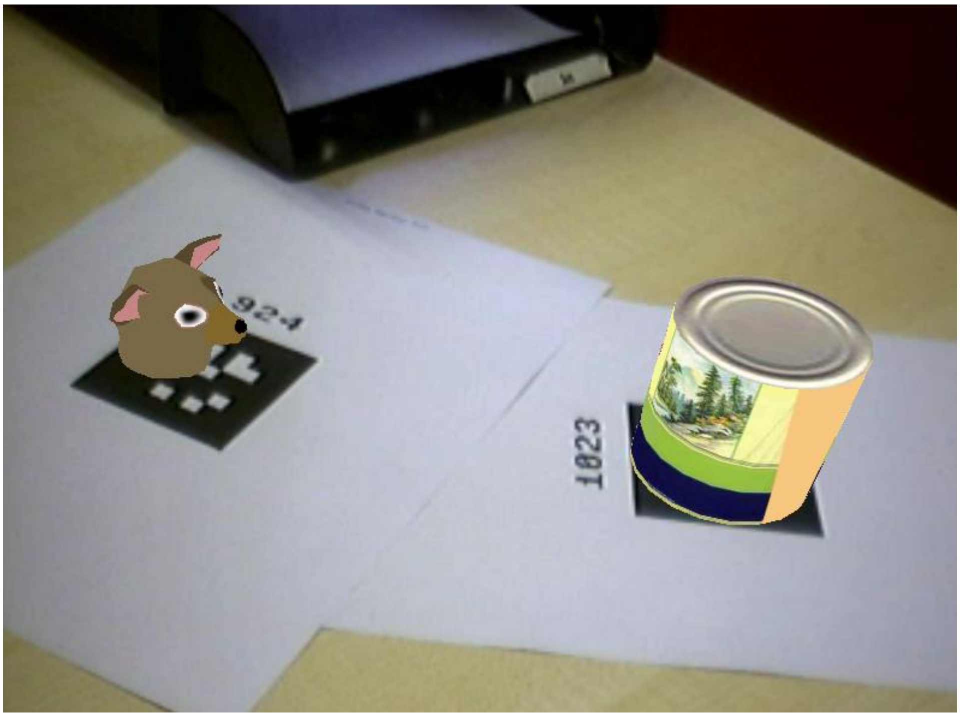


Figure 2.7: Marker Based Augmentations

tion, *augmentation* and *image output* modules. Hardware is a requirement for successful operation at each stage. Required and optional hardware is discussed in this section.

Augmented reality systems will typically employ a camera, a mobile computing device, and a display. Additional sensors are sometimes used to assist with the AR process. Figure 2.11 shows an example of typical mobile augmented reality hardware.

2.1.1 Processing

Augmented reality requires a computing platform on which to operate. Some techniques require large amounts of processing power, and therefore require more computational resources. Image processing and registration are the most demanding tasks. Less complex techniques that are relatively simple in terms of computation are able to operate on devices with low processing speeds. Mobile computing devices have recently been developed that are able to provide AR on the move. Devices such as the Apple iPhone, Android phones and some PDA devices are able to produce augmented imagery in real time. As many mobile devices, including some phones, are equipped with cameras they make ideal simple AR platforms. Today's mobile devices have relatively low processing power when compared to static solutions, therefore their AR ability is often limited. Many mobile AR applications are confined to 2D augmentations, or have very

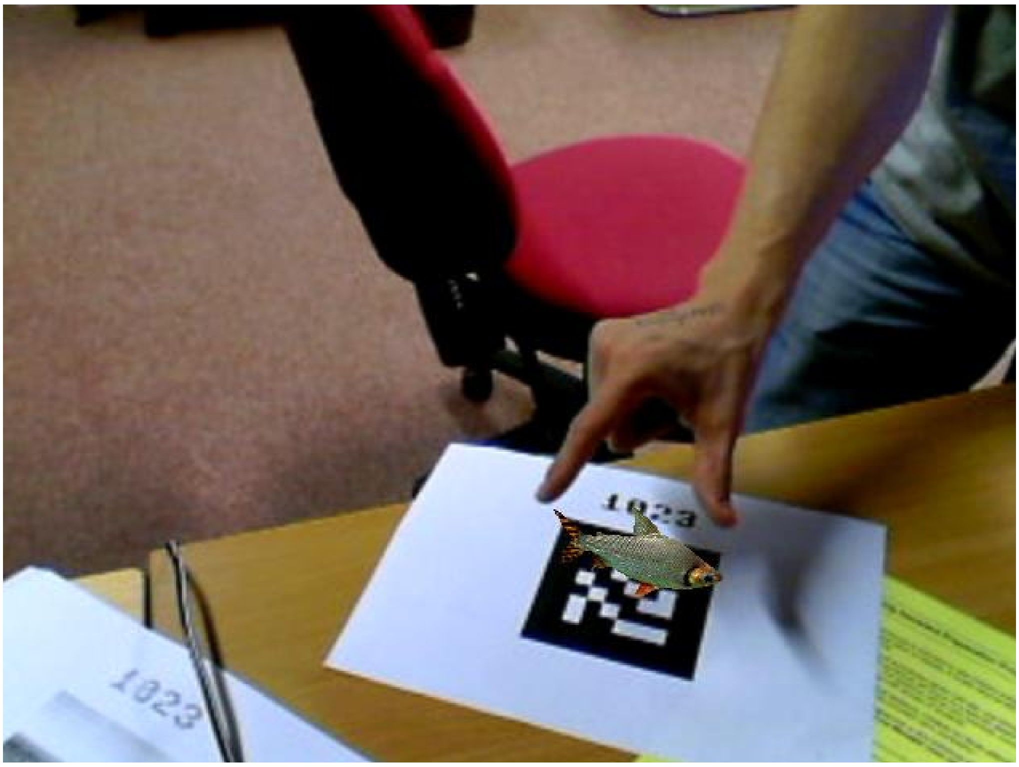


Figure 2.8: An Augmented Scene

simple 3D tracking capabilities. The Apple iPhone uses internal sensors to detect orientation and assist with pose estimation and registration processes. However this device is expensive compared to other mobile solutions. Laptop solutions offer slightly higher computing power but desktop computing usually offers the greatest flexibility. Most 3D AR solutions require no more than a computer, a standard webcam input device and a display. The desktop computers of today offer sufficient processing power to allow registration calculations to be computed in real-time on a per-frame basis. Multi-core systems offer spare processing threads that are capable of performing additional computation should it be required. Improvements to AR realism will require additional computing resources, including increased random access memory (RAM) and central processing unit (CPU) clock cycles. Parallel processing using multiple CPU cores would also improve AR performance within functionality where multi-threading is possible. It is possible to run AR applications on embedded devices, using microprocessors such as the OMAP3530 processor which is able to execute 1,200 million instructions per second (MIPS), providing 256KB of L2 cache running at up to 600MHz. Digital

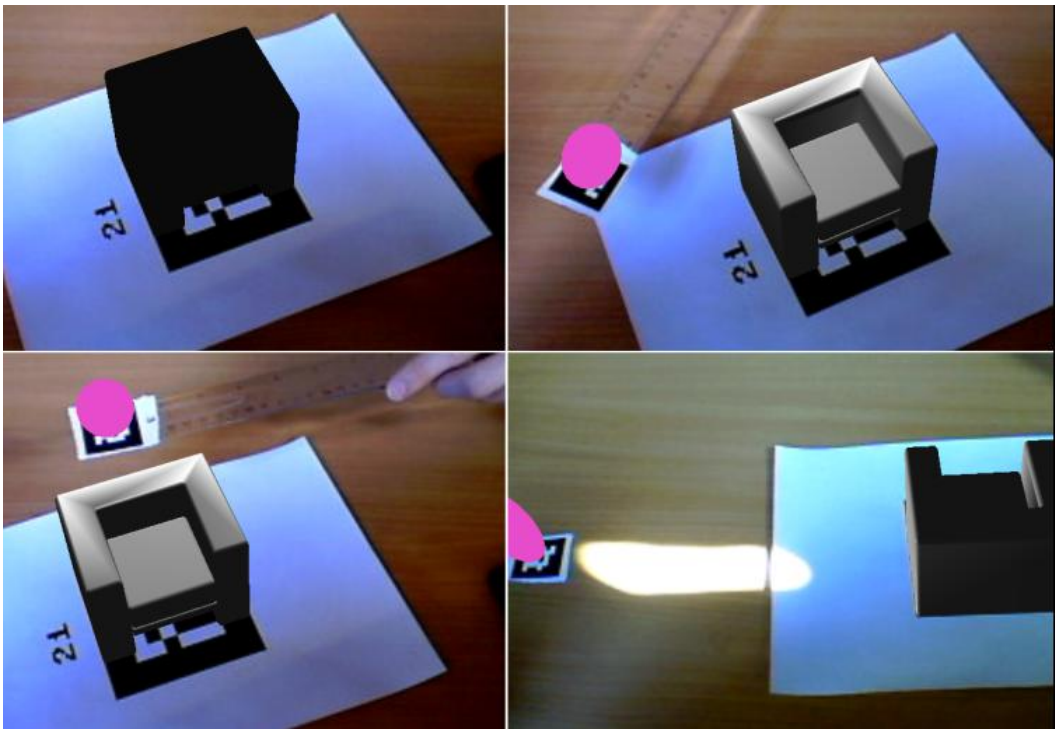


Figure 2.9: AR Scene Lighting

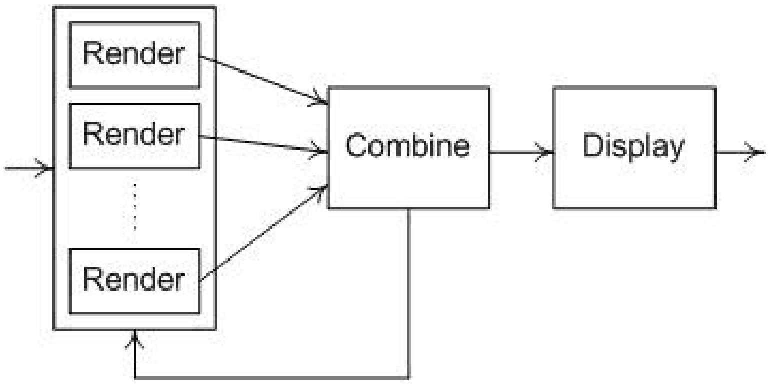


Figure 2.10: Multi-Pass Rendering

signal processing (DSP) can be accelerated by using co-processors such as the TMS320C64x+ which is designed specifically for executing DSP operations. It is able to process high definition (HD) video imagery at a cycle rate of up to

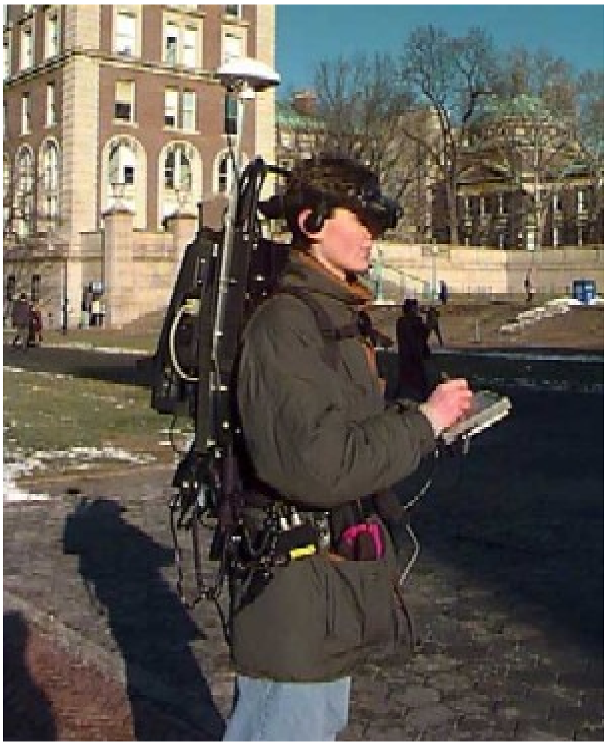


Figure 2.11: AR Hardware[80]

430MHz. Both chips operate at low power, making them ideal for mobile devices that operate on battery. Devices such as the BeagleBoard[†] integrate both mobile processor and DSP co-processor in a compact package that is well suited for AR use. Should additional computing power be required tasks may be transferred to one or more desktop or server machines via a wireless connection such as a Bluetooth, wireless ethernet, or ZigBee radio link. When data is passed to an external device for computation a certain amount of lag is induced by the transfer and scene realism may be affected as a result, especially if the delegated calculations are part of, or linked to, graphical operations. Such calculations, for example the geometric registration process, should be executed on the device where possible, and any additional processing may be delegated to a remote host. High end desktop processors such as the Intel Core i7 980X clocked at 3.33GHz are available to accommodate more complex AR functionality. Processor technology is developing at such a rapid pace that if a technique were to operate slowly today, it may be able to operate at higher rates in the near future. Computational requirements should not overwhelm the processing units as to cause deterioration of temporal resolution.[‡]. In extreme circumstances, processing can be passed off to a cluster of computers. This is commonly referred to as grid or cloud com-

[†] A compact, fan-less single board computing device

[‡] Visual systems are best viewed at 60 frames per second or above

puting. The concept involves sending packets of data, such as image frames, to be processed on various individual computers simultaneously. Each computer calculates its share of the work and returns the result to the client. In situations where slight network lag is acceptable, the processing speed can be improved dramatically at no cost to visual realism. Cluster solutions can be created easily using software such as the Boot-able Cluster CD (BCCD). The BCCD is able to create a cluster from any number of standard machines that are booted with the CD in their drive. Other cluster solutions such as the Berkeley Open Infrastructure for Network Computing (BOINK) provide free, volunteer based computing, and are able to supply vast computing power. BOINK also makes use of any graphics processing units (GPUs) available on each client machine. The graphics cards in today's computers are vastly more powerful than most CPUs. If network lag would cause unacceptable visual results then it is possible to pass processing calculations off to the GPU. This activity is referred to as *general-purpose computing on graphics processing units* (GPGPU). GPGPU computing is an emerging field that is providing solutions that enable real time processing of complex tasks. It should be noted that although GPGPU processing is much faster than CPU processing, there is a delay when retrieving the result of the calculation from the GPU. These lag times are much less than those caused by network processing. This should be taken into consideration when deciding which tasks are delegated and where they are delegated to. High end GPUs of today, such as the ATI FireGL V8650, offer 320 parallel shader processors that may be clocked at speeds of 3.5GHz and supply 2GB of DDR4 RAM. GPGPU capable GPUs are programmed using a language such as CUDA or OpenCL. CUDA is a language developed by the NVidia Corporation and allows for the programming of NVidia GPUs for general purpose computing. OpenCL is a new language that provides a framework for writing applications that execute across heterogenous platforms consisting of CPUs, GPUs and other processors. OpenCL gives access to the graphical processing units available on the machine for non graphical computing.

2.1.2 Displays

Video output devices such as televisions, computer monitors and video projectors are all suitable as augmented reality displays. The chosen display affects the realism and believability of the experience. Three dimensional computer generated views may be created in order to enhance realism. Two different views of the same scene are required to achieve 3D depth perception via the process of stereopsis. Stereopsis is the perceptual transformation of differences between the two images as seen by human eyes. The eye separation, or interocular distance, causes image perception to be slightly shifted horizontally and rotated about the vertical axis. This causes retinal disparity, which is a 2D relative displacement between two projections of a single object. The two images allow for estimation of the approximate distance of objects. This is possible providing that the object exists within the human binocular vision region as shown in figure 2.12. Depth information can not be approximated if the disparity is too large. The monocular and foveal vision fields are also shown. The monocular vision field is the 180° horizonal and 130° vertical field of view in which human vision extends. Regions that fall within this range but not the binocular range can be seen but can provide no depth estimation. The foveal field of vision is the area where both eyes can see in focus, it extends over a 60° field of view.

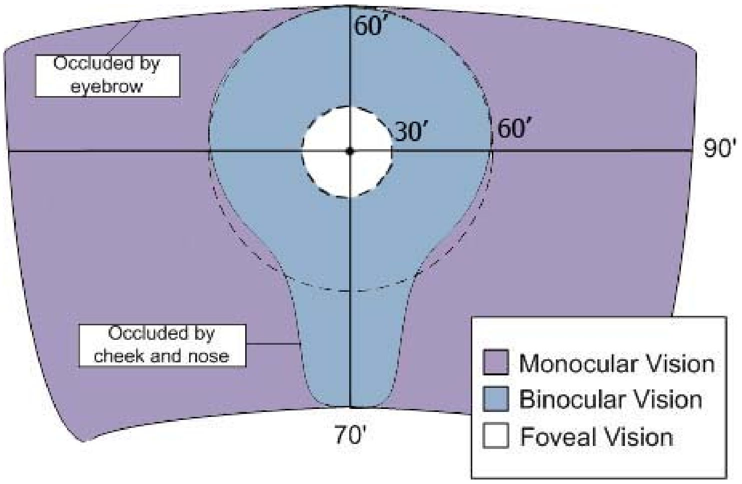


Figure 2.12: Human Visual Fields

Human stereoscopic vision can be tricked by using a stereoscopic display whereby two computer images with an artificial disparity are displayed to each eye respectively. The brain is then tricked by the illusion of artificial depth. Stereoscopic displays can be divided into two main categories, goggle-bound and auto-stereoscopic. Auto-stereoscopic displays do not require additional glasses to support the proper separation of stereo images. Available displays and their classification are outlined in figure 2.13.

Goggle-bound displays separate the stereo images using additional glasses, they can be classified as either head-attached or spacial. Head attached displays provide a separate dedicated display for each eye and are able to present each image simultaneously. Most head mounted displays use two liquid crystal display (LCD) screens or miniature cathode-ray tube (CRT) screens, boom-like HMDs are also available. Retinal displays are available that scan modulated light directly onto the human retina using a low power laser beam[53]. The resolution improvement when using a retinal display is significant[81]. Head mounted projective displays such as discussed by Parsons[73], Inami[42], and Hua[41] and projective head mounted displays such as discussed by Kijima[48] are head-mounted projectors that can be used instead of small displays. Head mounted projective displays work by redirecting the projection frustum with a mirror beam combiner and the images are beamed onto surfaces in front of the user.

Augmented reality systems have typically made use of head mounted displays, however more recently the use of mobile phone displays has increased[95]. Spatial displays make use of screens that are spatially aligned. The user has to wear shutter glasses or light filtering glasses, which are either polarized or colour filtered goggles. These goggles facilitate the separation of stereo imagery. Shutter glasses present the imagery sequentially, alternating between each view and essentially shutting out the eye which is not the intended recipient. This is not true of filter glasses where images are displayed simultaneously. Each eye uses a different filter, only the correct image is allowed to pass through the filter. In this circumstance the images are encoded in two different colours which are typically red/green, red,blue or red,cyan. The filter glasses ensure that only the one correct image reaches the desired eye. This is called using a *passively filtered anaglyph*. The main limitation of this technique is that although the brain is able to fuse the two images into one and estimate depth information, the image is perceived as monochrome. This problem can be solved by displaying a third image that contains the colour information, this is called a *full colour anaglyph*. Displays also exist that make use of the Chroma depth method[5] and the Pulfrich effect[87]. Polarization glasses are the most common stereoscopic passive shuttering technique. They operate by polarizing the light in two directions using special filters. This allows the glasses to correctly separate the two images. It does so by using filter polarization that is identical to the corresponding light polarization. Therefore only the correct image is allowed to enter the eye. The advantages of this technique are that it maintains proper colour and intensity information, does not disrupt scene quality and scene realism is unaffected. Spatial displays can make use of desktop configurations such as a standard computer monitor or projection systems. The field of virtual reality often makes use of desktop monitors as stereoscopic displays. A refresh rate of 120hz is required for time sequential shuttering, therefore LCD shutter glasses are mostly used for stereo separation using an active shuttering technique. Reach-in systems such as those discussed by Knowlton[51], Schmandt[85], Poston[78] and Wiegand[99] can present stereoscopic 3D graphics to a user who is able to reach into the visual space by interacting below a mirror that reflects an image on an upside down CRT display. This method of interacting with the virtual world does not occlude the graphics being displayed. Projection displays beam either stereo or standard imagery onto one or more planar or curved surfaces using CRT, LCD, liquid crystal on silicon (LCOS) or digital light projectors. Projectors can be classified into front or rear projection systems. Front projection systems project light onto a surface from the same side of the surface as the observer, whereas rear projection systems are located behind the surface. Active and passive shuttering techniques work with projection systems in a similar way as with screen-based systems. When using passive shuttering, two projectors are required in order to project both filtered or polarized stereo images simultaneously onto a single surface. When using active shuttering only one projector is required as the images are displayed sequentially. A refresh rate of 120hz is still required with projection systems. This obtains a refresh rate of 60hz per eye, ensuring that no flicker can be seen by either eye. Surround screen displays are available that work by surrounding the viewer with multiple displays such as CAVEs as discussed by Cruz-Neira[14] and CABINs as discussed by Hirose[37]. These systems attempt to remove as much of the real environment from the perception of the viewer as possible. Planar surfaces are usually employed for this, but curved surfaces such as domes and panoramic displays can also be used to further enhance the immersion of the user. Multiple projectors allow for the coverage of extended surfaces. Embedded screen displays can be integrated into the real environment, to provide a semi-immersive environment such as discussed by Krueger [56][55]. CAVE displays project imagery onto each wall of a cube shaped room in which the user stands. The resulting 3D environment is highly immersive.

Autostereoscopic or automultiscopic displays do not require the use of additional glasses. Autostereoscopic devices function by use of autostereoscopy methods for which a number of solutions exist, including the use of lenticular lenses or parallax barriers. These techniques cause each eye to see a different image. They function by varying the image shown depending on the angle it is viewed at. Human eyes are spaced some distance from each other, therefore using this concept it is possible to create a display that will differ as viewed from each eye. A parallax barrier device allows a LCD to create depth via a series of precision slits. The slits control which pixels can be seen by each eye. The drawback of a parallax device is that the user has to be within a very specific area relative to the display. Lenticular lenses are an array of magnifying lenses that are designed to magnify different images depending on the location of the viewer. Similar to parallax devices, the angle the device is viewed from decides which image is to be magnified and displayed. This concept was originally used in lenticular printing which involves digitally combining several images by interlacing them. The combined image is then printed and a lenticular lens placed over them. When viewed from different angles different images are shown, giving the illusion of animation. More recently this concept has been adapted to work with digital video displays. Although autostereoscopic displays have the advantage of not needing additional head-mounted hardware or glasses, they have the disadvantage of causing headaches and dizziness over prolonged use. Re-imaging displays are available that make use of lenses and/or mirrors in order to generate copies of existing objects. An early example of this being the Pepper's ghost configurations which make use of half silvered screens, placed at 45° angles, that partially mirror the environment[§]. By doing this the screen artificially combines two images[96]. Modern Pepper's ghost implementations are able to use this approach to combine computerized images with real scenes. Instead of combining two real scenes they are able to combine a real scene with an LCD or CRT display or two LCD or CRT displays together[89]. This makes them useful in augmented reality or stereoscopic application [67][68]. Volumetric displays function by directly illuminating spatial points within a display volume. The graphic is formed in three physical dimensions instead of being projected onto a two dimensional plane or curved screen. This allows a volumetric display to project images of voxelized data and 3D primitives. They work by filling or sweeping out a volumetric image space. A variation on this is the solid-state volumetric display. These displays project voxel data within a translucent substrate by generating light points. Lasers with varied wavelengths are used for this purpose and are scanned through the substrate causing an image to form[18]. Volumetric imagery can also be generated by use of a time-multiplexed series of 2D imagery displayed via a fast moving or spinning display element. Such displays are referred to as multi-planar volumetric displays. Varifocal mirror displays such as discussed by Traub[93], Fuchs[25] and McKay[67][68] use flexible mirrors to take the image on a CRT screen and place it at different depth planes within the image volume. Some implementations do this by vibrating the mirror optics using a loud speaker [25]. Other approaches make use of vacuums to vary the mirror position and change the focal length of the optics [67][68]. This method synchronizes vibrations with the refresh rate of the reflected screen, allowing the spatial appearance of reflected pixels to be controlled and placed at the desired depth. This requires no stereoscopic separation. Holography, the process of creating a hologram, can also be used. Holograms are defined as photometric emulsions that record the interference pattern of coherent light. The stored photometrics include the amplitude, wavelength and phase of light waves as opposed to standard cameras which only record amplitude and wavelength information. This allows holographic displays to reconstruct the entire optical wavefront, resulting in a 3D appearance than can be viewed from a variable perspective. Computer generated holograms, created via a process known as electro-holography[62] can construct holographic recordings from renders taken at various perspectives and combining them. Vast amounts of storage and processing are required for this operation, therefore results are limited by the computing hardware currently available.

In an attempt at improving scene realism both 2D and 3D displays are being researched and developed within both academia and industry. Shutter glasses such as those currently being marketed by companies such as NVidia are popu-

[§] Named after Professor John Henry Pepper lar for computer gaming application. Autostereoscopic 3D devices that do not require additional headsets are currently under development as the below quote from a memo released by the Nintendo Corporation on March 23, 2010 shows.

> To Whom It May Concern: Re: Launch of New Portable Game Machine Nintendo Co., Ltd.(Minami-ward of Kyoto-city, President Satoru Iwata) will launch *"Nintendo 3DS"* (temp) during the fiscal year ending March 2011, on which games can be enjoyed with 3D effects without the need for any special glasses.
>
> "Nintendo 3DS"(temp) is going to be the new portable game machine to succeed "Nintendo DS series", whole cumulative sales consolidated sales from Nintendo amounted to 125million units as of the end of December 2009, and will include backward compatibility so that the software for Nintendo DS series, including the ones for Nintendo DSi, can also be enjoyed.
>
> We are planning to announce additional details at E3 show, which is scheduled to be held from June 15, 2010 at Los Angeles in the U.S.

It would seem that industry can see significant commercial gain in the advancement of vision and virtual display devices and the associated realism of imagery. When choosing a suitable augmented reality display the following visual attributes should be considered:

- Colour range and clarity
- Brightness
- Contrast
- Spatial resolution
- Number of display channels
- Focal distance
- Opacity
- Field of view
- Susceptibility to occlusion
- Field of regard
- Temporal resolution

In addition to this, logistical attributes that should be factored into feasibility decisions include:

- User mobility
- Tracking method compatibility
- Portability
- Multi-user capability
- Encumbrance
- Safety
- Cost

2.1.3 Sensors

A number of techniques, such as presented by Haller[32], Piekarski[77] and Behringer[9], use additional information from sensors such as the global positioning system (GPS) and radar equipment to assist the visual tracking process. The GPS system is a United States space-based global positioning and navigation system. A GPS client device is able to derive its position anywhere on or near the Earth so long as it has an unobstructed view of at least four satellites. The system has become widely used as a navigation aid for both civilian and military purposes, on land, at sea or in the air. GPS has been used for many other applications including augmented reality. The Tinmith device[77] uses GPS for user location and to provide camera coordinates during the pose estimation process. Figure 2.14 shows a commercial GPS system used for navigational purposes.

The GPS network can be used to determine the position of the camera. Such systems are only able to operate when sufficient signal from four or more satellites can be obtained. GPS signals become sporadic in built up areas and can be blocked completely when the receiver is indoors. GPS information is less reliable in such areas.

Devices such as inertial sensors, laser measuring devices and magnetic field sensors have also been used to measure the orientation of AR devices. They are often used to track the head or hands of a user. Accuracy and precision of such equipment is usually proportional to its cost. These sensors have limitations, for example GPS devices are only able to provide directional information so long as the device is in motion. The direction vector is calculated from two sequential position fixes, therefore if a device is stationary, or is not moving fast enough to derive a direction, then the calculation is not possible. An electronic compass is an alternative. Electronic compass devices allow the system to determine the heading of the user relative to the magnetic field of the Earth's magnetic poles. Although today's electronic compasses are accurate compared to traditional magnetic compasses, they are still susceptible to interference from localized magnetic fields. Fibre optic gyrocompasses are more robust[10] but they are also more expensive. However gyroscopic sensors are prone to errors due to the drift caused by bearing friction within the gyroscope. Sensors may provide accurate data given that the correct sensors are used for the surrounding environment, however may be costly and bulky to carry. Due to this, many AR researchers and developers prefer to make sole use of visual sensing equipment. Such visual equipment is discussed in section 2.1.4.

Inertial measurement units (IMU) are electronic devices that measure and report the velocity, orientation and gravitational forces acting upon a device using a combination of accelerometers and gyroscopes. They require an onboard embedded processing device such as a programmable interface controller (PIC) microprocessor. IMU devices are typically used to maneuver unmanned aerial vehicles but have also been used in other autonomous applications and within augmented reality systems[40]. A gyroscope device is able to measure or maintain orientation using the principles of the conservation of angular momentum. Accelerometer devices are able to measure acceleration of a body on which they are mounted. The magnitude and direction are expressed as a vector relative to the acceleration vector experienced during free-fall. Cumulative error is experienced with both gyroscopes and accelerometers. The error with accelerometers is much worse than gyroscopes. Gyroscopes such as the LISY300AL are fit for this purpose and cost $7.95 USD per unit. Suitable accelerometers include the MMA7260Q which costs $11.80 per unit. Error within such devices can be partially compensated for. With combined GPS and IMU systems it is possible to detect the location and pose of the AR user, and then track movement and changes in orientation. The addition of IMU sensors allows AR applications to rapidly update and provides robustness in times where some or all satellites are occluded by objects such as buildings or trees. They are able to do this using a navigation method known as dead reckoning. Dead reckoning is the process of estimating the current position of a device based upon a previously determined position and advancing it by an estimated velocity. The problem with this approach is that the errors induced by both IMU devices and the dead reckoning calculation are cumulative, leading to *drift*. This is an ever increasing distance between where the device is actually positioned and where it thinks it currently is. Additional sensors such as the GPS device and an electronic compass are able to correct drift at frequent periods. In AR application both the error and the correction of the error would cause realism problems. As such error would create problems with geometric registration by causing the virtual and real scenes to appear out of alignment periodically. Once drift and other errors were compensated for, the world would realign, causing a jerk of movement that would look unrealistic and would be potentially disorientating for the user. Lower cost sensors typically accumulate greater error than high cost sensors. Tables 2.1, 2.2, 2.3 and 2.4 show the functionality and associated cost of sensing devices. A visual solution would allow robust pose estimation at reduced cost. All AR applications require a video input, therefore if visual approaches were used instead of accelerometer and gyroscope based IMU approaches then no additional equipment is required, although a hybrid sensor/visual approach would offer improved robustness. Figure 2.16 shows a number of components that when combined would form a complete AR pose estimation platform. This includes GPS module, Miniature complimentary metal-oxide semiconductor (CMOS) camera, PIC microprocessor[¶], IMU board complete with gyroscopes and inertial sensors and two ZigBee[‖] nodes for communicating with external computing devices[74]. Possible external computing devices include a laptop computer for performing image processing and augmentation, and visual display units (VDUs). Devices such as laser distance measuring and radar may be useful in some circumstances to assist with pose estimation. Imaging photometers and colourimeters such as the LI-210 Photometric Sensor are able to measure light intensity. Locating the light sources using hardware alone is still problematic, especially when the illuminant is out of the field of view of the sensing array. This type of sensor is expensive at a cost of around $1600 USD per unit. Sensors to establish the colour of light emitted from an illuminant, such as the Avago ADJD-S371-Q999, are cheap and can be obtained for around $9.96 USD per device. General purpose cameras can also be used for this purpose. Camera hardware is discussed later in this chapter. Distances from walls and other objects can be measured using laser or ultrasonic measuring devices, and can assist with the pose estimation process, especially when operating indoors, however these devices are costly.

GPS	Ch	Sensitivity	Acc.	Lock	Dim.(mm)	Power	Cost
58048-00	12	-152dBm	<3m	39sec	19x19x2.54	28.5mA,3.3V	$51.95
MN1010	12	-152dBm	<3m	42sec	10x10x2	35mA,1.8V	$19.95
GS405	20	-143dBm	<5m	42sec	52.1x25.6x3	75mA,3.3V	$89.95
FV-M8	32	-158dBm	2.6m	36sec	30x30x8.5	33mA,3.3V	$99.95
GS407	50	-158dBm	2.6m	29sec	47.1x22.9x7.5	75mA,3.3V	$89.95

Table 2.1: Comparison of GPS Components

A three way tradeoff exists with position sensing sensing devices between the following factors:

1. Accuracy, position and update speed

2. Susceptibility to interference

[¶] For processing sensor data

[‖] Long range wireless mesh devices

Compass	Heading Res.	Update Rate	Power	Cost
HMC6352	0.8°	5hz	1mA, 3V	$34.95
HMC6343	0.5°	1-20hz	3.5mA, 3.3V	$149.95
OS4000-T	0.4°	1-20hz	3.5mA, 3V	$249.95
OS5000-S	0.2°	1-20hz	3.5mA, 3.5V	$289.95
HMC5843	0.5	10hz	3.3mA, 2.5V	$49.95

Table 2.2: Comparison of Compass Components

Gyro	Axis	Max Rate/sec	Cost
ADXRS401	1	75°	$44.95
MLX90609NZ	1	75°	$39.95
ADXR5610	1	300°	$51.95
IDG1215	2	67°	$24.95
ITG3200	3	15°	$24.95

Table 2.3: Comparison of Gyro Components

Accelerometer	Resolution	Resolution	Cost
MMA7260Q	3	$^{+}/_{-}$2.56g	$11.80
ADXL203	2	$^{+}/_{-}$1.9g	$27.95
ADXL213	2	$^{+}/_{-}$1.9g	$27.95

Table 2.4: Comparison of Accelerometer Components

3. Encumbrance

Higher cost devices may be more optimal but no electronic sensor based technology is able to provide perfect results in all three areas. A videometric approach is a solution that would mitigate the limitations and constraints of methods that are exclusively sensor based.

2.1.4 Video Input Devices

Camera input devices are required for AR application as they provide the live feed on which virtual augmentations are superimposed. The cost of video devices vary significantly. Low cost devices include webcam video capture devices, often connected via universal serial bus (USB). They provide basic frame capture functionality via a frame grabber[**]. Cheap webcam devices usually have low bandwidth capability and therefore bandwidth saving techniques such as interlacing are frequently adopted. Interlaced video is divided into scan lines and

[**] Software or driver that acquires an image snapshot from the hardware and delivers the data to the requesting application

is transmitted in half each frame, alternating for each subsequent frame. Although this technique has solved the bandwidth limitation problem, it causes many problems when performing image processing, especially when the camera or scene elements are fast moving. When interlaced video is fast moving it appears to be torn, as a result it is difficult to locate any features that might have been present. Figure 2.17 shows the problem caused by interlacing an image.

De-interlacing techniques exist, however they introduce lag that presents realism problems within augmented reality worlds by causing additional delays when performing registration. The image quality of low cost webcam images is sufficient for basic augmented reality applications, usually when the scene to be augmented is constrained to a small area which is close to the camera. This is primarily due to typical charged-coupled device (CCD) sensor resolutions of 320x240 and 640x480 for streaming video. It may be possible to obtain higher resolution still images from the same devices but this is of no use for most AR applications that require video streams of at least 15 frames per second. Medium cost webcam devices are able to provide higher resolution images at a greater framerate. Higher image resolutions provide greater precision when performing registration as is explained in chapter 4. High end devices offer quality imagery at both high spatial resolution and temporal resolution. This creates greater demand for bandwidth. Firewire connections can handle data transfer at a much higher rate than USB connections, therefore when using a high quality camera the computing device should be able to handle firewire connectivity. High cost cameras are able to provide better quality imagery due to the quality of the sensors and lenses used.

Table 2.5 compares different camera input devices and the associated cost.

Camera	Resolution	FPS	Features	Cost
LT Quickcam 3000	640x480	30	None	£14.99
LT Quickcam 5000	640x480	30	Medium quality lens, Auto-lighting	£42.95
LT Quickcam 9000	1600x1200	30	High quality lens, Auto-lighting	£80.00
HP Premium Webcam	640x480	30	Auto focus	£30.00
Nikon CP S1000PJ	12.1MP	60	High quality lens, Auto-lighting Auto-focus, High-resolution	£285.00

Table 2.5: Comparison of Camera Input Devices

2.2 AR SDKs

A number of SDK frameworks have been created to facilitate the rapid development of augmented reality applications. They focus on geometric registration and mostly make use of fiducial marker technology in order to achieve camera pose estimation. In order to use such development kits, the programmer merely has to print out the chosen marker set and create program code that tells the software exactly how to react to each marker. AR specific functionality such as registration and augmentation is handled by the API library itself allowing the programmer to concentrate on application specific code.

The first to emerge was ARToolkit which was designed as a simple framework for creating real time augmented reality applications on multiple platforms. It was originally created for the C++ programming language but has since been ported to other languages including web-based Flash. The varieties and ports of ARToolkit include:

- ARToolkit
- FLARToolkit
- AndAR
- ARDesktop
- nyARToolkit
- SLARToolkit
- ARToolkitPlus

The ARToolkit application inspired the development of the ARTag SDK by Fiala[23], which is an augmented reality SDK that allows virtual objects, games and animations to enter the real world by adding 3D graphics to live video. It does this by performing geometric registration using specially designed fiducial markers. ARTag estimates camera pose for each marker, allowing augmented entities to be placed upon them. Two demonstration applications that are shipped with ARTag include the *magic lens* which allows the user to look through a device, viewing the augmentations as if they were placed in front of him or her, and the *magic mirror* application that allows a user to see a reflection of themselves complete with augmentations. The augmentations in the magic mirror application are able to track the movements of body parts, this allows for the seamless augmentation of clothing and the addition of other virtually worn accessories. The SDK allows markers to be tracked and the modelview matrix[††] is exposed. The magic lens and magic mirror functionality of ARTag is shown in figures 2.18 and 2.19 respectively.

The mixed reality toolkit (MRT) is another augmented reality framework currently being developed at University College London. It operates in a similar manner to ARTag but claims to be more efficient and robust due to its geometric modeling technique[24].

[††] This modelview matrix describes the pose of each marker

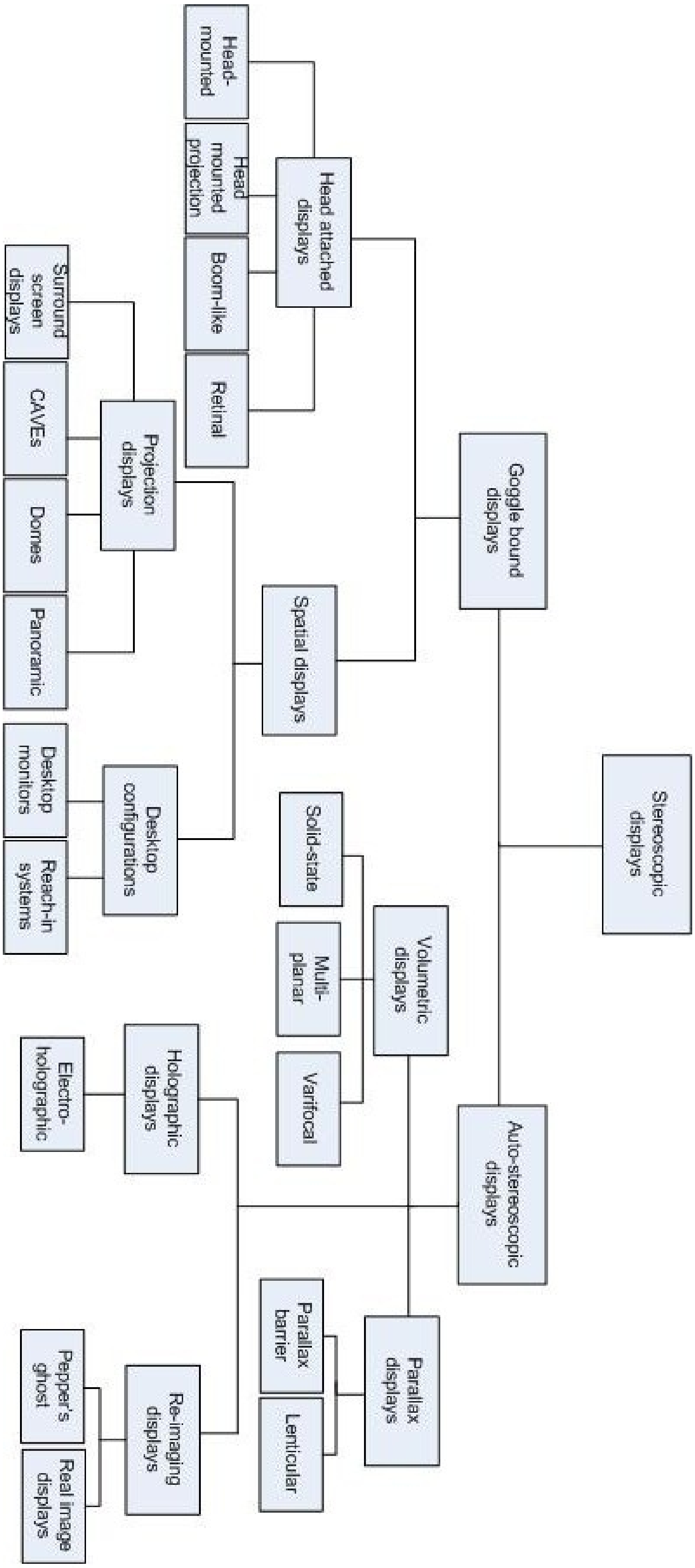


Figure 2.13: Classification of Stereoscopic Displays

Figure 2.14: GPS Navigation [20]

Figure 2.15: GPS Component [20]

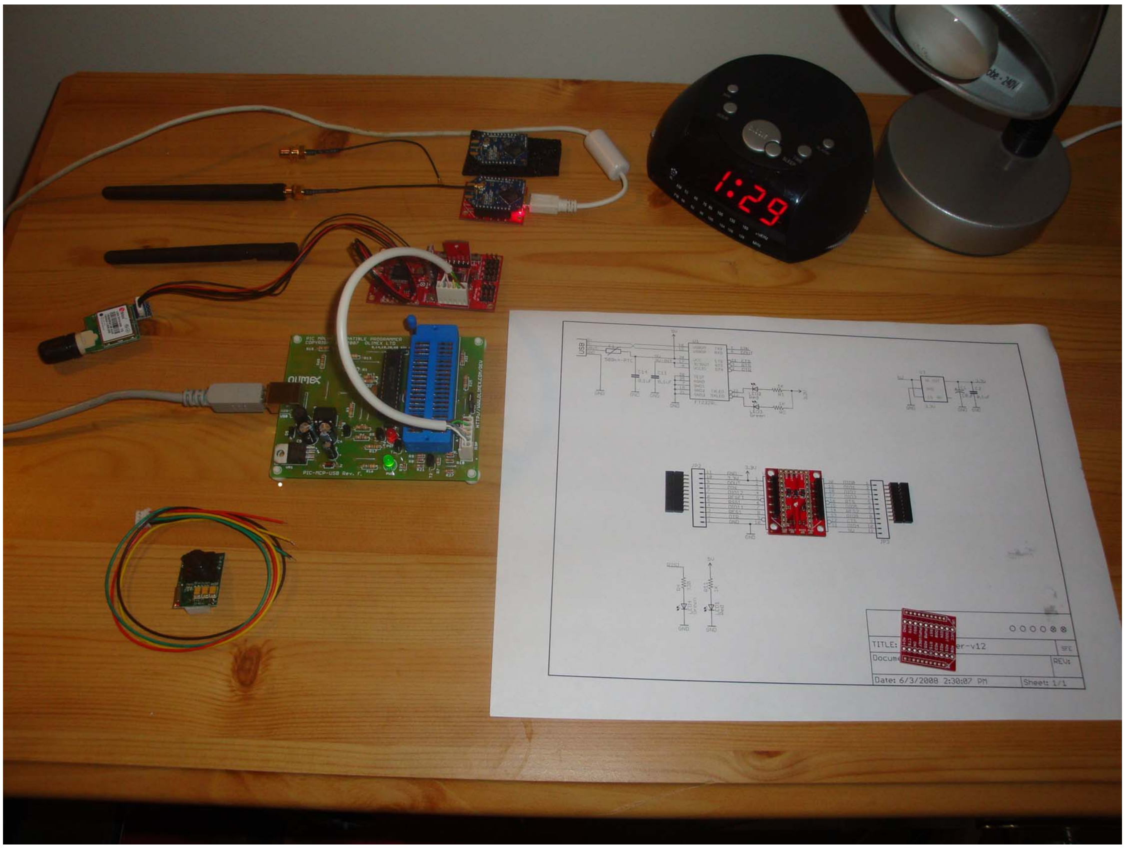


Figure 2.16: AR Base Hardware Components

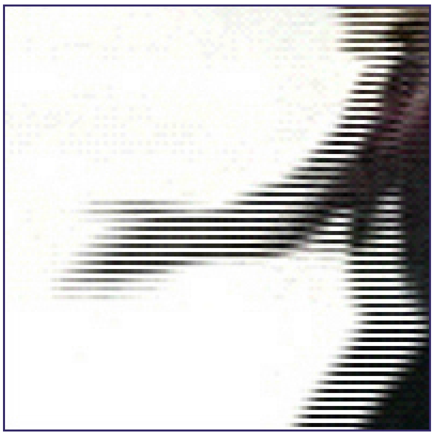


Figure 2.17: Interlace Problem

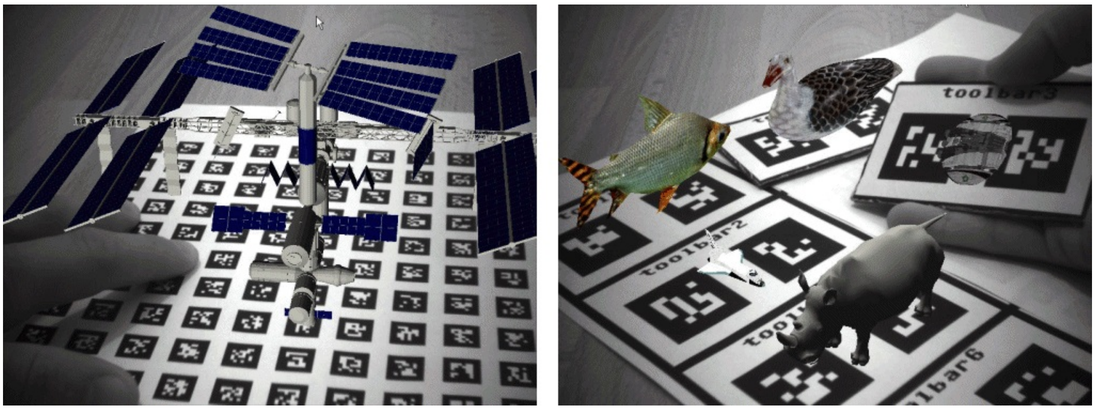


Figure 2.18: ARTag Magic Lens[4]

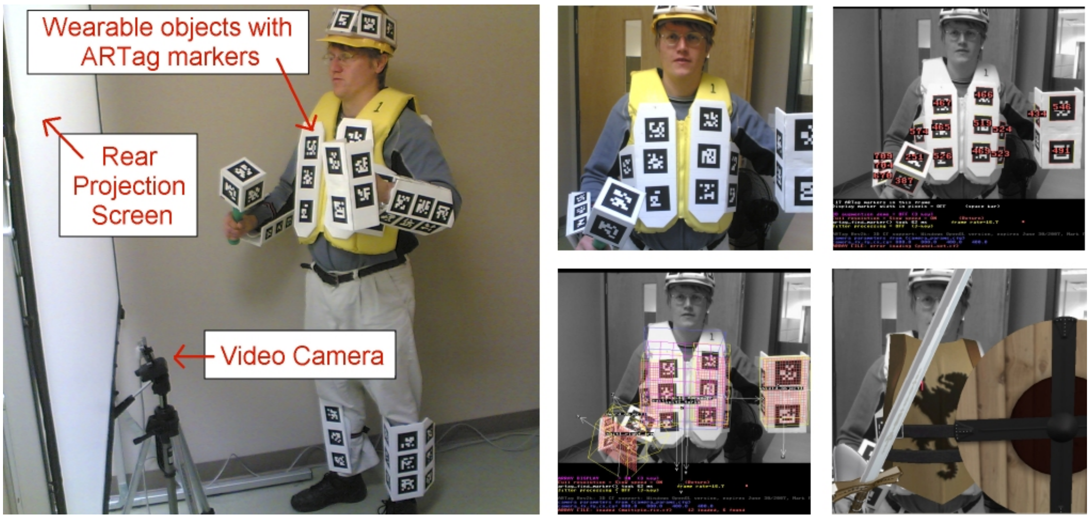


Figure 2.19: ARTag Magic Mirror[4]

Chapter 3

Related Work

This chapter discusses relevant topics of research and identifies concepts that are applicable to this project. Such applicable fields include registration, image processing and image segmentation. This chapter is not concerned with processes that would take place after the execution of the proposed technique. Concepts that would make use of output data* are discussed within the appendices. Lighting and shading calculations that take illuminant positions as input data are discussed in appendix B and shadowing and relighting techniques are discussed in appendix C. This chapter focusses on other research and methods that facilitate the success of the interest point based photometric registration technique proposed in chapter 4. Geometric and photometric registration is discussed in section 3.1.1 and 3.1.2 respectively, image processing techniques are discussed in section 3.2 and image segmentation is discussed in section 3.3. The photometric research discussed in section 3.1.2 aims to achieve similar goals to this research project, except by different means.

3.1 World Registration

The field of world registration can be divided into the two main categories of geometric registration and photometric registration. Geometric registration techniques deal with the world alignment and pose estimation problem. They strive to ensure that virtual augmentations line up with real-world geometry. Photometric registration methods attempt to detect real-world illumination conditions so that an AR application can duplicate lighting, shading and shadow characteristics and apply them to the virtual world component. As the user is able to directly compare the two worlds simultaneously, the synthesized component should not

*Illuminant metrics as discussed in chapter 4 only be visually detailed but also correctly registered to ensure seamless scene augmentation and high believability.

3.1.1 World Alignment & Geometric Registration

Detailed research into the process of geometric registration for augmented reality has been undertaken and the methods by which real and virtual worlds are accurately aligned in real-time have been explored. Both visual and sensor based registration approaches are commonly used for AR purposes and often multiple techniques are combined to form hybrid solutions. Many visual approaches require camera calibration to take place before they may function. Camera calibration is the calculation by which the extrinsic and intrinsic camera properties are obtained given the relationship between 2D pixels and 3D locations. The camera calibration process is discussed in more detail in appendix A.

Sensor Based Techniques

A number of techniques use information obtained by devices such as the sensors discussed in section 2.1.3 to aid the visual tracking process. One such technique was presented by Behringer[9] which uses the GPS network to determine the position of the camera. Behringer uses outdoor horizon silhouettes to assist the registration process. The technique works by tracking visual features whose real world positions are known. The camera is initially located via GPS, then terrain data is gathered from digital elevation maps for the camera location. Providing that the terrain is sufficiently well structured the horizon extrema can be evaluated visually and the camera orientation can be calculated. The horizon outline is extracted using the Sobel operator as discussed in section 3.2.2. The horizon outline is then used to determine camera pose. This technique would fail in flat areas where the horizon contains too few dips and peaks to obtain any useful information. As such it may be more useful as part of a more involved hybrid system than as a stand-alone pose estimation technique. GPS based systems only operate when sufficient signal from four or more satellites can be obtained. GPS signals become sporadic in built up areas and can be fully occluded when the receiver is indoors. GPS information is less reliable in such areas.

Sensors have also been used to measure the orientation of AR devices. There are two approaches to tracking the real world visually. The first requires the environment to be prepared with fiducial markers which identify key areas and are used as reference points. These can then be tracked, and software is able to calculate the camera pose by estimating the marker's posture. Accurate superimposition can take place once such information has been gathered. Substantial initialization effort is often required when using certain vision-based techniques to achieve accurate registration. Other approaches involve locating interesting features within the image and then tracking them, instead of fiducial markers. Many systems have been developed that rely on placing markers at strategic points within the environment before registration can occur, some of which are discussed by Hoff[39], Koller[52] and Kato[47].

Visual techniques

Marker based systems are commonly used due to the complexities associated with markerless techniques however such use disturbs scene believability. Markerless approaches are such as presented by Gordon[28][29] attempt to detect unique natural image features, as discussed in section 3.2.2, instead of artificial markers. In this technique instead of markers being used to assist tracking, stable natural features are extracted from an image using the SIFT algorithm which is also discussed in section 3.2.2. These SIFT points are used as descriptors of local image patches. The features are invariant to image scaling and rotation and are also partially invariant to translation and changes in viewpoint. Multi-view correspondences are then used to create a metric model of the real world and the system learns scene geometry. Models are then recognized and tracked. As a camera moves, its pose is calculated in relation to these models. Gordon[29] presents a fully automated system architecture for markerless augmented reality. The system performs model-based augmentation and results in robust tracking in the presence of occlusions and scene changes using highly distinctive natural features to establish image correspondences. The only preparation required is a set of reference photos taken with an uncalibrated camera. Cornelis[13] presents completely markerless techniques whereby the virtual scene is registered using the results of global bundle adjustment and camera self-calibration. Devarajan[17] provides a detailed explanation of this concept. Cornelis provides mitigation for the augmentation jitter problem but is not robust to occlusion and changes in illumination. Simon[88] presents a method of using the planar surfaces that exist within outdoor environments in order to estimate the pose of the camera. It is able to track multiple planes simultaneously in a computationally simplistic manner compared to model tracking approaches. Rosten[83] uses visual edges and points using methodology that allows for fast camera motions. The technique employs the FAST feature detector in order to perform full-frame feature detection at speeds of 400hz. Fast feature extraction such as this provides robustness to camera translation and rotation and can even cope with 50° rotational shakes at 6hz. A pose estimation method that makes use of SIFT is outlined by Gordon[29]. Other techniques such as the Features From Accelerated Segment Test (FAST) algorithm and the Kanade-Lucas-Tomasi (KLT) tracker are also frequently used. The latter is primarily used within the robotics field of Systematic Localization and Mapping (SLAM). When tracking camera pose by visual means, accuracy varies in proportion to the range of objects within the image.

Lee[57] presents a method that reduces the number of pre-calibrated entities required. The technique requires knowledge of camera poses relating to three or more reference images and uses an omnidirectional camera to track motion in 6 degrees of freedom (DOF). This technique does not require a database of environment images as many other techniques do. The system tracks to 5DOF and estimates camera pose requiring only 2D to 2D correspondences. A 6DOF camera pose is then derived directly from two 5DOF motion estimates between two reference images and an image from the tracked camera. Two techniques that enable estimation functionality are the Least Squares technique (LS) and the Unscented Kalman Filter technique (UKF). Systems that make use of databases of environment images include those presented by Behringer[9], Thompson[92] and Coors[12]. The technique presented by Coors makes use of images within the environment database as reference points and compares them with the real images to derive camera rotational and translational information. Registration methods exist that estimate the camera pose by visually matching image features with those of a 3D model. High levels of success in this area have been achieved by Reitmayr[82]. Model-based tracking relies on the detection of appropriate features within both images and textured 3D models. Such features could be points, edges or corners. Davison[15] has also presented methods of generating this model on the fly.

Hybrid Techniques

Kim[49] presents a hybrid system that makes use of aerial and frontal views in combination with sensor data to dynamically generate models and datum that are used for registration purposes. To realize convincing augmentation in natural environments the camera pose must be accurately estimated in real-time, even when the environment has not been specifically prepared. Reitmayr[82] uses a model and gyro based hybrid tracking system for outdoor augmented reality. This system makes use of textured models of the world to improve correspondence match accuracy. This hybrid registration method uses a novel edge-based tracker for accurate localization, and gyroscopic measurements to deal with fast motions. Magnetic field and gravitational measurements are used to avoid drift and a back store of frame information is saved to mitigate the effect of occlusion. Ahn[2] presents a technique originally intended for the field of robotics which detects unique features using the Harris[33] and Scale Invariant Feature Transform (SIFT)[61] feature extraction algorithms. Figure 3.1 shows an example such features extracted in preparation for registration.

The hybrid registration method presented by Hirose[38] utilizes edges and vertices of a 3D model of the target object. When this object comes into view, the camera position and orientation are estimated by detecting the vertices and true edges every frame. Multiple edge candidates are considered and the most

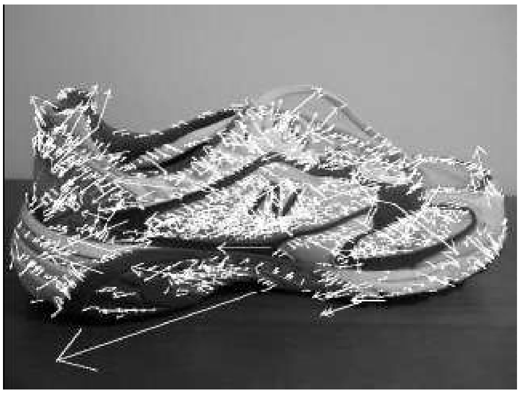


Figure 3.1: SIFT Extracted Features [29]

suitable is used in order to reduce the influence of misleading edges. Either a magnetic sensor or artificial visual markers are used in combination. This allows the system to obtain the approximate camera position and orientation when the target object goes out of view or if the camera moves too fast to detect natural objects. By using such hybrid techniques the accuracy and robustness is increased, especially during times of rapid camera movement. Model based techniques only work when an object within the environment is known in advance. Using markers alongside natural features allows for fast and stable pose estimation. However substantial preparation time is needed to deploy markers and calibrate them. Calibration involves measuring the size of and the spacing between each marker. Mis-correspondence and mis-tracking causes decrease in the accuracy of the estimation. However using a feature based approach the original scenery is left intact and augmentation can take place anywhere. Model based systems such as that presented by Lepetit[59] provide higher registration accuracy. However it is difficult to construct a 3D model of *everything* within the environment that could be tracked, therefore some researchers and developers prefer to model just some of these objects and consider the implementation of a hybrid system. Hybrid techniques allow a system to take the advantages from each technique used while mitigating any weak areas. Hirose[38] uses cameras that have pre-estimated intrinsic parameters as devices to capture image sequences and a 3D object with known shape, such as a house or a box, is placed within the environment. The technique provides the option of using either magnetic sensors or artificial markers to increase robustness. This technique uses the KLT tracker which tracks features as shown in figure 3.2.

Kim[49] draws on SLAM techniques and utilizes GPS and inertial data, aerial photography and frontal imagery. SLAM based techniques are able to dynamically map the surroundings as they are seen. The aerial and frontal images

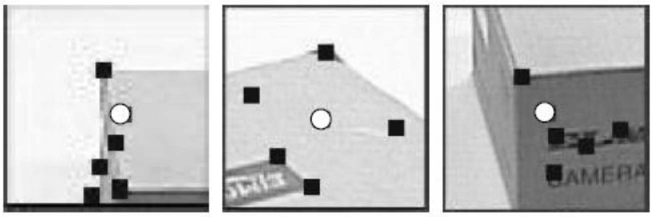


Figure 3.2: The KLT Feature Tracker [38]

are used to visually generate models of buildings with sufficient detail for tracking purposes. This eliminates the need to manually create 3D models prior to augmentation.

3.1.2 Illuminant Detection & Photometric Registration

Literature shows that researchers have attempted to photometrically register augmented reality worlds in a number of ways, calling upon a variety of machine vision and image processing techniques. Most existing techniques work well providing that the environment is heavily constrained but fail if certain conditions are not met. A small number of techniques operate well in less constrained environments, however their computational complexity is high and therefore they are unfeasible for real-time augmented reality processing or they may cause disruption to AR realism. This section provides an overview of existing photometric registration techniques and outlines relevant low level functionality.

Early photometric registration techniques were able to detect the azimuth of an illuminant from image intensity information. Zheng[105] presents two methods of estimating the azimuth of a single illuminant. The first proposed method makes use of *local voting.* This method assumes that for any point $(x_o, y_o, z_o(x_o, y_o))$ its neighbors can be locally approximated using a spherical patch where $(a(x_o, y_o), b(x_o, y_o), c(x_o, y_o))$ is the centre of the sphere[†] and $r(x_o, y_o)$ is the sphere radius. This technique is able to derive the direction of the illuminant but not the absolute position, or depth on which it lies. It is therefore only partially useful for photometric registration purposes. Without depth information shadows and shading can not be correctly simulated. The technique relies on prior or obtainable knowledge of scene plane normals in order to execute correctly.

Mukaigawa[69] suggests a photometric image-based rendering approach that aims to estimate lighting directions without needing to virtually reconstruct any part of the scene geometry, whereas this is required with model-based approaches. This approach decomposes complex input images into linear and non-linear fac-

[†] The sphere is a local approximation whose radius and centre depend on the local surface shape

tors. The linear factors cover diffuse reflections and obey the Lambertian reflectance model, an image with any light direction can therefore be synthesized by performing linear combination of three images[69]. Non-linear factors consist of specular reflections and shadows. Mukaigawa processes these separately. As per the dichromatic reflection model a reflection is classified into diffuse and specular reflections[50] as shown in figure 3.3.

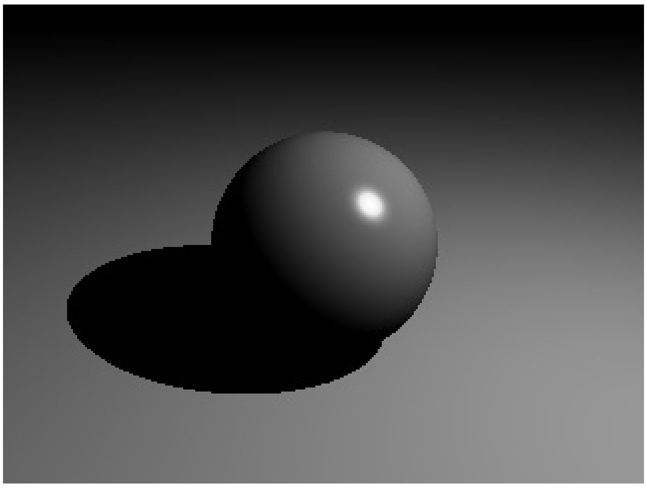


Figure 3.3: Object Reflectivity

A diffuse reflection is equally observed from every direction and does not depend on the viewing angle whereas the specular reflection is intensely observed from the mirror of the incident angle. Mukaigawa states that shadows are classified as either self or cast shadows. A self shadow is the dark region on the object that is not directly lit by the illuminant. A cast shadow is the shadow that is cast on to a nearby surface due to the occlusion of the light source. The self shadow therefore depends on the relationship between the surface normal and lighting direction. It can be observed where the surface faces away from the light source. The cast shadow depends on both the lighting direction and the 3D shape of the scene. The basic Lambertian reflection model is given as:

$$i = (ls) \cdot (rn) \tag{3.1}$$

l is a lighting power, s is a unit vector representing the light direction, r is a diffuse reflectance and n is the unit vector surface normal. This can be simplified to:

$$i = S \cdot N \tag{3.2}$$

In this equation S denotes the lighting property and N denotes the surface property. As real scenes are more complex, Mukaigawa also considers specular reflectivity and ambient environmental illumination[‡]. Shadow regions are also taken into account. As such an image is formulated as follows:

$$i = \alpha(i_D + i_s) + i_E, \alpha = \begin{cases} \text{0- light is occluded} \\ \text{1- light not occluded} \end{cases} \tag{3.3}$$

Here, $i_D = S \cdot N$, forms the diffuse reflection factor, i_s is the specular reflection factor and i_E is the environmental illumination factor. To perform photometric registration, Mukaigawa formulates rendered pixel intensities as a sum of these three factors[§]. When synthesizing image output it is possible to use a simple algorithm to shade pixels. Mukaigawa uses principal component analysis[50] and converts a real image that satisfies equation (3.3) into an imaginary image which satisfies equation (3.2). Mukaigawa first eliminates the constant ambient lighting by subtracting a pre-acquired background image from each input. This action is taken because the environmental lighting is not an effect of the local illuminant. In order to estimate lighting direction, Mukaigawa then attempts to obtain the lighting property. Mukaigawa does this by directly measuring it when the images are taken. The vector obtained often contains error[69]. The lighting property vectors are calculated, assuming Lambertian surface, from three base images. Mukaigawa decomposes the images until the above equations are satisfied. Once all parameters are obtained, then the information can be used to relight a scene or used when synthesizing output pixels. This method does not determine the absolute location of an illuminant, and is not suitable for scenes containing complex geometry. The accurate recreation of shadows would not be possible with this technique as only the shadow direction is obtained and not the 3D coordinates at which it is positioned.

Sato[84] describes a method for estimating illumination distribution by observing a radiance distribution inside shadows cast by scene objects. The illumination distribution of the scene is estimated form the radiance distribution within shadows cast by an object of known shape onto some other object. The reflectance properties of the object are estimated at the same time as the illumination distribution using an iterative optimization framework. The technique does not attempt to locate the actual illuminants that are present within the scene. Sato first initializes the reflectance parameters of the shadow surface, with the assumption that it is Lambertian. The diffuse parameter K_d is set to be the pixel value of the brightest point within the shadow region. The specular parameters are zeroed, $(K_s = 0, \sigma = 0)$. Radiance values, $L(\theta_i, \phi_i)$, of imaginary directional light sources that model the illumination distribution of the scene are estimated. This is calculated using the reflectance parameters and image brightness. Sampling is repeated and sampling directions are increased until sufficient

[‡] An even illumination of the scene

[§] Inter-reflections are not considered accuracy is achieved. This technique assumes distant light sources that create directional light which projects parallel rays onto the surfaces of objects. The technique makes a number of assumptions that are not true of most real world scenes. Basso[8] presents a method whereby intensities caused by existing light sources are modified to improve illumination conditions within video conference applications. It provides scene improvement by correcting poorly lit areas in the same manner as existing lighting. It lights 2D images by placing 3D lights. This requires polygonal reconstruction of the scene which in turn requires a calibrated camera. Real lights are virtually modeled by analyzing the scene and comparing them to controlled images and specular highlights are observed. This technique is not computationally complex, however it is constrained in that it requires a static camera and does not attempt to locate illuminant 3D coordinates. Kanbara[46] proposes a combined geometric and photometric registration method that utilizes a fiducial marker for calibration and world alignment purposes, and a mirror ball to facilitate illumination analysis. The marker is detected using the method presented by Kato[47]. Once located, geometric registration takes place and the mirror ball is identified. The brightness of real world illuminants is determined by the intensities of pixels in the mirror ball region. The directions of light sources are estimated by using the camera pose and surface normals of mirror ball points at the detected illuminant pixels. Kanbara[46] claims to have improved upon this concept to produce composite images from real and virtual images at almost full video framerate, maintaining correct cast and self-shadow consistency.

Wang[97] introduces a method for detecting and estimating the location of multiple directional illuminants using a single image containing an object with known geometry and Lambertian reflectance. This technique claims high accuracy but does not operate in real time. It is not suitable for real world augmentation as prior knowledge of scene geometry can not be guaranteed. Wang claims that natural scene geometry is suitable, however it is still constrained to pre-determined environments as knowledge of scene geometry is required. Wang attempts to use shadows and shading independently to obtain the illuminant direction. Wang integrates the results from the shadow and shading methods in order to improve accuracy. Recursive least-squares algorithms[35] within the shading-based calculation are used. Patches are detected that correspond to each light present and associated directions are calculated from four points on each patch. Where A, B, C and D are points within a patch relating to an individual light source and n_A, n_B, n_C and n_D are their normals. Given the Lambertian equation (3.1), augmented by ambient light, α, the below is stated:

$$\begin{bmatrix} n_{\mathrm{Ax}} & n_{\mathrm{Ay}} & n_{\mathrm{Az}} & 1 \\ n_{\mathrm{Bx}} & n_{\mathrm{By}} & n_{\mathrm{Bz}} & 1 \\ n_{\mathrm{Cx}} & n_{\mathrm{Cy}} & n_{\mathrm{Cz}} & 1 \\ n_{\mathrm{Dx}} & n_{\mathrm{Dy}} & n_{\mathrm{Dz}} & 1 \end{bmatrix} \cdot \begin{bmatrix} L_x \\ L_y \\ L_z \\ \alpha \end{bmatrix} = \begin{bmatrix} I_A \\ I_B \\ I_C \\ I_D \end{bmatrix} \tag{3.4}$$

I_A, I_B, I_C and I_D represent the intensity of pixels at four respective points, A, B, C and D. Wang[98] explains that if n_A, n_B, n_C and n_D are non-coplanar the direction of the corresponding light source, L, $[L_x, L_y, L_z]^T$ and the ambient light, α can be obtained through equation (3.4). If the scene contains shadow information then the illumination can be recovered from radiance distribution within shadow regions. Region boundaries are determined using the Hough transform on detected critical points. These regions are tested with shadow detection techniques and are evaluated before results are integrated. This method does not require the use of a pre-calibration object. Additionally the data collected from the technique allows for the virtual recreation of three dimensional object shapes. The illuminant detection results that this technique yields are directly applicable to the development of realistic augmented reality systems, however the calculations required are slow and therefore would not be capable of processing a live video stream in real-time. Wang's technique provides good results compared to a number of other techniques as it analyzes both shadows and the shading of arbitrary scene objects. The technique finds it easy to obtain multiple illuminant information from shading when specular reflections are present but finds the task difficult when observing diffuse reflections alone. This technique adds robustness as it is less prone to error caused by cast shadows moving outside the camera's field of view, or being occluded by techniques that observe either object shading or cast-shadows exclusively. Wang makes use of binocular vision for depth calculations. Agusanto[1] presents a technique that acquires scene radiance through high dynamic range (HDR) photography. Radiance is obtained through use of a light probe with a wide angle or omnidirectional visual sensor. An illumination map is then created from the obtained radiance map. This process involves creating a virtual environment containing a box. The radiance map is textured to this box, and a sphere is placed within it. The ratio of the size of the sphere and box should be between 1:50 to 1:500. The material property of the sphere is then set to diffuse or glossy. The illumination map is synthesized by running global illumination calculations using ray tracing. The faces of the box act as light sources depending on the radiance map. Illuminant directions can be estimated from the resulting sphere render. Zhou[107] presents a calibration sphere[106] based method that can cope with multiple types of light source. The calibration sphere must have a specular surface and the camera must be well calibrated. Once the calibration sphere is located within an image specular patches are segmented from Lambertian intensities. Figure 3.4 shows a calibration sphere with specular reflections that facilitate the determination of illuminant direction. Two spheres may be used to achieve absolute location estimation.

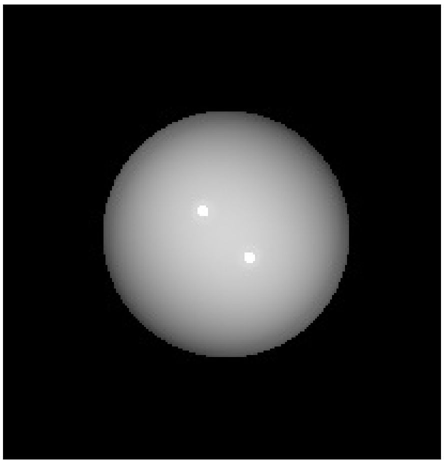


Figure 3.4: Photometric Calibration Using Specular Highlights[107]

The source region, A, is defined as a segment of plane, $\wp$. Once this plane is determined the source segment can be estimated by intersecting retraced rays with it. A plane can be determined for each light source and each plane, $\wp_i$, is defined in 3D space as:

$$(X - X_{\mathrm{pi}}) \cdot N_{\mathrm{pi}} = 0 \tag{3.5}$$

Where X_{pi} is a point on the plane, $\wp_i$, and N_{pi} is the normal vector of the plane. Multiple intensities are estimated and ray intersections calculated for each light source that is detected. Feng[22] suggests a technique that makes use of two spheres with Lambert surfaces as artificial calibration objects in order to gather illumination parameters. The author claims to achieve an identical match between real and virtual components, the result being a seamless augmented reality scene. Calibration spheres are covered with lusterless paint in order to achieve the desired diffuse surface. This technique operates in real-time with relatively low operational complexity but fails if multiple real light sources are present. The technique is able to retrieve point light intensity and position and also ambient intensity. The technique is not suited for combination with any geometric registration approach as a stationary camera is required once pre-calibration has taken place. Feng does not observe or attempt to reproduce cast shadows as absolute illuminant position is not detected. The illuminant position is calculated from two direction vectors obtained from the two calibration spheres. The marker spheres have a Lambert surface and constant albedo. Therefore the BRDF $f(\theta_i, \phi; \theta_e, \phi)$ is constant and each surface point is equally bright from all viewing directions, V. The below equation describes the illuminance of any point on the spheres by the equation:

$$I = I_{\text{ad}} + I_{\text{ld}} \tag{3.6}$$

$$= k_a I_a + k_d (I_d \cdot \frac{1}{d+c}) \cos(L, N) \tag{3.7}$$

Where I_a is the intensity of the ambient light source and I_d is the intensity of the point light source. The values k_a and k_d are the diffuse reflectance to both lighting types. I represents the intensity of reflecting light. N and L are the surface normal and illumination direction vectors respectively. Equations (3.6) and (3.7) consider attenuation due to distance, where d is the distance between the point on the sphere and the illuminant. The value c is an undetermined coefficient. The greatest point of illumination on any sphere is the point, p, whereby the sphere surface normal is parallel to the illuminant direction vector. This is a point at which the angle between N and L is zero. The illuminance at p is given as:

$$I_p = k_a I_a + k_d (I_d \cdot \frac{1}{d+c}) \geq k_a I_a + k_d (I_d \cdot \frac{1}{d+c} \cos(L, N) \tag{3.8}$$

This technique is illustrated in figure 3.5. The center of the spheres are O_1 and O_2 and the points of highest illumination are P_1 and P_2 respectively.

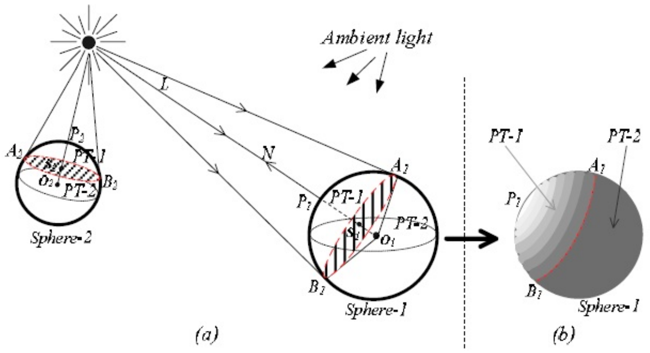


Figure 3.5: Calibration Spheres[22]

A ray cast between each center point, O, and corresponding point of highest intensity, P would pass through the illuminant. The intersection of these rays is the illuminant position. When geometrically registered, this technique does acquire 3D illumination coordinates, including depth. As this technique requires two artificial calibration objects to be present at all times overall scene realism is disrupted. The method by which Wang visually locates spheres and and locates illuminated regions requires high computational complexity and would likely operate slowly, although Wang does not publish operational speed metrics. Pessoa[75] presents an image based lighting approach to global illumination and BRDF solution to photo-realistic augmented reality. This technique deals with the recreation of realistic lighting effects by dynamically generating environment maps. Ma[63] describes a recently proposed method by which multiple illuminant directions are obtained from a single image. The method uses a square fiducial marker for geometric registration purposes. A mirror sphere with known size is used for illumination detection. Highlight pixels within the sphere are analyzed using a c-means clustering algorithm and its initialization with the max-min distance method. This technique makes use of a self correction algorithm. Figure 1.5 shows such a marker and sphere setup. This technique does not establish the absolute location of any illuminant. Table 3.1 shows a summarized comparison of the most significant photometric registration techniques.

Technique	Advantages	Disadvantages
Zheng [105]	Finds direction	Requires planar surfaces Only detects azimuth
Mukaigawa [69]	Finds direction	Only detects azimuth Result often contains error
Sato [84]	Finds direction	High complexity / slow Makes assumptions not true about most scenes
Basso [8]	Relight 2D scenes Low complexity	Not suitable for AR Only detects azimuth Requires static camera
Kanbara [46]	Finds direction Low complexity	Requires artificial scene features
Wang [97]	Absolute location Multi-illuminant	Poor results when no visible specular

Table 3.1: Comparison of significant photometric registration techniques

3.2 Image Processing

A number of image processing techniques facilitate the easy extraction of features required for AR processing. The functionality that is directly related to this project include low level processing to prepare the images for processing at a higher level, feature localization and extraction and also correspondence detection. Such techniques are discussed within this section.

3.2.1 Image Preparation

Input imagery is sometimes unsuitable for further processing, or is not compatible with a specific technique. In many cases it is possible to alter the image in order to correct this problem. Images that contain noise present analysis difficulties, as such this noise should first be mitigated. Techniques to reduce visual noise include low/high pass filtering in the frequency domain and blur techniques. Blurring can be achieved by performing a convolution with a weighted kernel in the spatial domain or by performing a multiplication within the frequency domain. The Fourier transform and inverse Fourier transform are used to convert between spatial and frequency image representations[94]. A number of spatial blurring techniques are commonly used including the box and gaussian blur techniques. The box blur technique can be implemented with a simple, fast algorithm. It works by iterating through each pixel and making its value equal to that of the average of its neighbors. All neighbors are weighted evenly. This is the equivalent of performing a convolution with the kernel as shown in table 3.2.1, which is relatively fast but is less accurate than a Gaussian blur which priorities the weighting of neighbors as per a Gaussian function. The Gaussian distribution of the kernel can be calculated with the following equation:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \tag{3.9}$$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Table 3.2: Box Blur Convolution Kernel

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Table 3.3: Gaussian Convolution Kernel

This yields an integer valued convolution kernel similar to as shown in table 3.2.1. The normalized values are used to calculate the average pixel intensities resulting in a more accurate blur that reduces noise whilst mitigating the destruction of scene detail[64].

3.2.2 Feature extraction

Interest points assist with the detection of more complex geometry and world conditions. An interest point is defined as a two-dimensional signal change; for example, where there is a corner, an edge or where the texture changes significantly[27]. Much work has been undertaken in the field of interest point detection and feature detection techniques. A number of these are able to extract edges and corners of interest. An image can be reduced to pixel-wide edges that represent geometric boundaries of objects within a scene as seen in figure 3.6.

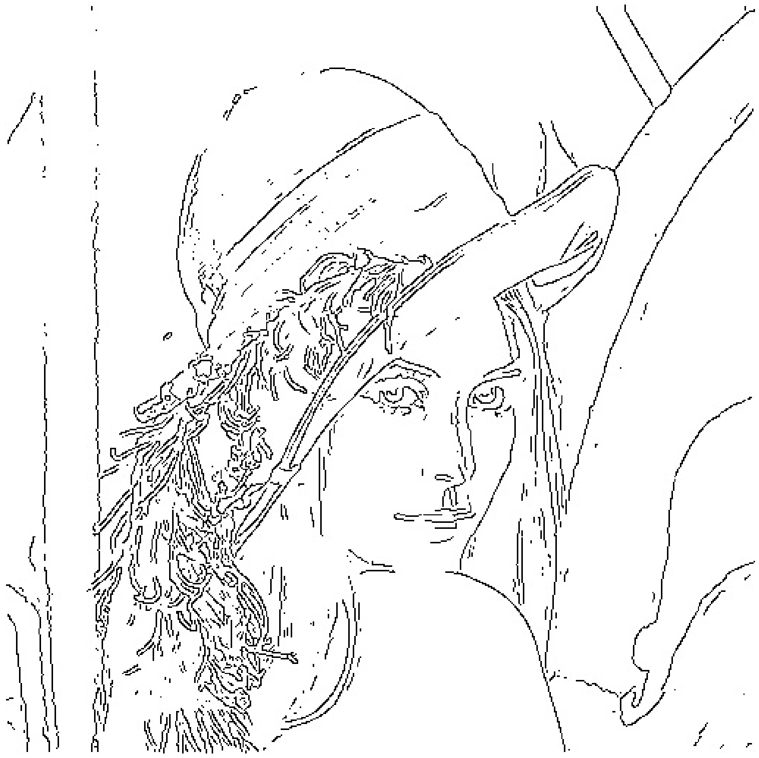


Figure 3.6: Canny Edge Detection

Most autonomous edge detectors are comprised of three stages:

- Smoothing
- Differentiation
- Labeling

A large number of methods for achieving edge detection exist, however they may be grouped into two categories. These categories are search or zero-crossing based. Search methods first compute a edge strength measurement. This is usually a first-order derivative expression such as the gradient magnitude. They then attempt to discover the local directional maxima of the gradient magnitude using an estimate of the local orientation of edge in question. This is usually the gradient direction. Prior to edge detection, a smoothing stage is used. This is typically Gaussian based smoothing.

Edge detection methods generally differ in the filters applied and also calculation that determines the strength of edges. As many edge detection methods rely on the computation of image gradients, they may differ in the techniques used for computing gradient estimates in the x- and y- directions. One edge detection method is that presented by Canny[11]. The Canny edge detector is a multistage technique containing the following stages:

1. Gaussian blur
2. Find image gradient
3. Determine angle of edges
4. Round angles
5. Non-maximum suppression

The detector first smooths the input image to reduce noise, ensuring that no one noisy pixel will interfere with the result. Four filters are then applied to the image to detect vertical, horizontal and diagonal edges. The filters used include the Roberts, Prewitt and Sobel edge detection operators[3]. The gradient at any image coordinate, $[x, y]$, can be found as follows:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.10}$$

The direction of a particular edge is then found:

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.11}$$

Where Θ is the angle, which is then rounded to 0, 45, 90 and 135 degrees in order to represent the four possible directions when dealing with pixels. In order to ensure that the detected edges are one pixel in thickness non-maximum suppression is used. This is the process of giving zero intensity to pixels that are not located at the peak of each gradient. Pixels that are located at the peak are given maximum intensity.

Corners can be detected by observing the intersection of two edges that have significantly different directions. The Harris and Stephens corner detector is able to detect corners within grayscale images[33]. This corner detector can be computationally intense when operating on large image patches. Smallest univalue segment assimilating nucleus (SUSAN) and features from accelerated segment test (FAST) are alternative feature detection techniques. The most robust approach is currently the scale invariant feature transform (SIFT) as presented by Lowe[61]. There are four main stages to the SIFT feature extraction technique[26]:

- Scale-space extrema detection
- Key-point localization
- Orientation assignment
- Key-point descriptor

A gaussian-based cascade filter approach is used to detect potential key-points. These are then compared across different scale-spaces, as discussed by Witkin[101], to determine which are invariant to changes in scale. Only stable key-points which are robust to changes in scale are used. Points that are poorly located on an edge or areas of low contrast are filtered out by applying a threshold to the ratio of principle curves both across and perpendicular to an edge, poor edge points are rejected. Each key-point is then assigned a consistent orientation in order to achieve rotational invariance. This is derived from an orientation histogram which is formed from the orientation of gradients at each key-point. It is possible to acquire multiple orientations at a key-point, for example where a corner exists or where two edges cross. A descriptor is then formed in the form of a vector that is based on orientation histograms at neighboring locations immediately surrounding a key-point. The descriptor is then normalized in order to reduce susceptibility to global contrast and brightness changes. This resulting descriptor is robust to scale, rotation, contrast and brightness and performs well relative to other feature extraction techniques. Figure 3.1 shows the result of performing the SIFT operation on an image.

3.2.3 Correspondence detection

For a number of years researchers have been interested in detecting correspondence between images and image segments. The field of autonomous mosaicing has been main drive of correspondence detection research. This method is generally used prior to the automatic stitching of panoramic imagery. Correspondence detection techniques aim to detect both the corresponding points and the transformation between the pairs. Some techniques aim to compute eigenimage features[¶] using principal component analysis (PCA) for finding corresponding areas[26][104]. Wavelet-based edge-preserving approaches are also used to find image correspondences[6] however these are not robust to changes in viewpoint.

[¶] A set of eigenvectors used for recognition

Gledhill[26] shows that the SIFT algorithm has proven to be robust as a correspondence search and matching method for both panoramic imaging and object recognition. Therefore it can be assumed that the above techniques and methodology would be capable of providing a basis for shadow and object interest point matching as discussed in chapter 4.

3.3 Image Segmentation

The field of machine vision strives to autonomously identify objects and significant scene geometry, and much has been accomplished with the detection and segmenting of shadows and objects within both still images and video footage. A number of techniques exist that allow for the detection of shadows and object regions within a scene. Such techniques are discussed within this section. This project considers shadow and object segmentation techniques that are useful within the augmented reality process and would be useful when photometrically registering the real and virtual worlds.

3.3.1 Shadow Segmentation

Shadow segmentation techniques have been developed for a number of computer vision purposes including the improvement of vehicle detection when monitoring the flow of road traffic[79], and to assist with object segmentation and discrimination[86]. By segmenting and subtracting shadows it is possible to eliminate them when performing object processing, reducing the likelihood that multiple objects or moving blobs merge and are poorly detected. Yao[102] presents a shadow segmentation method that operates on colour images and creates an undirected graph that models the image. Shadow detection is achieved by maximizing the graph using the EM algorithm[16]. Martel-Brisson[65] presents a shadow detection method that is targeted at surveillance applications. The technique is a pixel-based statistical approach that models moving cast shadows of non-uniform and varying intensity. It makes use of the Gaussian mixture model learning ability in order to build a statistical model that describes moving shadows that are cast onto surfaces. Leone[58] and Joshi[44] both present solutions that detect moving shadows. Leone makes use of texture patches and shadow texture characteristics to determine whether a shadow is present. Joshi makes use of background segmentation and low/mid-level reasoning to detect shadow regions. Shadow detection techniques have become robust in recent years and are frequently used in surveillance and monitoring applications. Current state-of-the-art techniques are suitable for use in enhancement of the proposed technique.

3.3.2 Object Segmentation

Object segmentation techniques are widely used in the field of computer vision and with robotics. Shapes that can be recognized range from simple primitives to complex geometry through various different approaches. The simplest techniques involve shape matching. One such technique was presented by Pao[72], where the straight line Hough transform (SLHT) is used. This technique allows for the easy decoupling of the translation, rotation and intrinsic parameters of a curve detected within an image. A scalable translation invariant rotation-to-shifting signature is calculated and detection takes place by performing a 1D correlation calculation. This method of detecting simple primitives is used by more complex techniques. Methods that are able to detect the presence of and segment more complicated geometry include that presented by Heisele[36] which is a supervised learning approach to the classification and identification of scene objects. Heisele uses a set of training images and determines relationships using statistical learning theory. Other approaches use Haar like features to achieve object recognition[60]. Techniques are able to segment foreground objects when given a priori background image or by detecting moving blobs. The technique presented by Kosir[54] detects predetermined geometric shapes by making use of pattern spectrum characteristics within digital imagery. Mutch and Lowe[70] present a method that is able to recognize multiple object types in natural images by modeling the human visual cortex to achieve feature sparsification, lateral inhibition and feature localization. This technique makes use of a machine learning method that mimics the way neurons respond to visual stimulus. Once an object location is detected its boundaries can be located using an edge detection technique as discussed in section 3.2.2. As object segmentation is robust to known scene geometry and partially robust to unknown geometry it is feasible to use the above techniques within implementations of the proposed photometric registration technique. Limitations within any segmentation technique would however impact on the ability of the proposed technique to correctly classify interest points.

3.4 Summary

The techniques discussed in this chapter contribute to the realism of augmented reality worlds in some way. No current photometric registration technique is able to deliver believable results under all circumstances and many are able to detect the illuminant azimuth and not the absolute position. Some techniques improve one aspect of realism only to reduce it elsewhere; for example techniques that require artificial calibration objects or place constraints on the operational environment. Any AR photometric registration technique that requires that artificial components be present is in fact reducing realism and is therefore a failure.

Any constraint places on the environment reduces either realism or applicability for AR to some degree whereas techniques that are too complex are slow and completely unsuitable for real-time AR. The issues discussed have fueled much motivation behind this research; it has been observed that by making use of the natural features already present in the majority of scenes it is possible to locate the absolute illuminant position without requiring algorithms of high complexity. Although this research accepts that some assumptions are required, any such assumption should have as little impact on overall realism as possible. Factors such as technique complexity, applicability to AR, overall realism and the avoidance of environmental constraint were considerations that played a key part in the design of the novel approach presented in the following chapter.

Chapter 4

Illuminant Tracking Technique

4.1 Technique Overview

The aim of the proposed technique is to detect the 3D position of an illuminant by observing visible cast-shadows and scene objects. The suggested approach assumes input scenes contain one or more object and a single illuminant that is causing a cast shadow. Two or more data sets are required in the form of still image files, video files or live video feeds. This data forms the real component of the augmented reality world. By using natural shadow features the technique is able to minimize disruption to scene realism. Metrics gathered by performing segmentation and interest point detection provide information that is used to derive the position of the illuminant. The presence of shadow and object regions is required for the technique to function. Detected interest points are classified as belonging to such regions. It is assumed that existing techniques are able to perform this task. Correspondences between the shadow and object interest points are then to be detected. The illuminant position is then found in 2D, for each input image, of which a minimum of two are required. Once geometric registration has occurred, the 3D location is then found using rays obtained by reverse projecting each 2D illuminant coordinate. Once the 3D coordinates are obtained, it is possible to generate an augmented reality scene whose virtual light conditions mimic that of the real environment. Operational complexity is low and therefore photometric registration occurs in real-time. The results obtained from two input images are of sufficient accuracy for realistic augmentation as discussed in detail in chapter 6. If more input images are used, the overall accuracy may be improved. In an augmented reality application, this technique typically captures input imagery from two real-world input devices simultaneously. However three dimensional renders, such as shown in figure 4.1, can also be used.

These images can be acquired from almost any angle, so long as sufficient

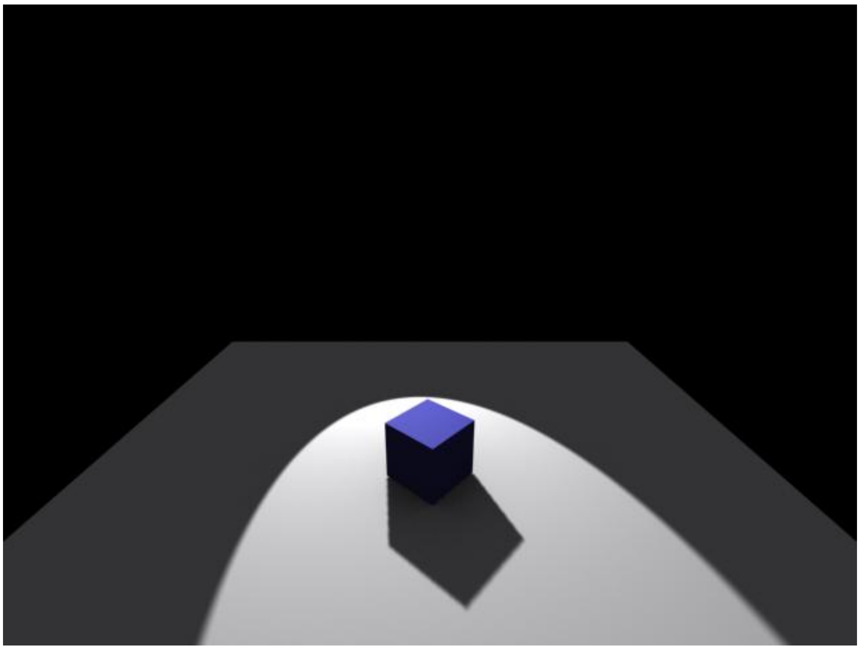


Figure 4.1: Synthetic Input Imagery

object geometry and shadow edges are visible. The prototype system can automatically detect the angle between cameras providing that sufficient geometric registrational information is present within both images. This information is potentially provided by a marker or object of known geometry.

A certain amount of computation is required for each input image, therefore the more input images used the more operationally complex the function becomes. This causes a tradeoff between accuracy and the resource requirements of the technique. The illuminant information is provided by interest points within the input images during the photometric registration stage of the reality augmentation process shown in figure 4.2.

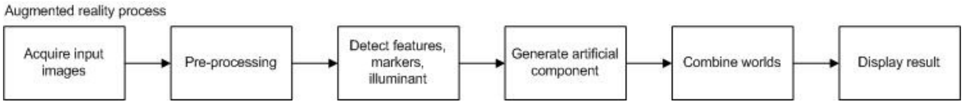

Augmented reality process
Acquire input images
Pre-processing
Detect features, markers, illuminant
Generate artificial component
Combine worlds
Display result

Figure 4.2: Augmented Reality Process

The photometric registration functionality is itself comprised of a number of child functions as shown in figure 4.3 and discussed in section 4.2.

Successful operation of this technique is dependant on a number of prerequisites. For example, the assumption is made that sufficient data is available from two or more different data sets. Should only one data set be available then

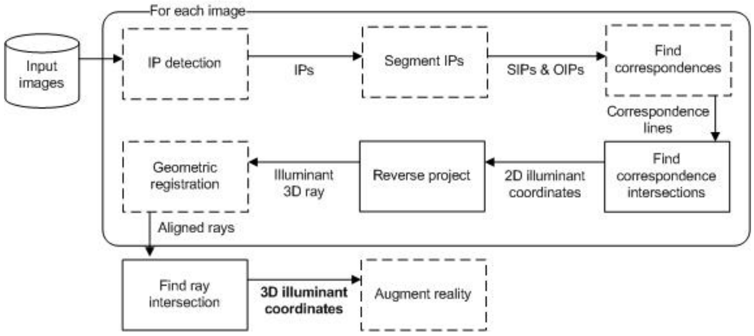


Figure 4.3: Technique Process

the absolute location of the illuminant would not be determined. Instead the technique would operate on a best endeavors basis and would determine a 3D ray on which the illuminant lies. As this provides the illuminant direction it may provide enough for certain AR application. It would however not be able to provide absolute illuminant condition matching as required for realistic shadowing. It is assumed that all input imagery contains both object geometry and the cast shadows that are associated with such geometry. The technique is unable to function should this not be the case. Input data is expected to arrive in the form of two-dimensional unsigned character arrays. Pixels may take either RGB or BGR format where the value of each colour component may range between 0 and 255 in intensity. Most camera input devices are able to supply data in these formats. The technique can also handle files containing still images or sequential video frames. Webcam type input devices are suitable so long as sufficient image quality is provided. It is presumed that input images are of a clear nature and contain low levels of noise. Both scene geometry and cast shadows should be free of occlusions and be visible within the field of view of all cameras. Images may be pre-processed in order to make them more suitable to geometric and photometric registration and to prepare them for interesting feature extraction techniques. However it should be noted that such pre-processing is only able to compensate for a certain level of noise and other bad data. Successful geometric registration is required in order for the results of the proposed photometric registration technique to be meaningful. Therefore the scene must be suitable to either marker-based or markerless geometric registration as discussed in section 3.1. In summary, the prerequisites of the proposed technique are as follows:

- A minimum of two input images

- Sufficient variation in viewing angles
- Geometric registration of camera pose
- Low noise input
- Identifiable scene shadow regions
- Identifiable scene object regions
- Two or more shadow-object feature correspondence

4.2 Photometric Registration

The system as a whole is comprised of a number of stages; the proposed technique exists from stage 6 through 16:

1. Object segmentation
2. Shadow segmentation
3. Image feature extraction
4. Object and shadow feature classification
5. Shadow and object feature correspondence detection
6. Formulate 2D lines between each correspondence pair
7. Find intersections between all correspondence lines
8. Obtain the average intersection
9. Reverse project between 2D and 3D world space
    a) Using intersection coordinate and near plane
    b) Using intersection coordinate and far plane
10. Geometric registration of intersection coordinates
11. Ray trace between near and far intersection coordinates
12. Repeat stages 1 to 9 for each input image
13. Find closest points on each illuminant ray
14. Formulate lines between each closest point set
15. Locate the centre point on these lines

16. Get the average of each centre points

Items 1 through 9 are repeated for each input image that is used. As a minimum of two suitable images are required these items are iterated through at least twice. Each iteration will deal with data that images a scene from a different angle. Such data is obtained prior to registration. A frame-grabber is used to obtain such data from video devices such as digital cameras (Including standard webcam devices). A frame grabber is a virtual device that captures a still image frame from an analog video signal or a digital video stream. As such, a frame grabber can also be used to obtain sequential frames from a video file. Figures 4.4 and 4.5 show suitable input data in the form of 3D virtual scenes as observed from two different angles. It is important that these images are obtained simultaneously. If this is not the case then the output of the technique will be unpredictable.

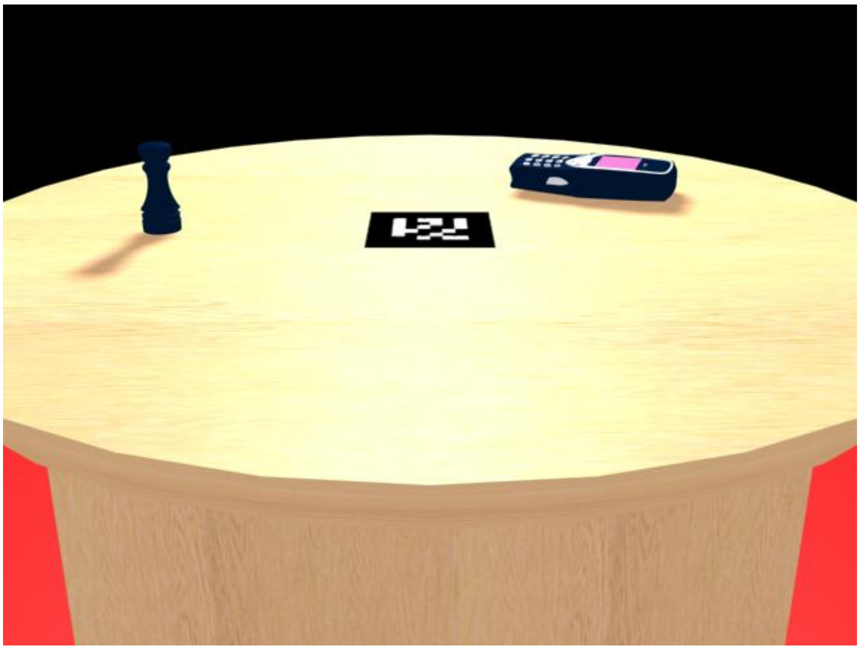


Figure 4.4: Suitable Camera View A

If the input imagery is to be obtained from picture files on a hard disk drive then the following occurs:

1. The file is loaded
2. The image header is processed
3. The appropriate image pixel data is stored accordingly

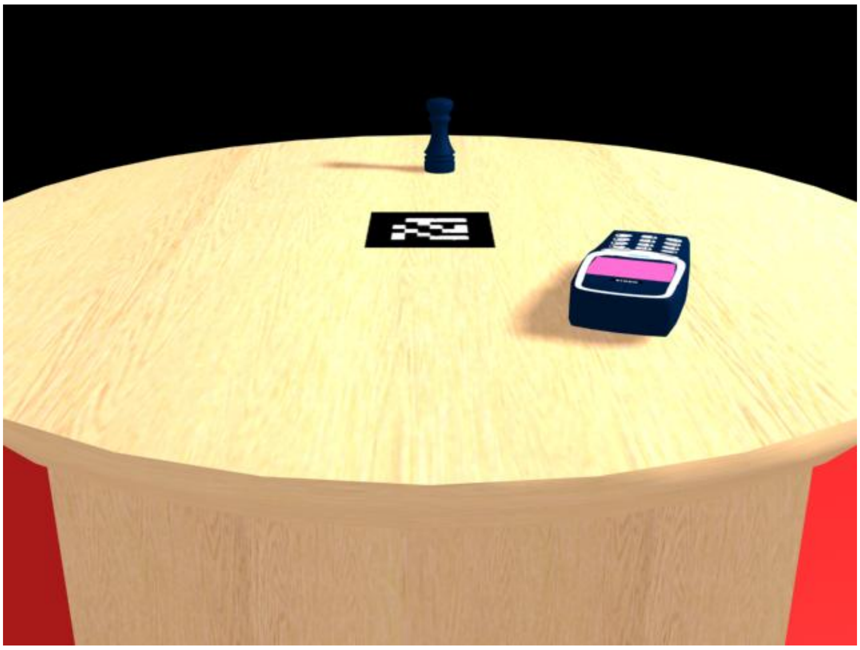


Figure 4.5: Suitable Camera View B

The frame data is converted into a raw image data format[*] and stored in an array of an appropriate size. The memory required to store each frame is dependant on the resolution of the input image. Each colour component requires a single byte of memory. This means that a single image pixel requires 3 bytes of space. Given an input image of width, w and height, h the total required bytes of storage, s can be calculated as below:

$$s = w * h * 3 \tag{4.1}$$

Given this equation we can derive tables 4.1 and 4.2 which outline the storage requirements for each input frame at different resolutions and aspect ratios for both grayscale and colour input.

It should be noted that higher resolution images imply greater computational cost due to the nature by which the image preparation and processing techniques operate. Additionally, input images may require pre-processing in order to make them suitable for geometric and photometric registration. Such pre-processing includes:

- Noise reduction and smoothing
- Colour to grayscale conversion

[*] Typically Red Green Blue (RGB) or Blue Green Red(BGR) formats

Resolution	Aspect	Memory
320x240	1.333	76800
640x480	1.333	307200
800x600	1.333	480000
1024x768	1.333	786432
1152x864	1.333	995328
1280x960	1.333	1228800
1400x1050	1.333	1470000
1600x1200	1.333	1920000
2048x1536	1.333	3145728
3200x2400	1.333	7680000
4000x3000	1.333	12000000
6400x4800	1.333	30720000
852x480	1.777	408960
1280x720	1.777	921600
1365x768	1.777	1048320
1600x900	1.777	1440000
1920x1080	1.777	2073600

Table 4.1: Memory Requirements of Grayscale Frame

Resolution	Aspect	Memory
320x240	1.333	230400
640x480	1.333	921600
800x600	1.333	1440000
1024x768	1.333	2359296
1152x864	1.333	2985984
1280x960	1.333	3686400
1400x1050	1.333	4410000
1600x1200	1.333	5760000
2048x1536	1.333	9437184
3200x2400	1.333	23040000
4000x3000	1.333	36000000
6400x4800	1.333	92160000
852x480	1.777	1226880
1280x720	1.777	2764800
1365x768	1.777	3144960
1600x900	1.777	4320000
1920x1080	1.777	6220800

Table 4.2: Memory Requirements of Colour Frame

- Edge detection

If input images such as shown in figures 4.4 and 4.5 are too noisy then steps to reduce this noise can be undertaken. These steps include the execution of blurring techniques which may be performed in either the spacial or frequency domain. Blurring in the spatial domain is achieved by performing a convolution with a kernel. After performing a box blur convolution a reduction in noise can be observed, however edges can become distorted significantly. This is especially true when a large amount of noise exists as its removal will require a large blur radius. A better approach is to instead use a kernel with a gaussian distribution. This allows the neighboring pixels to be weighted when calculating the resultant pixel. The box and gaussian smoothing techniques are discussed in section 3.2. As operational complexity is high when processing larger images the resolution of the input image should be taken into consideration when deciding which pre-processing technique is appropriate. When dealing with larger images the calculation would ideally be performed in the frequency domain. This would help keep operational complexity low[†]. This is not the case with lower resolution imagery as an overhead exists when performing the transform between the two domains[‡]. Figure 4.6 below shows a noisy input image that is unsuitable for processing. Figure 4.7 shows the same image after a gaussian smooth operation which is more suitable to registration.

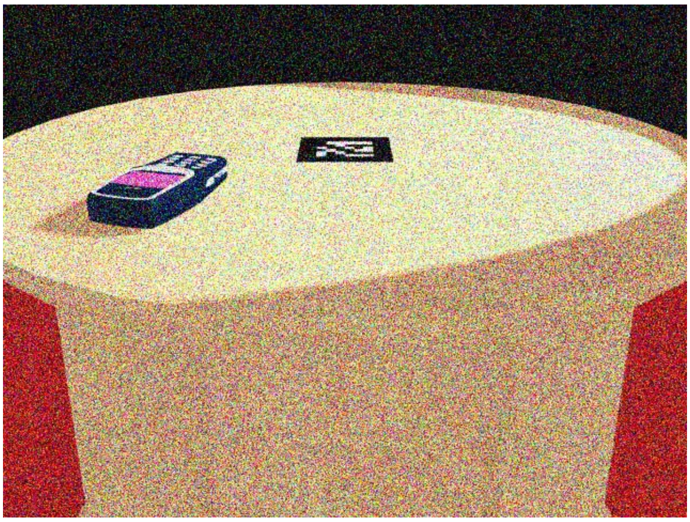


Figure 4.6: Noisy Input Data

Edge detection is performed prior corner detection. The canny edge detector is preferred for use with the proposed technique as it makes use of non maxima suppression and thresholded line completion which produces a pixel wide line

[†] A convolution in the spacial domain is equivalent to a simple multiplication in the frequency domain

[‡] Facilitated by the fourier and inverse fourier transform

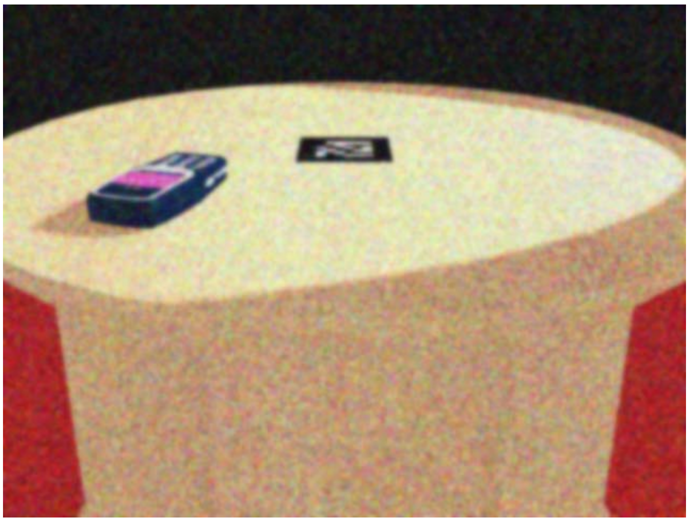


Figure 4.7: Gaussian Smoothing / Noise Reduction

with few undesired breaks due to noise or anomalous data. Figure 4.8 shows the results of performing the Canny edge detection algorithm on the gaussian blurred input image in figure 4.7. This image is now compatible with this technique.

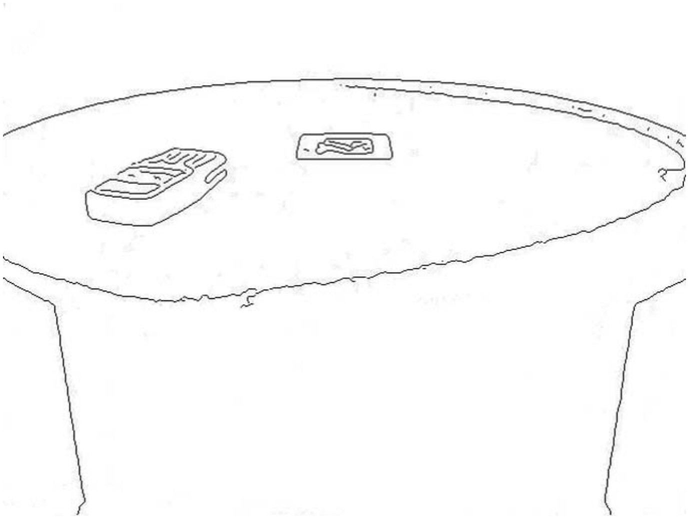


Figure 4.8: Input Canny Edge Detection

It is important not to over process an image so that its geometry can not be extracted by the chosen edge detector. This would be problematic as information relating to both shadow and object regions may be lost, as can be seen in figure 4.7. As such it is important to implement pre-processing only as required. Automated noise detection may be used to enable or disable image pre-processing techniques. However manually applying settings should be considered if this results yield anomalies. Additionally, the parameters passed to the canny function can be automatically adjusted until more desirable results are obtained. Lowering the canny threshold would allow for the detection of more difficult edges, however may also produce false positives that may confuse the corner detector. Feature extraction algorithms are then used on the image to obtain interest points. The SIFT method is preferred as it makes available additional information that is of use when detecting the correspondence between shadow interest points and object interest points. Figure 4.9 shows SIFT features extracted from a suitable input image. This figure shows many more SIFT descriptors than would be required for the intended purpose. SIFT allows for a threshold to be applied that would reduce the number of feature extractions, limiting output to corners on objects or shadows within the scene.

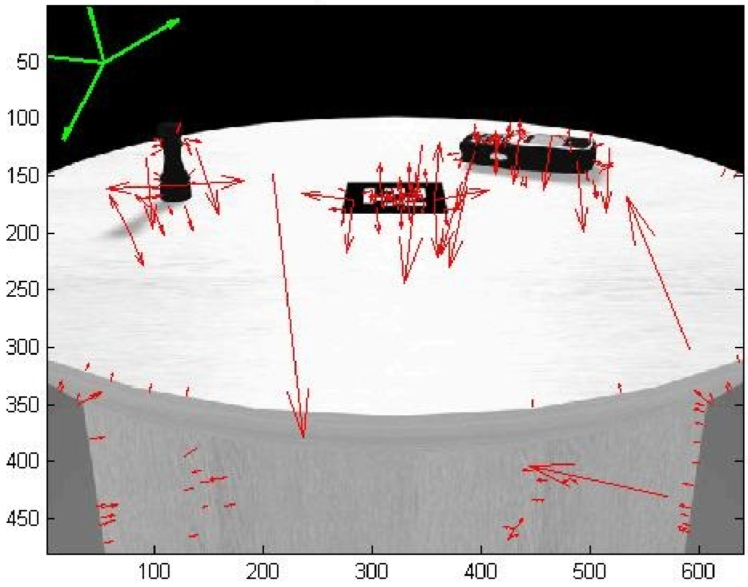


Figure 4.9: Performing SIFT on Input Using Matlab

Once interest points have been obtained they are to be classified as being associated with either a cast shadow or object geometry. Literature shows that existing image segmentation techniques would be suitable for this purpose and therefore IP classification is considered external to the scope of this thesis. Once regions are segmented it can be said that if an IP falls under a shadow region it is classified as being a shadow interest point (SIP). Similarly if an IP is detected on a region belonging to an object it is classified as an object interest point (OIP). This concept is illustrated in figure 4.10.

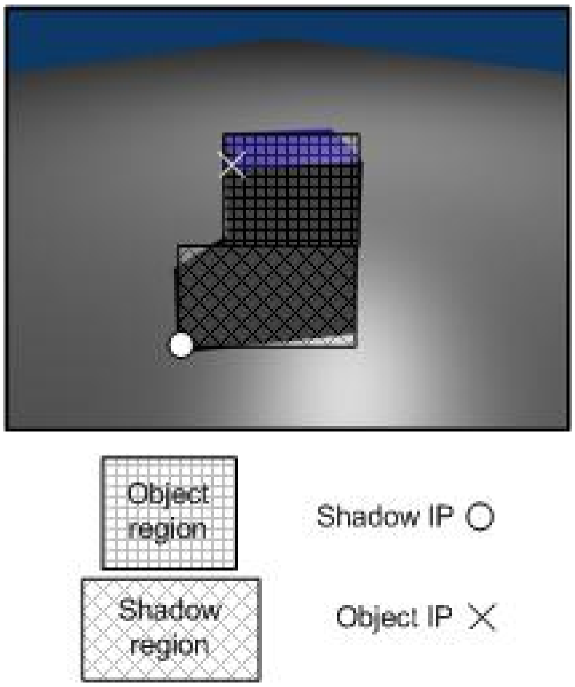


Figure 4.10: Shadow and Object Segmentation and Classification

It is important that the correspondence between shadow interest points and object interest points be determined. At least two correspondences per image are required, but more will offer improved accuracy. The determination of IP correspondence involves the matching of the point on a shadow corner to the point on the object that cast it. This problem is beyond the scope of this project and future work will adapt existing correspondence detection techniques for this purpose. A line can be drawn between the associated points once such correspondence has been found, as shown in figure 4.11. This line is hereby referred to as the correspondence line (CL).

A single CL is essentially a vector that points towards the illuminant on the 2D image plane. One CL is not enough to determine the absolute location of the illuminant, but if extrapolated to infinity it will eventually pass through it. This statement assumes accurate detection of interest points and sufficient image resolution. A second CL obtained from two other interest points is required to find the absolute location of the illuminant two dimensionally. More correspondence lines can be factored into the calculation to mitigate inaccuracies. It is important that anomalous lines are excluded as they will introduce error. Anomalous correspondence lines include those that do not point towards a similar point as

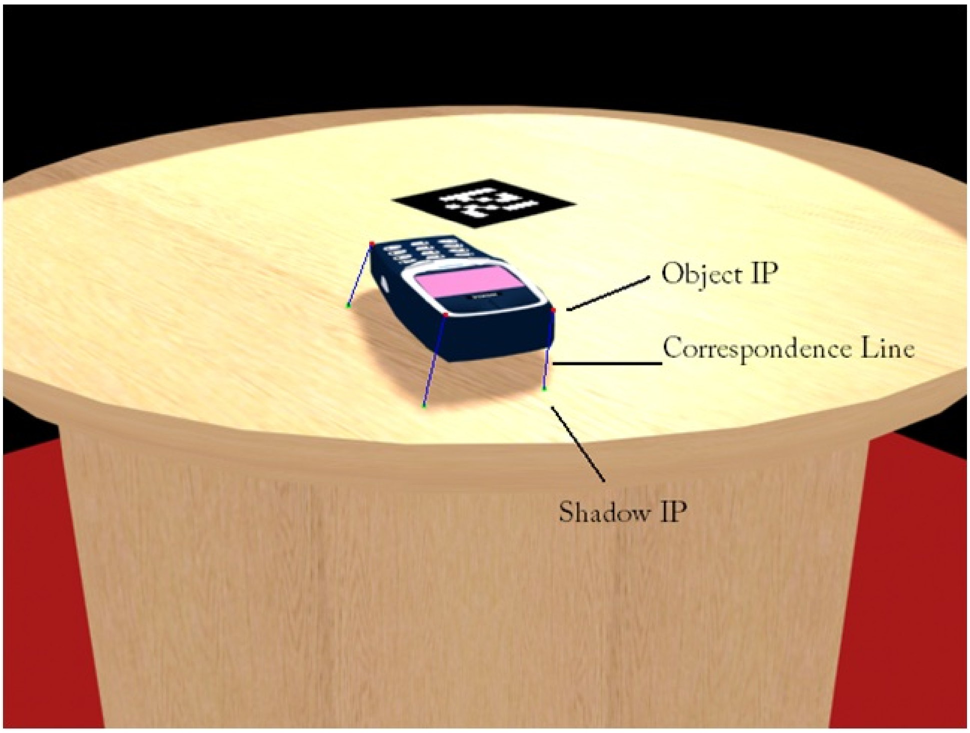


Figure 4.11: Interest Point Correspondence

the majority. This can be detected and dealt with by thresholding. Such an error can be caused by incorrect CL formation due to anomalous IP matching. Future directions in this area are discussed in chapter 7.

The 2D location of the illuminant can be found by calculating the intersection, or average intersection of two or more correspondence lines. Anomalous correspondences should be omitted from the calculation to avoid tainting the results. Anomalous lines include those with the same gradient, such as parallel and identical lines. These anomalies may occur when viewing from obscure angles. Parallel lines should be ignored as they introduce a divide by zero into the calculation. The system checks for such eventualities and throws an exception should they occur. The implementation is able to instead make use of alternative correspondence lines should they be available. If two correspondence lines have the same gradient and overlap then they would intersect at multiple pixel locations, and cause a divide by zero error. This eventuality is detected and handled prior to performing the intersection calculation. If no errors are present then each correspondence line is checked for intersections against every other correspondence line and the results are averaged. Intersections that occur in the direction of the coordinate vector are referred to as forward intersections, those that occur in its inverse are referred to as reverse intersections. Only forward intersections are considered valid. This concept is illustrated in figure 4.12. Backward intersections may occur due to errors when detecting interest points or by mismatching correspondence pairs.

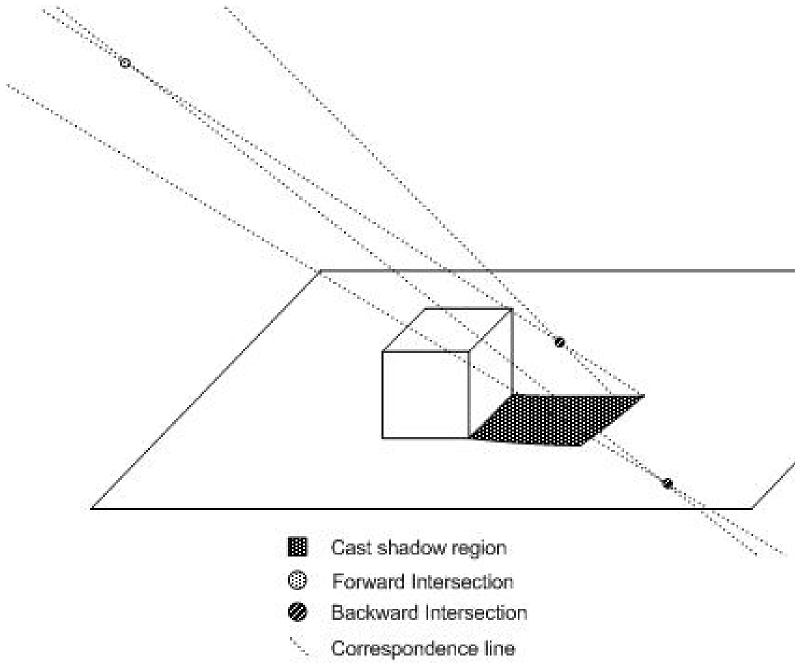


Figure 4.12: Forward and Backward 2D Intersections

Given the two correspondence lines expressed implicitly as:

$$a_1x + b_1y = d_1 \tag{4.2}$$

$$a_2x + b_2y = d_2 \tag{4.3}$$

The 2D illuminant position is calculated as the intersection of these lines as per the following two equations:

$$x = \frac{b_2d_1 - b_1d_2}{a_1b_2 - a_2b_1} \tag{4.4}$$

$$y = \frac{a_1d_2 - a_2d_1}{a_1b_2 - a_2b_1} \tag{4.5}$$

Where, a and b are points on a line, the indices 1 and 2 represent separate lines, and x and y represent 2D unknown intersection coordinates. Only two correspondence lines are required for accurate results, providing that the chosen interest points are themselves accurate and the correspondence lines do not create one of the error conditions as discussed above. Additional correspondence lines may be used and the results averaged in order to achieve greater accuracy. Robustness may be achieved by using several correspondence lines. The average intersection is calculated as follows:

$$\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i \tag{4.6}$$

Where x is an array of vectors containing the intersection points that represent potential illuminant positions. At this stage the technique requires the results of geometric registration in order for the following calculations to have meaning. It does not matter when the geometric registration calculations take place so long as they occur before this point in execution. Any geometric registration technique may be used but realism and complexity implications should be considered when making that decision. Fiducial markers are not a requirement of this photometric registration method but it should be noted they may be required by the chosen geometric registration technique. To mitigate disruption to scene realism it is preferable to use markerless registration and instead make use of natural features. The camera pose is determined as illustrated by figure 4.13. Consequently a world view matrix is created that defines the transformation to scale, rotate and translate the virtual world so that it correctly aligns with the real world.

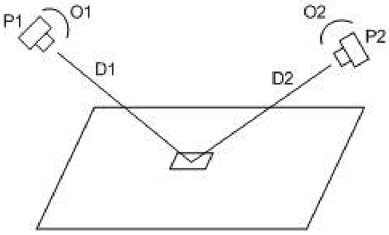


Figure 4.13: Camera Pose Estimation

Once multiple images have each yielded two dimensional positions the technique is able to determine the location three dimensionally. This can only be done in a meaningful way if information from geometric registration is available. In computer graphics, the transformation between 3D and 2D coordinates is usually a one way process. Depth information is lost during the projection and without it the 2D coordinate can not be transformed back into 3D space. As the technique is dealing with 2D images in the first place, this reverse transformation is similarly not possible. Instead, it is possible to use the information available to derive a 3D ray on which the illuminant lies. In order to convey how this process works it is first important to explain the forward process that is used by graphics

APIs[§] to convert from 2D to 3D. The forward projection process makes use of both a projection matrix and a modelview matrix in order to map a point in 3D space into 2D pixel locations. This is called a projection transform and usually takes place within the transformation pipeline of a graphics API. The transform maps the 3D world space coordinates onto a 2D plane, essentially flattening them. Figure 4.14 shows the components of this process. Projection matrices are either of an orthographic or perspective nature and are governed by 4x4 homogeneous projection matrices. Scene vertices, represented as homogenous coordinates, are multiplied by the combined modelview and projection matrix.

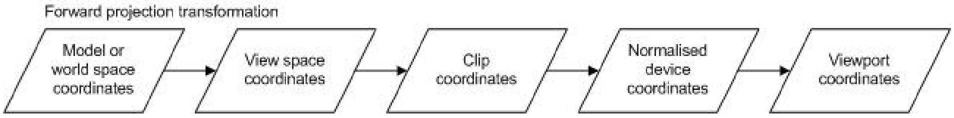

Forward projection transformation
Model or world space coordinates
View space coordinates
Clip coordinates
Normalised device coordinates
Viewport coordinates

Figure 4.14: Forward Perspective Projection Transform

Orthographic projections do not cause objects to appear smaller as they move off into the distance. This is not the case with a perspective transformation which is typically used when simulating cameras. A perspective transformation simulates objects visually appearing to become smaller and vertices closer together as they get further away. An example being two parallel rail road tracks that, by definition, never meet, however they would appear to do so when looking down them into the horizon. The homogeneous coordinate, w, facilitates this. The required projection matrix is created from six values that define the truncated pyramid frustum that represents the view of the camera that is being simulated. These values are:

- l - Left
- r - Right
- b - Bottom
- t - Top
- n - Near
- f - Far

These values are distances relative to the camera position, which can be assumed to be located at the origin at this point. Information such as the image aspect ratio and camera field of view should be available. The field of view of the

[§] Such as OpenGL and DirectX

input device can be calculated during the geometric registration stage by using camera calibration. This information is already available as camera calibration is part of the geometric registration process. The image height and width are also available and are required to compute the image aspect ratio, which can be derived as per the equation:

$$a = w/h \tag{4.7}$$

In perspective projection, the representation of a 3D vector point within the frustum is known as the eye coordinates. During the forward transformation the eye coordinates are mapped to a cube and the resulting coordinates are known as the normalized device coordinates. The x coordinate is mapped from $[b, r]$ to the range $[-1, 1]$, the y from $[b, t]$ to $[-1, 1]$ and the z is mapped from $[n, f]$ to $[-1, 1]$. Figure 4.15 shows this mapping.

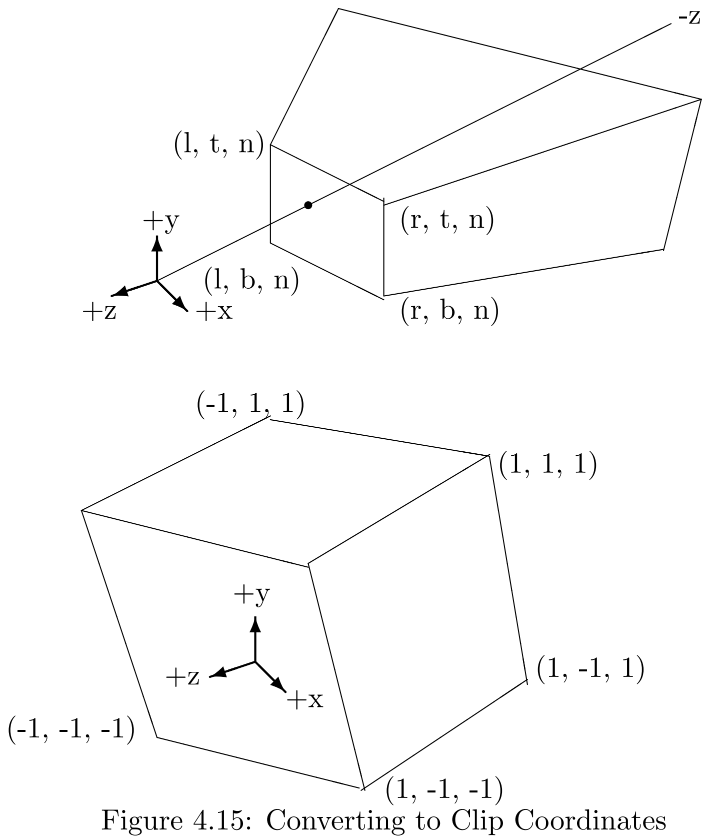


Figure 4.15: Converting to Clip Coordinates

Here the eye coordinates are defined within a right hand coordinate system, normalized device coordinates are defined using a left hand system. The frustum boundaries are obtained as below, where the field of view is specified in radians.

$$hh = tan(FoV * 0.5) * near \tag{4.8}$$
$$hw = hh * aspect \tag{4.9}$$
$$l = -hW \tag{4.10}$$
$$r = hW \tag{4.11}$$
$$b = -hh \tag{4.12}$$
$$t = nn \tag{4.13}$$
$$n = near \tag{4.14}$$
$$f = far \tag{4.15}$$

The flat 2D plane onto which the 3D points are projected is known as the near plane or the projection plane. The eye space coordinates are multiplied by the matrix to transform them into clip coordinates during a forward transform. At this stage, the clip coordinates are still homogenous. In order to obtain normalized device values (NDC) the x, y, z components of the clip coordinates are divided by the homogenous component, w.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = P. \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix} \tag{4.16}$$

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \end{bmatrix} \tag{4.17}$$

The standard transform projects the 3D point onto the near plane, otherwise known as the projection plane. Figure 4.16 and figure 4.17 show the point $[X_e, Y_e, Z_e]$ projected onto the near plane form the top down and side views respectively. The eye space coordinate x_e is mapped to the projected coordinate x_p using concept of the ratio of similar triangles. The coordinate y_e is calculated in a similar manor.

$$x_p/x_e = -n/z_e x_p = \frac{-n.x_e}{z_e} = \frac{n.x_e}{-z_e} \tag{4.18}$$

$$y_p/y_e = -n/z_e y_p = \frac{-n.y_e}{z_e} = \frac{n.y_e}{-z_e} \tag{4.19}$$

The projected coordinates x_p and y_p are inversely proportional to $-z_e$, therefore the w component of the clip coordinates can be set as $-z_e$. x_p and y_p are then mapped to x_n and y_n of NDC with the linear relationship $[l, r]$ to $[-1, 1]$ and $[b, t]$ to $[-1, 1]$.

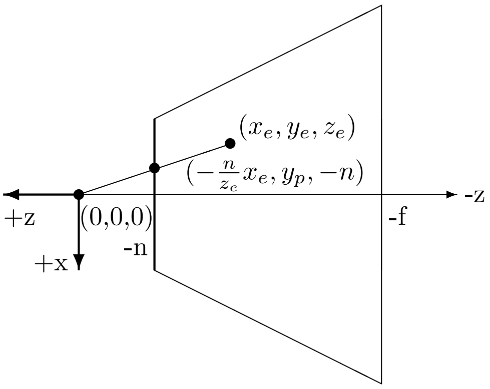


Figure 4.16: Point Projection: Top View

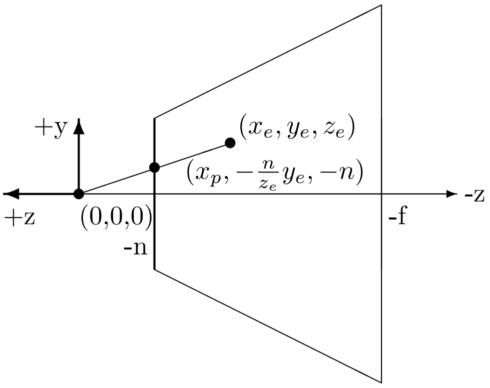


Figure 4.17: Point Projection: Side View

Equation (4.20) shows the frustum projection matrix[¶] used.

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{4.20}$$

As a non-linear relationship exists between Z_e and Z_n the precision is high towards the near plane and there is very little towards the far plane. Converting from NDC to screen coordinates occurs by performing a viewport transformation. The NDC is scaled and translated to fit to the rendering screen. These results

[¶] Perspective projection matrix

would then be passed to the rastorizer. The area to map onto is a rectangle, defined by x,y,w and h. The depth is defined by n and f. Which represent the near and far planes respectively. The window coordinates are computed with the given parameters:

$$\begin{matrix} x_w \\ y_w = \\ z_w \end{matrix} \begin{bmatrix} w/2x_ndc + (x + (w/2)) \\ h/2y_ndc + (y + (h/2)) \\ (f - n)/2z_ndc + (f + n)/2 \end{bmatrix} \tag{4.21}$$

The viewport transform formula is acquired by the linear relationship between NDC and window coordinates:

$$\begin{cases} -1 \quad -> x & -1 \quad -> y & -1 \quad -> n \\ 1 \quad -> x + w & 1 \quad -> y + h & 1 \quad -> f \end{cases} \tag{4.22}$$

In order for the technique to obtain a ray on which the 3D illuminant lies, it is required that the above process be reversed as shown in figure 4.18. As previously mentioned, this process was not meant to be fully reversible, therefore some minor changes have been made. The mathematical differences are presented in section 5.1. This is executed for each camera, as shown in figure 4.19. This mitigates the limitations of the backward transform, and allows for depth information to be derived by combining results for multiple cameras.

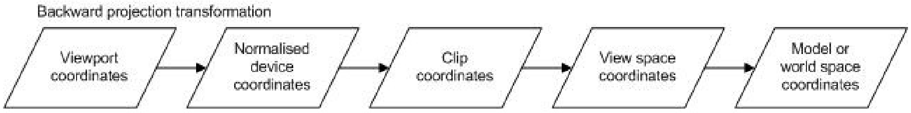

Backward projection transformation
Viewport coordinates
Normalised device coordinates
Clip coordinates
View space coordinates
Model or world space coordinates

Figure 4.18: Backward Perspective Projection Transform

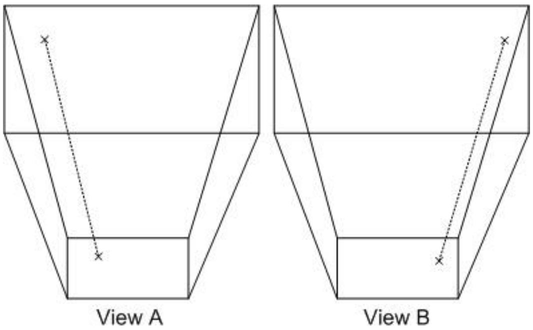


Figure 4.19: Reverse Projection Cam A & Cam B

Such a reverse projection transformation can be difficult as it is not possible to obtain depth information. However it is still possible to derive a ray that passes through the 3D illuminant coordinates in question. Figure 4.20 visualizes two rays cast through the plane of two sample input images. It shows that each ray is cast through the 2D location of the illuminant and that once aligned, the intersection of these rays would be the 3D location of the illuminant. The figure shows how the rays would be cast with an orthographic projection as would be obtained when using an orthogonal projection matrix, for visualization purposes only. The technique actually makes use of a perspective projection matrix, thus taking perspective distortion into account.

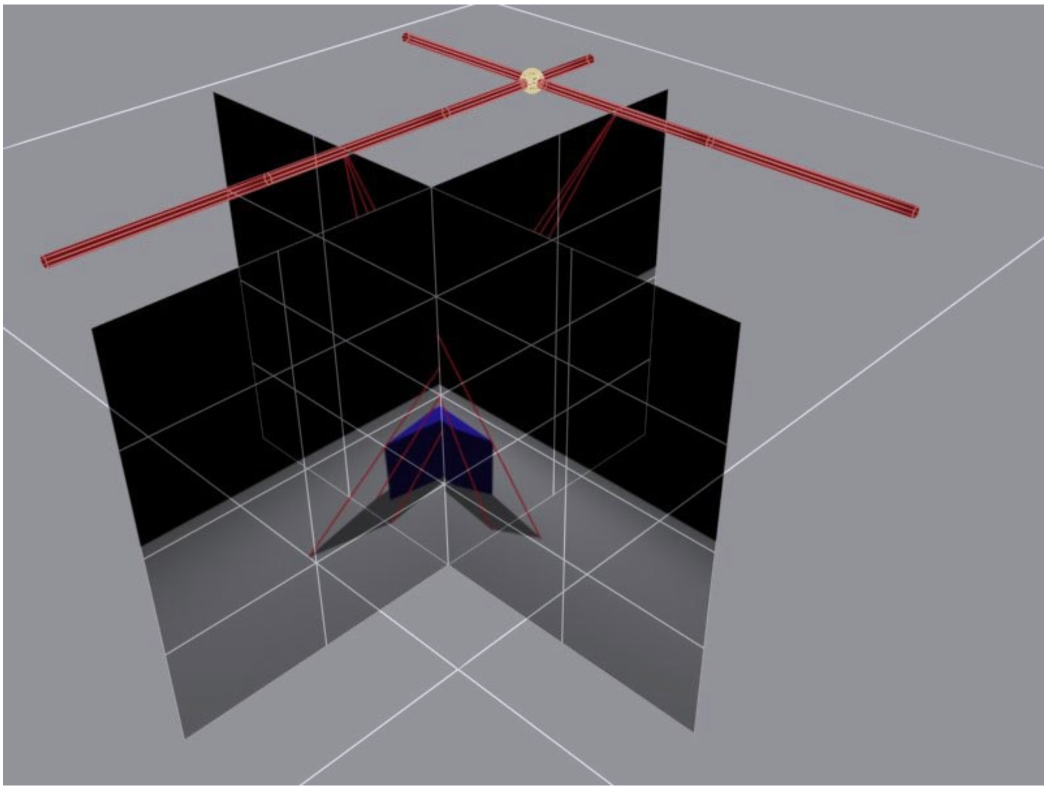


Figure 4.20: Illuminant Rays

Figure 4.19 shows the two 3D illuminant rays before alignment. The alignment process is possible once a modelview matrix is obtained. This occurs during the geometric registration and camera pose estimation phase. The modelview matrix is constructed from scale, rotational and translational matrices as shown in section 5.1. Therefore the requirements of such an un-projection are:

- A modelview matrix representing camera pose

- An appropriate perspective projection matrix
- Input screen coordinates
- Target depth value

The 3D world coordinates for the chosen depth can be calculated through this process providing that the above information is available to the technique. After each image has been processed the technique proceeds to combine the results from each. This enables the determination of the location of the 3D illuminant and therefore allows photometrically registered augmented reality. This 3D illuminant location is derived by treating the 3D rays obtained from each camera as approach lines and finding the closest point on each. The closest points on two 3D approach lines, L_1 and L_2, can be found by finding the minimum length of line, W_c, as shown in figure 4.21. Considering L_1 and L_2 to be infinite lines:

$$L_1 : P(s) = P_0 + s(P_1 - P_0) = P_0 + su \tag{4.23}$$

$$L_2 : Q(t) = Q_0 + s(Q_1 - Q_0) = Q_0 + tv \tag{4.24}$$

Let $W(s,t) = P(s) - Q(t)$ be a vector between points on the two lines. The technique then discovers the W line that has minimum length all potential values of s and t are considered. Eberly[19] presents a calculus based method and Teller[91] presents a geometric based approach of performing this calculation, however this technique adopts a faster approach.

It can be said that L_1 and L_2 are closest at unique points P(sc) and Q(tc), in any n-dimensional space, for which W(sc,tc) refers to its minimum length. The line segment $P(sc)Q(tc)$ joining the closest points is uniquely perpendicular to both lines at the same time when L_1 and L_2 are not parallel. There are no other line segments between L_1 and L_2 for which this applies. The vector represented by W_c = W(sc,tc) is uniquely perpendicular to the direction vectors U and V. Therefore the two equations are satisfied as follows:

$$U \cdot W_c = 0 \tag{4.25}$$

$$V \cdot W_c = 0 \tag{4.26}$$

These two equations can be solved by substituting:

$$W_c = P(sc) - Q(tc) = W_0 + s_c\vec{u} - t_c\vec{v}, \qquad W_0 = P_0 - Q_0 \tag{4.27}$$

To obtain:

$$(u.u)S_c - (u.v)t_c = -u.W_0 \tag{4.28}$$

$$(v.u)S_c - (v.v)t_c = -v.W_0 \tag{4.29}$$

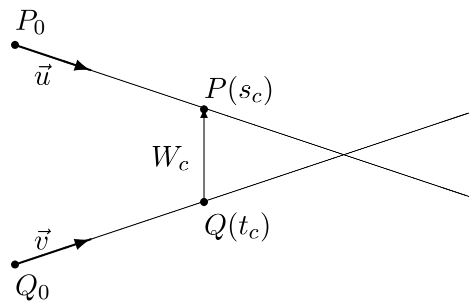


Figure 4.21: Closest Points of Approach Lines

Then, letting $a = u.u$, $b = u.v$, $c = v.v$, $d = u.W_0$ and $e = v.W_0$. s_c and t_c are solved as follows:

$$s_c = \frac{be - cd}{ac - d^2} \qquad t_c = \frac{ae - dd}{ac - b^2} \tag{4.30}$$

It should be noted that $ac - b^2 = ||u||^2||v||^2 - (||\vec{u}||||\vec{v}|| \cos q)^2$ is always non-negative.

In order to solve the parallel distance the value of one parameter can be hard-coded and one equation can be used to solve the other. Selecting $s_c = 0$:

$$t_c = d/b = e/c \tag{4.31}$$

Having solved for s_c and t_c the technique has values representing the points $P(s_c)$ and $Q(t_c)$ where the two lines L_1 and L_2 are closest. The distance between these points can then be calculated using the equation:

$$d(L_1, L_2) = ||P(s_c) - Q(t_c)|| = (P_0 - Q_0) + \frac{(be - cd)\vec{u} - (ae - bd)\vec{v}}{ac - b^2} \tag{4.32}$$

Once the closest points on the two illuminant rays are obtained the technique is able to predict the illuminant coordinates. It does this by simply finding the centre of the line between these points. This is the line marked W in figure 4.21. The centre of the line can be calculated using the equation:

$$I = W_0 + (W_0 - (W_1/2)) \tag{4.33}$$

The vector I represents the location of the illuminant. These coordinates can be used when instantiating a virtual light. Once the value I has been obtained the technique can continue to augment reality using photometrically registered illumination conditions. To do so, the artificial scene is generated by rendering the artificial components and overlaying them over one of the original input images. The scenes are correctly aligned using the information obtained during the geometric registration stage and photometrically registered by creating a virtual illuminant at the virtual coordinates that are the equivalent of the actual position of the real illuminant. If the technique is operating on sequential frames as opposed to still images then the result is a properly lit augmented reality video stream that can be relayed to a video device such as a monitor, projection unit or head mounted display. Once the light source has been located we can perform the augmentation. Geometric registration techniques should be applied to ensure accurate alignment between the real and virtual worlds. Once an artificial illuminant is positioned and augmented objects are registered to the same coordinate system, augmented and real objects will appear to be lit in the same manner. The scene can now be passed through AR shadow casting techniques such as those presented by State[90], Williams[100] and Haller[32] as discussed in appendix C. The generated shadows will appear to be cast from the same light source as actual shadows. Once conditions are correctly matched, the world is photometrically and geometrically registered as can be seen visually in figure 4.22

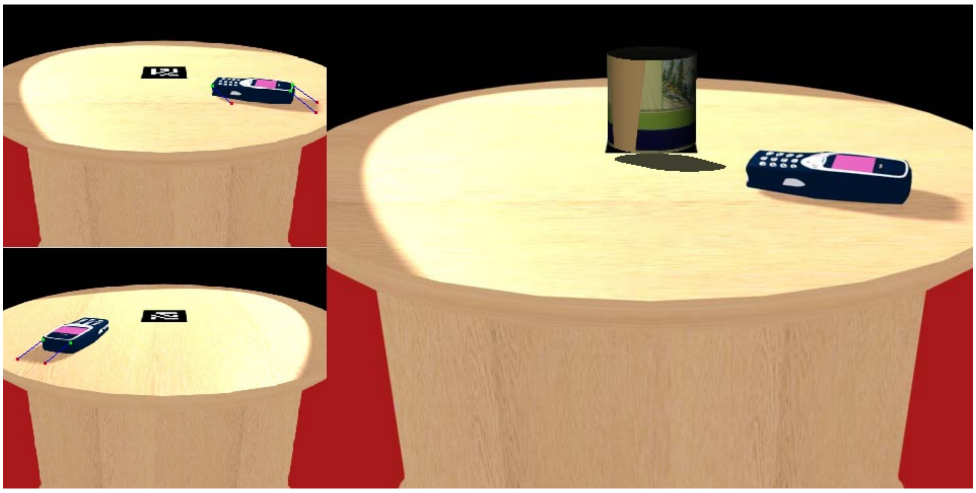


Figure 4.22: Virtual Shadowing Using Real Illuminant

4.3 Error Mitigation Strategies

The technique is expected to generate anomalous results under certain specific circumstances. Where possible, the technique attempts to mitigate such anomalies. Where the object or shadow are occluded by other scene geometry it is not possible to extract the associated interest points. Therefore if an object point is visible but the corresponding shadow point is occluded or out of the viewing frustum then another object and shadow pair should be sought out instead. This additional searching may add to computation time by an unknown amount but so long as two correspondence pairs can be found this should not present further

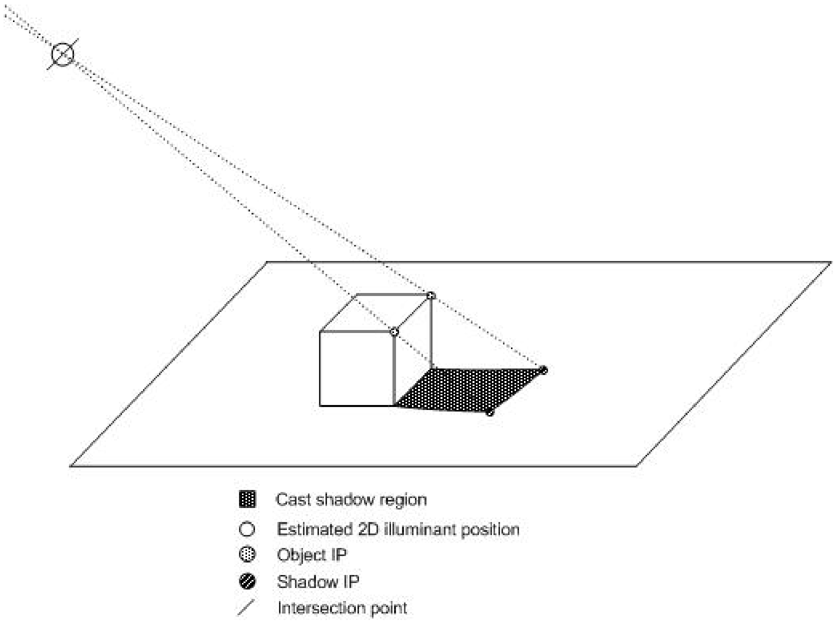


Figure 4.23: Correspondence Line Intersection

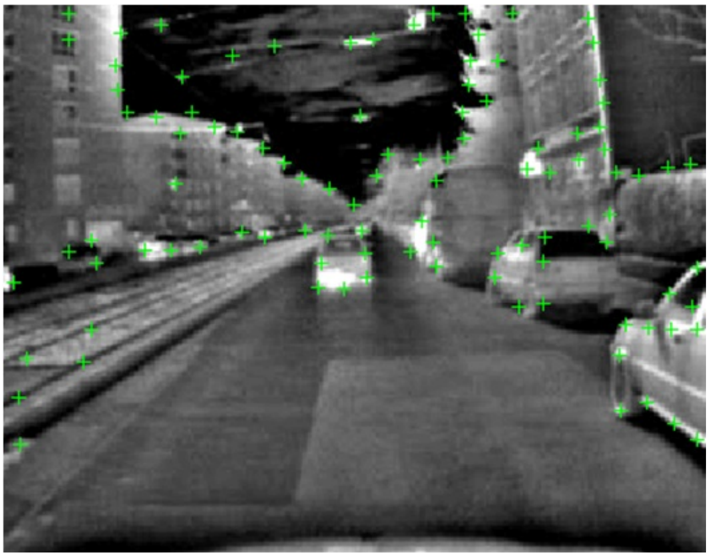


Figure 4.24: Interesting Image Features[61]

problems. Error in the accuracy of the final result may be introduced by a number of factors including internal rounding errors, visual noise or the inaccurate detection of interesting features. A number of strategies can be implemented in order to mitigate this error. Noise reduction and image processing techniques, the aver-

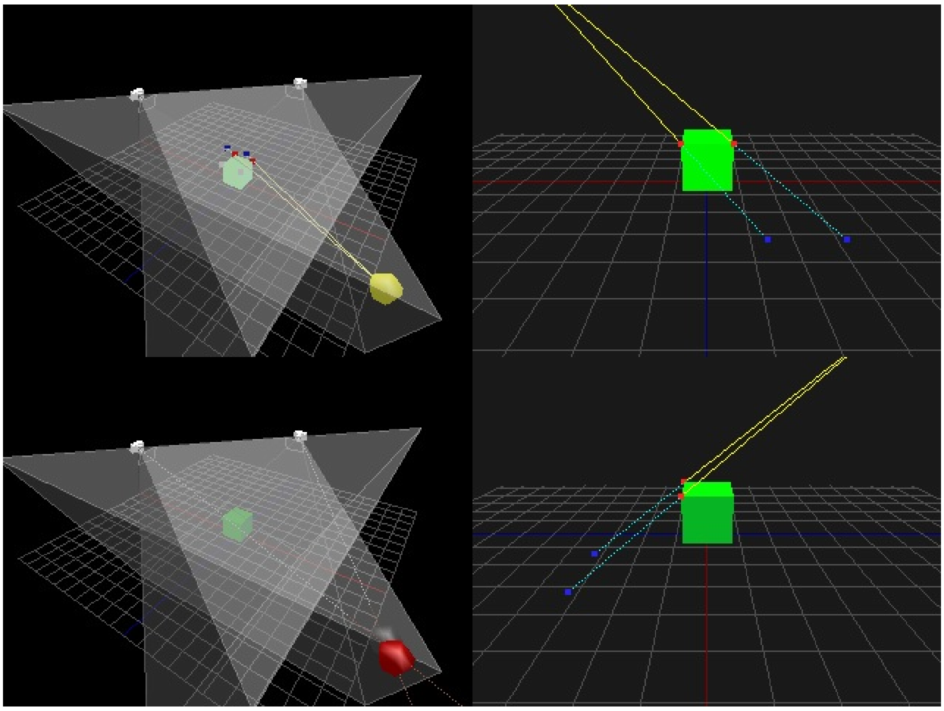


Figure 4.25: Tracking Pseudo-Illuminant

aging of values between multiple sequential frames and the omission of blatantly bad data are good first steps. Bad data includes illuminants that are detected in impossible places[‖] and correspondence lines that are the same line or otherwise parallel to each other. The former eventuality is less common than the latter and is usually due to the mismatching of correspondence points. The technique is able to detect reverse intersections as discussed in chapter 5 in order to eliminate such issues. Parallel or same correspondence lines[**] may occur when viewing the scene geometry from obscure angles. The mismatched correspondence problem is an area for future work as this is an additional complex problem that is beyond the scope of this thesis. When the above issues are experienced it is possible to introduce mitigation to compensate for individual factors. However it is also possible to make judgements based on temporal meta-data regarding the illuminant position. For example if an illuminant was detected as being in a certain position for a large number of sequential frames, only slighting deviating from a given position or trajectory then it would be perfectly reasonable to assume that any calculation that determines the illuminant to have rapidly moved a significant distance between frames is anomalous. In this circumstance it may be feasible to average the illuminant between historical frames, or drop the anomalous data

[‖] Such as beneath the geometry that is being analyzed or below the surface plane

[**] That intersect at multiple points

and instead use a previous known position. One additional mitigation would be to use temporal illuminant meta-data in order to extrapolate the velocity of any moving illuminant. The new position can be calculated by considering both this velocity vector and the time elapsed between the current frame and the previous frame. The technique does experience difficulties in a number of operational environments. These are extreme angle differences between two, or more, input cameras. It can be expected that multiple camera inputs would mitigate this problem, so long as they are all observing the same scene and geometric registration is possible for each input. The angles at which conditions would appear to be suboptimal are 180° and 0°. When approaching these angles the technique will begin to lose depth information. The threshold at which depth information is lost depends on the resolution at which the scene is imaged. The higher the angle the region around these angles in which accuracy is lost is reduced. Therefore higher resolution imagery mitigates this threshold angle and also partially mitigates the resulting error. The above concept is shown graphically and discussed in further detail in chapter 6. It should be noted that in the typical AR configuration where camera devices are head-worn it is physically impossible to achieve such extreme angles. Therefore, in practice, this would usually not be an issue.

Chapter 5

Implementation

During the course of the research project a number of applications and prototypes were designed and implemented. Those most relevant to the proposed technique are outlined in this chapter. The major implementations are:

- MathCAD mathematical model
- Detection framework classes
- 2D detection prototype
- 3D detection prototype
- Photometric tracking Simulation
- Sequential detection prototype

The above development was required for the successful progression of the technique proposed in chapter 4. Directly relevant functionality is discussed in this chapter. The functionality to perform image processing and automated feature detection has been developed separately and is not an explicit topic of discussion in this chapter.

5.1 Mathematical Model

A mathematical model of the proposed technique was created in the engineering software application 'MathCAD' by Mathsoft. The purpose of the model was to act as a proof of concept which confirmed the feasibility of the proposed photometric technique. The model also facilitated the rapid generation of metrics and data-sets, an analysis of which is performed in chapter 6. Additionally, it enabled the visualization of inner technique functionality at the numerical level, thus facilitating decisions throughout the initial design process.

The mathematical framework can be divided in to a number of relevant sections, including those discussed within the technique specification.

The following is mathematically modeled:

- Processing of virtual camera parameters
- Creating transformation matrices
- Generation of simulated interest points
- Find illuminant 2D coordinates using IP correspondences
- Reverse projection of both 2D estimations into 3D coordinates
- The use of 3D rays to locate illuminant 3D coordinates

Initial data is first input into a series of variables and constants prior to performing a given simulation. The input data describes the two cameras, the real illuminant position, $\vec{N}$ and scene geometry. The values include, θ_c, ϕ_c and d_c which are camera rotation angle, camera pitch angle and camera distance respectively, where c denotes the camera number. The values w_c and h_c represent camera viewport width and height respectively and indicate image spatial resolution. The value $\vec{p}$ is a 4 dimensional vector representing the position of a cube that simulates real scene geometry. It contains the elements $\vec{p_x}$, $\vec{p_y}$, $\vec{p_z}$ and $\vec{p_w}$. $\vec{p_w}$ is always zero and merely exists to facilitate matrix multiplication at a later stage. S represents the size of the cube. It is a 3 dimensional matrix containing the elements S_x, S_y and S_z. The camera field of view is represented by σ_c and near and far planes are represented by n_c and f_c respectively. The real illuminant position is represented by I which is a 4 dimensional matrix where the w value is initially set to 1. The image depth range is specified within, r, where the components, r_0 and r_1 represent the lower and upper limits respectively. No other input parameters are required.

The MathCAD model is able to simulate technique operation for the values supplied in order to determine its accuracy under such conditions. The camera viewing frustum parameters, bottom, top, left and right are required. These are represented by b, t, l, r respectively and are calculated as follows:

$$b_c = n_c \cdot \tan \sigma_c \tag{5.1}$$

$$t_c = n_c \cdot \tan \sigma_c \tag{5.2}$$

$$l_c = -a_c \cdot n_c \cdot \tan \sigma_c \tag{5.3}$$

$$r_c = a_c \cdot n_c \cdot \tan \sigma_c \tag{5.4}$$

Where, a, represents the aspect ratio for the given camera. From this it is possible to derive the correct projection matrix that describes each camera. In a real situation the camera parameters would have been obtained through the camera calibration process as described in appendix A, therefore this information would represent the real camera devices being used. As camera calibration is a typical component of geometric registration prior knowledge of this information can be assured. The projection matrix for each camera, P_c is defined as follows:

$$\begin{bmatrix} \frac{2 \cdot n_c}{r_c - l_l} & 0 & \frac{r_c + l_c}{r_c - l_c} & \\ 0 & \frac{2 \cdot n_c}{t_c - b_c} & \frac{t_c + b_c}{t_c - b_c} & 0 \\ 0 & 0 & \frac{-(f_c + n_c)}{f_c - n_c} & \frac{-2 \cdot f_c n_c}{f_c - n_c} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{5.5}$$

The viewport of each camera, V_c, is given as:

$$V_c = \begin{bmatrix} 0 \\ 0 \\ w_c \\ h_c \end{bmatrix} \tag{5.6}$$

The aspect ratio for each camera, a_c, is calculated by performing:

$$a_c = \frac{w_c}{h_c} \tag{5.7}$$

The modelview matrix that represents the pose of the camera is obtained as follows[*]:

$$M_c = T_c \cdot R_c \tag{5.8}$$

Where, T_c, represents translation matrices that represent initial camera positions. T_c is defined as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

This essentially moves the camera back by a specified distance, d, ready for rotation transformation using R_c. The value R_c, represents the camera rotation matrix which is calculated from pitch and rotation matrixes as follows:

$$R_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi_c & -\sin\phi_c & 0 \\ 0 & \sin\phi_c & \cos\phi_c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_c & 0 & \sin\theta_c & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_c & 0 & \cos\theta_c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

[*] It should be noted that when implementing this equation the correct order of multiplication is important

A matrix is calculated for later use when performing the reverse projection. This matrix is the inverse of the multiplication between the projection matrix and the modelview matrix as shown:

$$I_c = (P_c \cdot M_c)^{-}1 \tag{5.11}$$

The vertices of the cube that represent the simulated real world geometry are calculated as follows:

$$v_0 = \begin{bmatrix} \frac{-S_x}{2} + \vec{p}_x \\ S_y + \vec{p}_y \\ \frac{-S_z}{2} \vec{p}_z \\ 1 \end{bmatrix} \tag{5.12}$$

$$v_1 = \begin{bmatrix} \frac{S_x}{2} + \vec{p}_x \\ S_y + \vec{p}_y \\ \frac{-S_z}{2} \vec{p}_z \\ 1 \end{bmatrix} \tag{5.13}$$

$$v_2 = \begin{bmatrix} \frac{S_x}{2} + \vec{p}_x \\ S_y + \vec{p}_y \\ \frac{S_z}{2} \vec{p}_z \\ 1 \end{bmatrix} \tag{5.14}$$

$$v_3 = \begin{bmatrix} \frac{-S_x}{2} + \vec{p}_x \\ S_y + \vec{p}_y \\ \frac{S_z}{2} \vec{p}_z \\ 1 \end{bmatrix} \tag{5.15}$$

Here, S represents the size of the cube and its elements S_x, S_y and S_z represent width, depth and height respectively. The value $\vec{p}$ represents the cube position. Only the 4 topmost vertices are calculated as these are the only corners that will be casting shadows. Once projected into 2D these vertices will be used as object interest points. Thus effectively simulating real scene geometry within a 2D image. Additionally, a plane that represents the ground on which a shadow would be cast is created. This plane is defined by the following equations:

$$e3 = \beta - \alpha \tag{5.16}$$

$$e1 = \gamma - \beta \tag{5.17}$$

$$o = \frac{e3 \times e1}{|e3 \times e1|} \tag{5.18}$$

$$\delta = \alpha \cdot o \tag{5.19}$$

Where the value, o, in this equation is the plane normal, a fourth dimension, o_w, is added to this for calculation purposes. δ is the distance between the closest part of the plane to the origin and the origin. In this implementation the plane is always defined using the points $\alpha = [4 \quad 0 \quad -2], \beta = [2 \quad 0 \quad -3]$ and $\gamma = [1 \quad 0 - 1]$ although this may vary as required. This plane acts as a surface on which simulated shadow interest points can be cast.

The next segment of the mathematical model creates simulated 2D points for both object and shadow interest points by calculating forward projections. Object interest points are obtained by multiplying the cube vertices by the projection and modelview matrices as follows:

$$v_i \cdot P_c \tag{5.20}$$

Shadow interest points are simulated by casting a ray from the position of the real illuminant, through a vertex at the top of the cube and onto the plane. The intersection point is then projected and orientated by multiplying the coordinates by the projection and modelview matrices to obtain simulated 2D shadow interest points. The ray is cast by first obtaining its direction in the format of a unit vector as follows:

$$q_i = \frac{v_i - \vec{N}}{|(v_i - \vec{N})|} \tag{5.21}$$

As the shadow is cast by the furthest two vertices from the illuminant, only the two vertices with the greatest distance from the illuminant should be used. This distance is calculated by subtracting the 3D real object vertex position from the 3D real illuminant position as follows:

$$\vec{N} - v_i \tag{5.22}$$

The above is repeated for each vertex indexed by i. Only the vertices associated with the largest two distances are used. Ray intersections with the plane are then calculated as follows:

$$y_i = \vec{N} + (\frac{\delta - \vec{N} \cdot o}{q_i \cdot o} \cdot q_i) \tag{5.23}$$

This calculation takes place for each ray, identified by the index i, the result being a set of 3D intersection coordinates. A w value of 1 is added to y in each case. The 3D coordinates are translated into 2D pixel locations at which the interest points lie by undertaking a number of transformations. The object interest points[†] are first transformed from world coordinates into eye coordinates:

$$v_{\text{eye}} = M_c \cdot v_i \tag{5.24}$$

[†] The upper most cube vertices

The eye coordinates are transformed into clip coordinates:

$$v_{\text{clip}} = P_c \cdot v_{\text{eye}} \tag{5.25}$$

The clip coordinates into normalized display coordinates:

$$v_{\text{ndc}} = \frac{v_{\text{clip}}}{v_{\text{clip}_w}} \tag{5.26}$$

Finally, from NDC into pixel coordinates:

$$K_i = \begin{bmatrix} \frac{V_2}{2} \cdot v_{\text{ndc}_0} + \frac{V_2}{2} + V_0 \\ \frac{V_3}{2} \cdot v_{\text{ndc}_1} + \frac{V_3}{2} + V_1 \\ \frac{r_1 - r_0}{2} \cdot v_{\text{ndc}_2} + \frac{r_1 - r_0}{2} \end{bmatrix} \tag{5.27}$$

In the above equations, t represents a coordinate within 3D space. This coordinate can represent a vertex that is to be translated into an object interest point, or a plane/ray intersection that is to in turn be translated into a simulated shadow interest point. The above processes repeat to obtain object and shadow interest points for each camera view. This mathematical model supports two or more cameras with no upper limit, each camera increases result accuracy. Camera limits will apply within actual implementations due to physical limitations, available resources and cost. Multiple users observing the same scene would likely further improve accuracy. The mathematical implementation rounds the resulting projected 2D coordinates into integers in order to simulate pixel level precision. At this stage the system possesses simulation data equivalent to that which would be obtained from live input images.

The remainder of the mathematical model deals with the core functionality of the technique that was proposed in chapter 4. The location of the 2D illuminant is detected by drawing a line from the shadow interest point and the corresponding object interest point. This model considers the object IP derived from a cube vertex to be associated with the shadow IP generated by casting the ray through that vertex. A correspondence line exists between the shadow and object IPs. The mathematical model considers only one intersection per camera input, caused by two correspondence lines. Two shadow and two object interest points are used. The correspondence lines are extrapolated and the intersection is calculated for each camera as follows:

$$g = \frac{1}{(\hat{w}_x - \hat{x}_x) \cdot (\hat{y}_y - \hat{z}_y) - (\hat{w}_y - \hat{x}_y) \cdot (\hat{y}_x - \hat{z}_x)} \hat{T} \tag{5.28}$$

$$= \begin{bmatrix} ((\hat{w}_x \cdot \hat{x}_y) - (\hat{w}_x \cdot \hat{x}_y)) \cdot (\hat{y}_x - \hat{z}_x) - (\hat{w} - \hat{x}) \cdot (\hat{y}_x \cdot \hat{z}_y - \hat{y}_y \cdot \hat{z}_x) \cdot g \\ ((\hat{w}_x \cdot \hat{x}_y) - (\hat{w}_x \cdot \hat{x}_y)) \cdot (\hat{y}_x - \hat{z}_x) - (\hat{w}_y - \hat{x}_y) \cdot (\hat{y}_x \cdot \hat{z}_y - \hat{y}_y \cdot \hat{z}_x) \cdot g \end{bmatrix} \tag{5.29}$$

Where $\hat{w}$, $\hat{x}$, $\hat{y}$ and $\hat{z}$ are the first OIP/SIP, and the second OIP/SIP positions respectively. This calculation is repeated for each camera input. A real implementation may consider multiple correspondence lines and evaluate which provide the best results, omitting the least promising lines based on a threshold. As the data used here is simulated, errors due to noise and complex light environments are non-existent, therefore any two IP correspondence pairs can be used[‡]. Once illuminant positions have been estimated in 2D for every camera, the system proceeds to convert them into 3D rays. The system can not reverse project a 2D illuminant into an absolute 3D position directly due to the nature of the reverse transformation, therefore rays are obtained by transforming the x and y coordinates at different depths. The chosen depths may be arbitrary but for greatest accuracy the near and far planes are chosen. The conversion from 3D to 2D is multi-part. First the NDC coordinates are calculated:

$$\hat{R}_{\text{ndc}} = \begin{bmatrix} \frac{(\hat{T}_x \cdot 2) - V_0}{V_2} - 1 \\ \frac{(\hat{T}_y \cdot 2) - V_1}{V_3} - 1 \\ 2 \cdot \hat{d} - 1 \\ 1 \end{bmatrix} \tag{5.30}$$

Where $\hat{d}$ indicates the chosen depth, which in this case is either the value of the near or far planes. The following two calculations are performed for both results of the above equation. The NDC coordinates are converted into world coordinates:

$$\hat{R}_{\text{world}} = \hat{R}_{\text{ndc}} \cdot I \tag{5.31}$$

Where I is the combined modelview and projection inverted matrices relating to the current camera, as calculated above. The world coordinates are finally scaled by the w component to produce the illuminant ray point:

$$\frac{\hat{R}_{\text{world}}}{\hat{R}_{\text{world}_w}} \tag{5.32}$$

The whole reverse projection process is repeated for the data relating to each camera and associated 2D illuminant points. Once the points relating to two or more rays have been obtained the 3D illuminant is found. The closest points on each ray are discovered as follows:

[‡] To improve accuracy the average intersection of multiple pairs may be taken, but this is not dealt with here

$$\hat{U}_u = \hat{b} - \hat{a} \tag{5.33}$$
$$\hat{U}_v = \hat{d} - \hat{c} \tag{5.34}$$
$$\hat{U}_w = \hat{a} - \hat{c} \tag{5.35}$$
$$\hat{U}_a = \hat{u} \cdot \hat{u} \tag{5.36}$$
$$\hat{U}_b = \hat{u} \cdot \hat{v} \tag{5.37}$$
$$\hat{U}_c = \hat{v} \cdot \hat{v} \tag{5.38}$$
$$\hat{U}_d = \hat{u} \cdot \hat{w} \tag{5.39}$$
$$\hat{U}_e = \hat{v} \cdot \hat{w} \tag{5.40}$$
$$\hat{U}_D = \hat{a} \cdot \hat{c} - \hat{b} \cdot \hat{b} \tag{5.41}$$

Where the above $\hat{U}$ values are intermediary. They are used to calculate the closest positions along the two rays:

$$sc = \frac{\hat{U}_b \cdot \hat{U}_e - \hat{U}_c \cdot \hat{U}_d}{\hat{U}_D} \tag{5.42}$$
$$tc = \frac{\hat{U}_a \cdot \hat{U}_e - \hat{U}_b \cdot \hat{U}_d}{\hat{U}_D} \tag{5.43}$$

The closest points between each rays are then calculated:

$$L_0 = \hat{a} + (\hat{t}_{\hat{c}} \cdot \hat{U}_v) \tag{5.44}$$
$$L_1 = \hat{c} + (\hat{s}_{\hat{c}} \cdot \hat{U}_v) \tag{5.45}$$

Once (5.44) and (5.45) are evaluated L will represent a line between the closest points on each line. The illuminant, R, lies at the centre of this line and is obtained as below:

$$\hat{R} = L_0 - (L_0 - L_1) \cdot 0.5 \tag{5.46}$$

The estimation of illuminant position, R, can then be compared against the original position, $\vec{N}$. The above equations describe the entire mathematical model and represent the following actions:

- Data input
- Rotation matrix construction
- Translation matrix construction
- Modelview matrix creation
- Projection matrix construction
- Inverse matrix construction

- Creation of cube vertices
- Ground plane simulation
- Simulation of interest points
- Location of 2D illuminant position
- Reverse project 2D positions into rays
- Detection of closest points on 3D rays
- Location of 3D illuminant position

Camera locations and associated parameters are manually specified when numerically simulating the technique, however in a real world situation this information would be available via output from a geometric registration technique. The mathematics outline the core functionality of the proposed technique but do not include error mitigation strategies. Additional functionality is added to deal with varying circumstances in the framework and prototypes discussed in the remainder of this chapter.

5.2 Detection Framework

Core functionality is implemented within a C++ API library that is comprised of a series of modules and classes which include:

- CVector
- CCorrespondence
- CCamera
- CLog
- CIlluminantDetection
- CImageProcessing
- CInterestPointManager

The architecture and class structure is shown in unified modeling language (UML) within figure 5.1. Only technique specific functionality is implemented and external processes are left to the individual implementation of the application. Image processing is handled within the *CImageProcessing* module. This class implements low level techniques such as discussed in section 3.2. The 2D and interest point functionality is dealt with within the *CInterestPointManager*

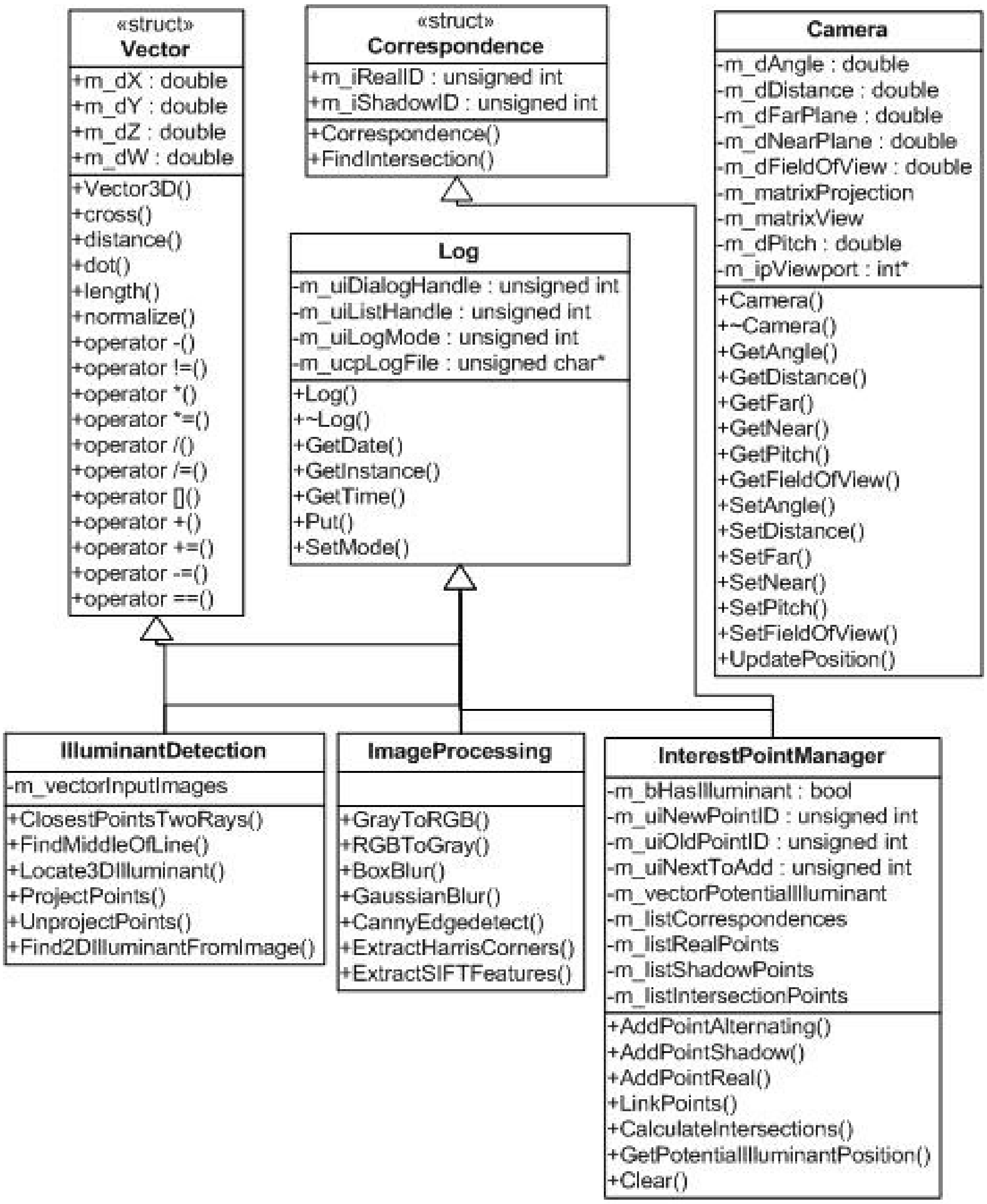


Figure 5.1: UML Class Diagram: Detection Framework

class. The 3D estimation is performed within *CIlluminationDetection*. Additional classes provide supporting roles such as basic mathematical, debugging and processing capability.

This code segment locates an illuminant in two dimensions:

```
for(unsigned int i=0; i<m_vCorrespondences.size(); i++)
```

```
{
for(unsigned int j=i+1; j<m_vCorrespondences.size(); j++)
{
Point a1 = m_vShadowPoints[m_vCorrespondences[i].s-1];
Point a2 = m_vRealPoints  [m_vCorrespondences[i].r-1];

Point b1 = m_vShadowPoints[m_vCorrespondences[j].s-1];
Point b2 = m_vRealPoints  [m_vCorrespondences[j].r-1];

Point p = Correspondence::FindIntersection( a1, a2,   b1, b2);

        /* checks to detect backward intersects omitted */
m_vIntersectionPoints.push_back( p );

}
}
```

The correspondence point intersections are located within the FindIntersections function as shown in the following code segment:

```
float ax = static_cast<float>(a1.x); //float cast,very slow !
float ay = static_cast<float>(a1.y);

float bx = static_cast<float>(a2.x);
float by = static_cast<float>(a2.y);

float cx = static_cast<float>(b1.x);
float cy = static_cast<float>(b1.y);

float dx = static_cast<float>(b2.x);
float dy = static_cast<float>(b2.y);

float temp_divide, temp_x, temp_y;

// solve for x and y
temp_divide = 1.0f / ((ax-bx)*(cy-dy) - (ay-by)*(cx-dx));
temp_x = (((ax*by)-(ay*bx))*(cx-dx) - (ax-bx)*(cx*dy-cy*dx)) * temp_divide;
temp_y = (((ax*by)-(ay*bx))*(cy-dy) - (ay-by)*(cx*dy-cy*dx)) * temp_divide;

// account for lack of rounding in reverse cast
temp_x += 0.5f;
temp_y += 0.5f;
```

```
return Point( static_cast<int>(temp_x), static_cast<int>(temp_y) );
```

The code below shows the algorithm that detects the three dimensional illumination coordinates from multiple illuminant rays:

```
#define SMALL_NUM 0.00000001
Vector3 u = L1.P1 - L1.P0;
Vector3 v = L2.P1 - L2.P0;
Vector3 w = L1.P0 - L2.P0;

float a = u.dot( u );
float b = u.dot( v );
float c = v.dot( v );
float d = u.dot( w );
float e = v.dot( w );

float D = a * c - b * b;
float sc, tc;

if( D < SMALL_NUM ) // Parallel
{
    sc = 0.0f;
    tc = ( b > c ? d / b : e / c );
} else // Not parallel
{
    // Closest time index (L1)
    sc = ( b * e - c * d ) / D;
    // Closest time index (L2)
    tc = ( a * e - b * d ) / D;
}

Vector 3 dP = w + ( sc * u ) / D;

dP.normalize(); // Closest distance

// W.P0 is closest point on L1
// W.P1 is closest point on L2
W = w + ( sc * u ) - ( tc * v );
```

As parallel and identical lines present a problem they are detected prior to performing the intersection in the above code. The type of error is returned and the application is able to respond using the relevant error mitigation technique as discussed in section 4.3. The centre of the line formed by the detected points is found as below:

```
Vector3 FindMiddleOfLine( Vector3 A0, Vector3 A1 )
{
return A0 - (( A0 - A1 ) * 0.5f) ;
}
```

The following code show the reverse projection technique that converts a 2D illuminant position into a 3D point.

```
   double m[16], A[16];
   double tempA[4], tempB[4];

   tempA[0] = (inputx - viewport[0]) * 2 / viewport[2] - 1.0;
   tempA[1] = (inputy - viewport[1]) * 2 / viewport[3] - 1.0;
   tempA[2] = 2 * inputz - 1.0;
   tempA[3] = 1.0;

   inv = invert(proj * model);

   tempB = inv * tempA;

   /* sanity checks omitted here */

   // Divide by w
   outputX = tempB[0] / tempB[3];
   outputY = tempB[1] / tempB[3];
   outputZ = tempB[2] / tempB[3];

   return 1;
}
```

This code is called twice in order to determine both points of the illuminant ray. The management of interest points, their classification, correspondence handling and association with individual cameras is handled within the *CInterestPointManager* class which contains a series of encapsulated linked list data structures. The API that the above classes form is built to be compiled into a dynamic link library (DLL) for inclusion at run-time and can be used by applications by calling the appropriate function when needed.

5.3 2D Detection Prototype

A prototype was developed to test the ability of the detection framework to locate an illuminant in two dimensions. The framework makes use of the detection and interest point functionality exposed by the API and is able to locate an illuminant in 2D when supplied an image of sufficient detail, and associated interest points which are loaded from file. Sequential image frame can be passed to the application in order to simulate a moving illuminant, camera or scene geometry. The illuminant location is output to the console via the stdout socket.

The below features are implemented in this prototype:

- Image input
- Geometric registration
  - Rotation matrix construction
  - Translation matrix construction
  - Modelview matrix creation
  - Projection matrix construction
  - Inverse matrix construction
- Location of 2D illuminant position

Figure 5.2 shows the detection of a 2D illuminant position when given two IP correspondence lines. It should be noted that in this image these lines are forced so that the detected position lies within the image boundary for visualization purposes. It should be noted that during normal operation this is rarely the case.

5.4 3D Detection Prototype

A further prototype was developed that makes use of multiple input images in order to achieve full 3D detection. Associated interest points are observed from multiple simultaneous views. This application performs the same actions as within the 2D version, repeated for each camera view. The results are then combined and the illuminant is located three dimensionally. This prototype assumes two input cameras as can be seen in the left two panes of figure 5.3. The prototype detects the illuminant in three dimensions and outputs the estimated location to the console. Additionally, it aims to augment reality using the acquired illumination metrics in order to achieve basic illumination consistency. To this end an artificial light source is positioned at the geometrically registered estimation coordinates and is used to both illuminate the artificial object and cause cast shadows of correct appearance. This prototype highlights the conditions under which the effectiveness of the technique is reduced. Such conditions are discussed

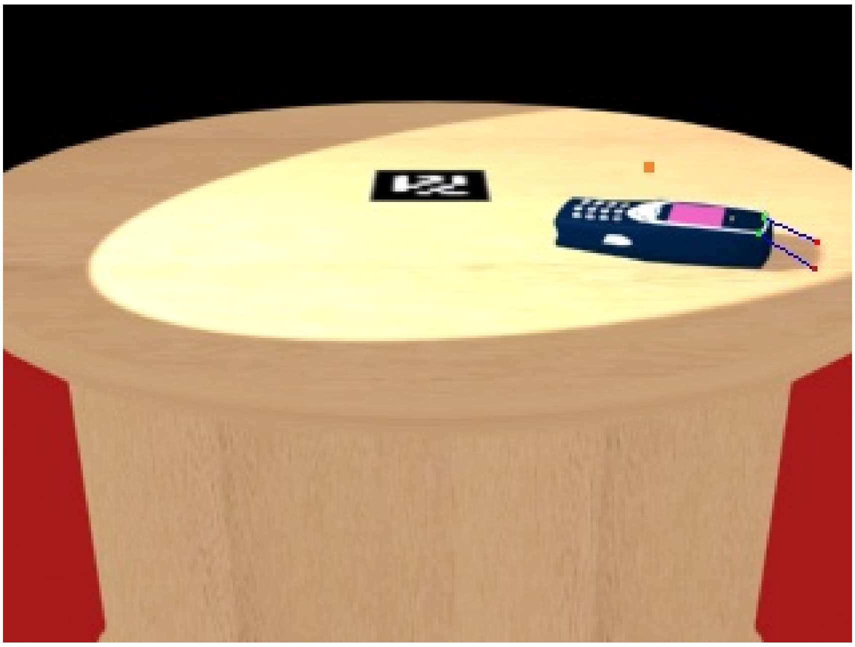


Figure 5.2: 2D Illuminant Detection

in chapter 6. This prototype is able to make use of static or sequential input. Augmentation takes place in real-time achieving a consistent frame-rate of 60 frames per second (FPS). Figure 5.3 shows the 3D prototype application. The following summarizes the functionality implemented in this prototype:

- Image input
- Geometric registration
  - Rotation matrix construction
  - Translation matrix construction
  - Modelview matrix creation
  - Projection matrix construction
  - Inverse matrix construction
- Reverse project 2D positions into rays
- Detection of closest points on 3D rays
- Location of 3D illuminant position

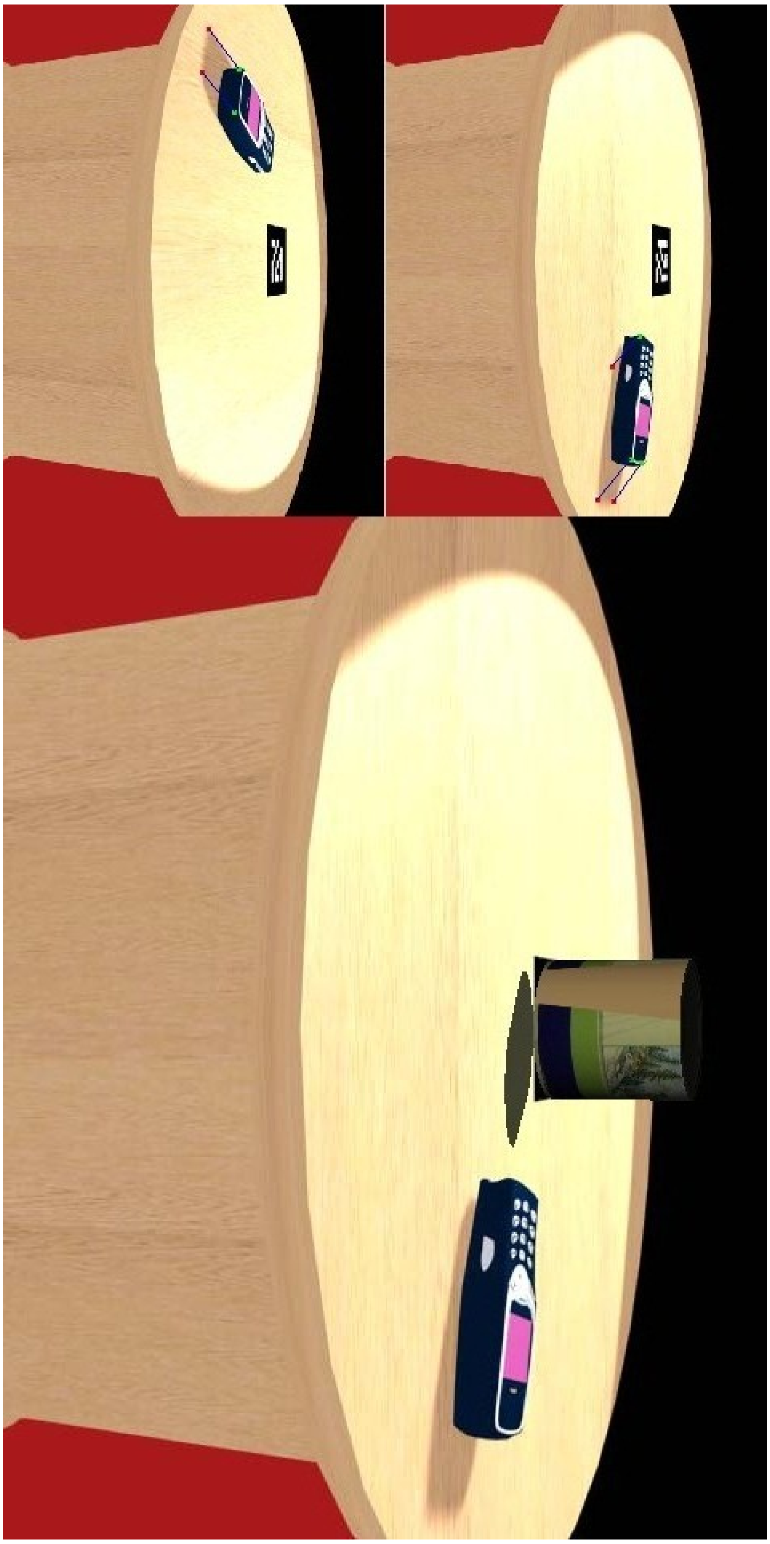


Figure 5.3: Fully Registered Augmentation

5.5 Tracking Simulation

The simulation prototype considers the full mathematical model and enables dynamic generation of artificial interest points in order to enable photometric tracking capability. It simulates real multi-view scenes by creating interest point locations based on pseudo-rendered geometry and calculated shadow corners. These 3D interest points are observed by two different virtual cameras and are projected into 2D image space for analysis. This allows for the dynamic modification of the scene in ways that would be of high computational cost and would be time consuming with real world imagery. The movement of scene entities, such as the camera and illuminant, can be scripted. This allows for the rapid visualization of multiple configurations. The main application window is shown in figure 5.4.

The top left pane shows a 3D perspective view of the *simulated real* scene. The yellow sphere represents the location of the real illuminant and the box represents real scene geometry. The bottom left pane shows the illuminant estimation for the current configuration. Note the grey sphere is a ghosted representation of the real position. Any disparity between the real and virtual illuminant coordinated can be seen here. In this pane the red coloured sphere represents the virtual illuminant. The top right and bottom right panes show the view of the first and second camera devices respectively. These two views form the images to be analyzed. Positioned to the right of these OpenGL panes is a Microsoft Windows based user interface that allows for the configuration of the application, scene and simulation parameters as shown in figure 5.5. This pane also displays information about the state of the application, the position of illuminants both simulated and detected. Intermediary variables such as 2D illuminant pixel locations are also shown. Figures 5.6, 5.7, fig:Track-SameImages and fig:Track-SameLine show the simulation in various states.

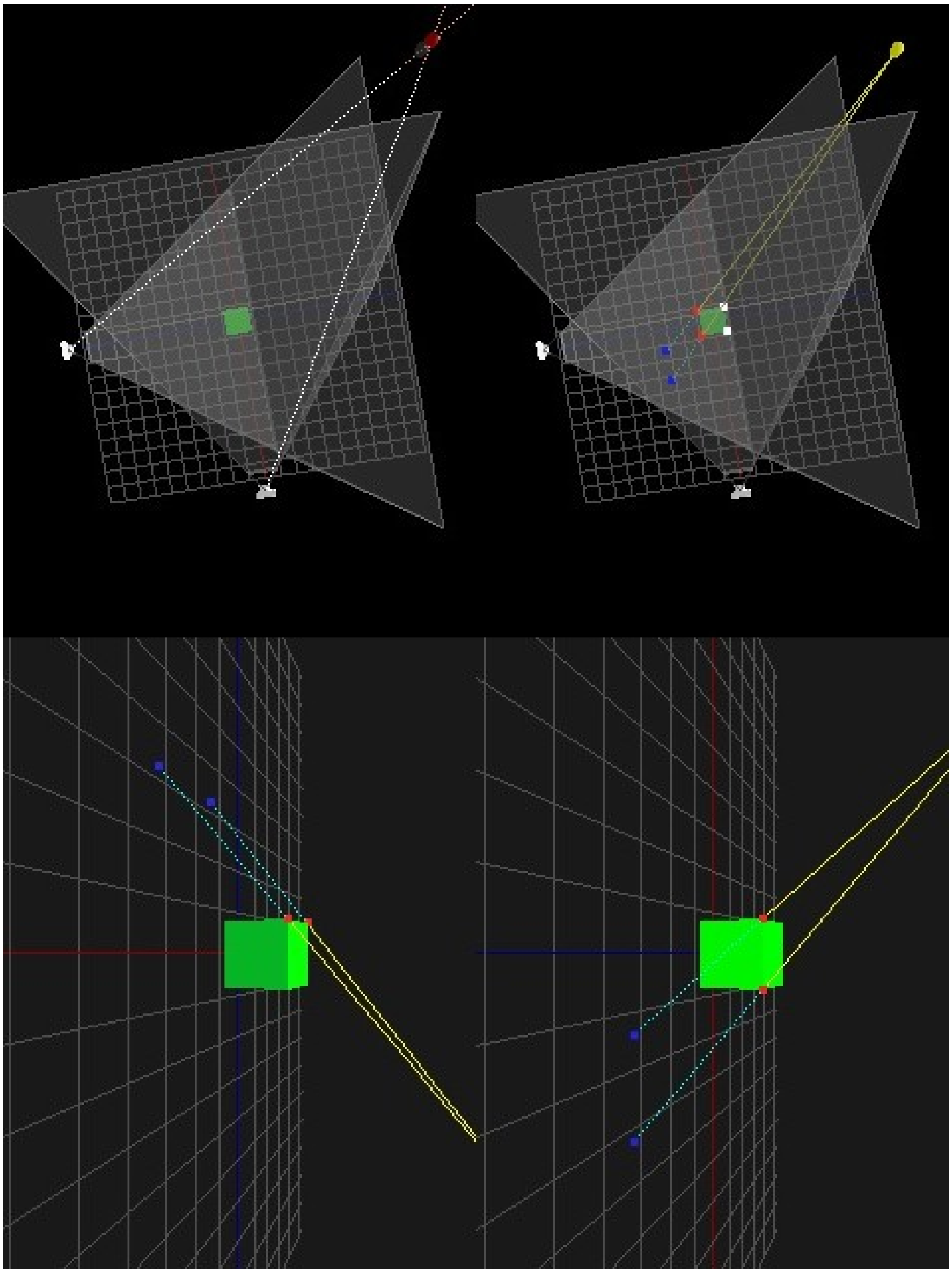
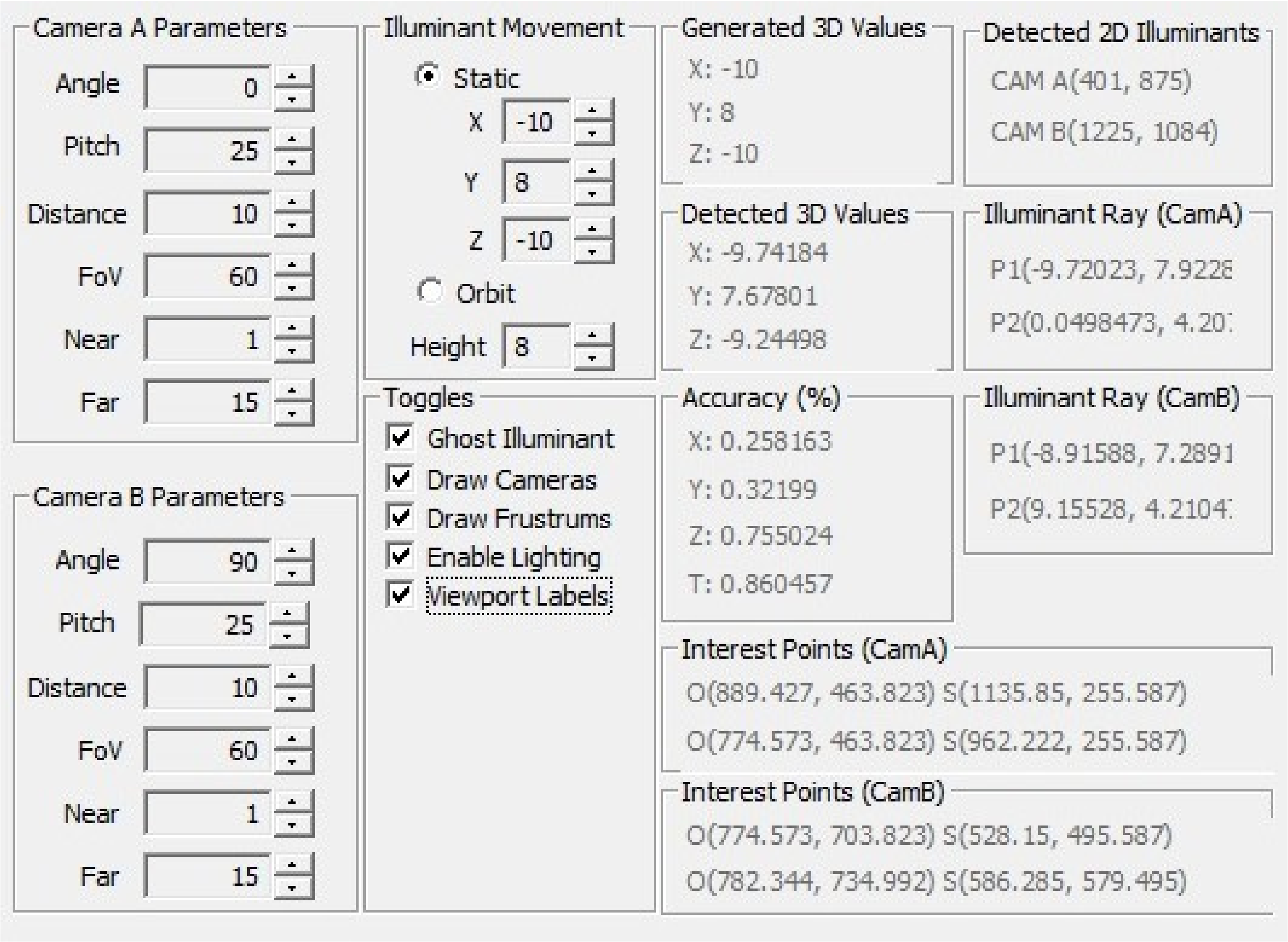


Figure 5.4: Tracking Simulation



Camera A Parameters

Angle 0

Pitch 25

Distance 10

FoV 60

Near 1

Far 15

Camera B Parameters

Angle 90

Pitch 25

Distance 10

FoV 60

Near 1

Far 15

Illuminant Movement

● Static

X -10

Y 8

Z -10

○ Orbit

Height 8

Toggles

☑ Ghost Illuminant

☑ Draw Cameras

☑ Draw Frustrums

☑ Enable Lighting

☑ Viewport Labels

Generated 3D Values

X: -10

Y: 8

Z: -10

Detected 3D Values

X: -9.74184

Y: 7.67801

Z: -9.24498

Accuracy (%)

X: 0.258163

Y: 0.32199

Z: 0.755024

T: 0.860457

Detected 2D Illuminants

CAM A(401, 875)

CAM B(1225, 1084)

Illuminant Ray (CamA)

P1(-9.72023, 7.922

P2(0.0498473, 4.20

Illuminant Ray (CamB)

P1(-8.91588, 7.2891

P2(9.15528, 4.2104

Interest Points (CamA)

O(889.427, 463.823) S(1135.85, 255.587)

O(774.573, 463.823) S(962.222, 255.587)

Interest Points (CamB)

O(774.573, 703.823) S(528.15, 495.587)

O(782.344, 734.992) S(586.285, 579.495)

Figure 5.5: Tracking Configuration and Information

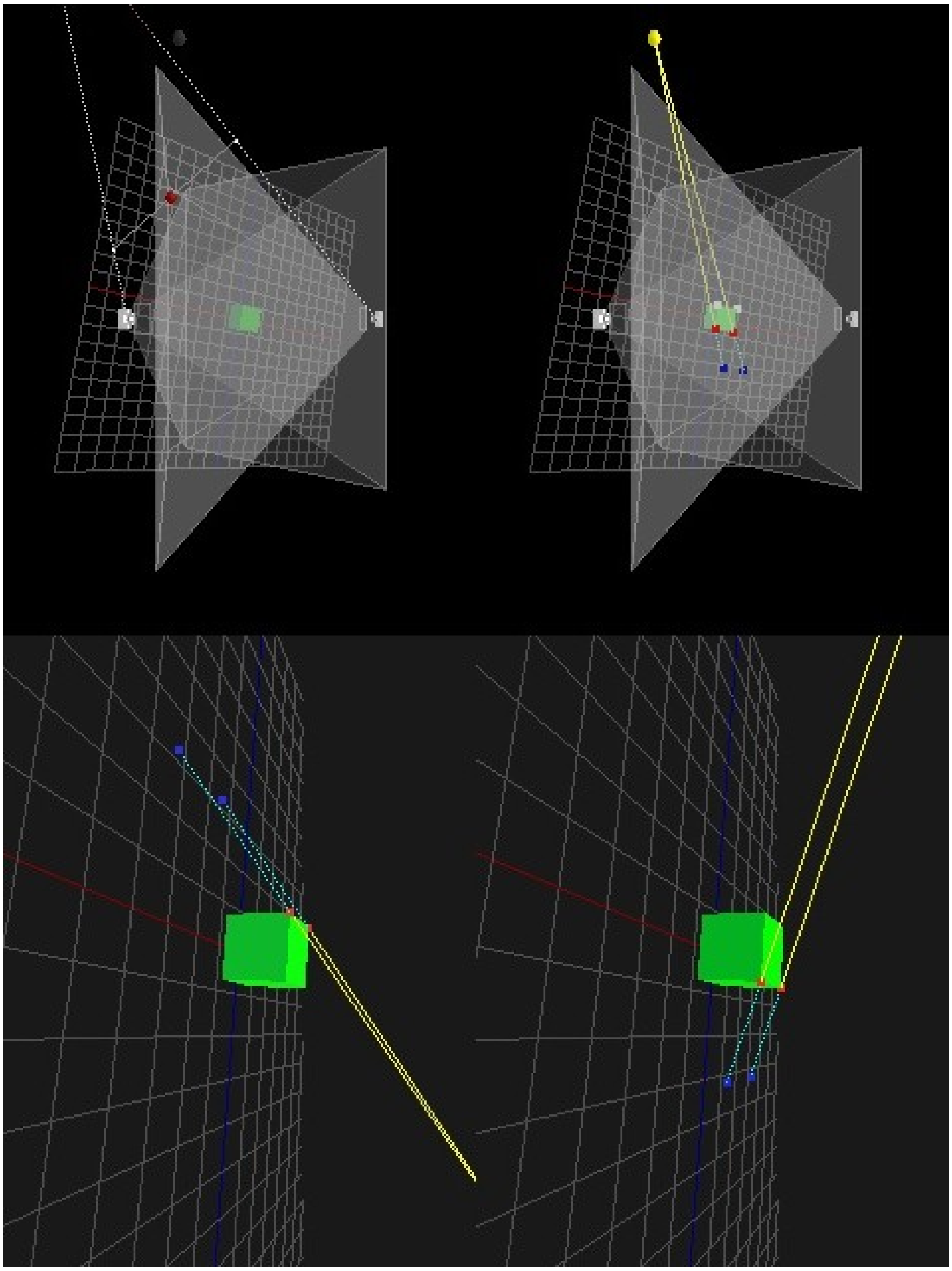


Figure 5.6: Directly Opposing Camera Problem

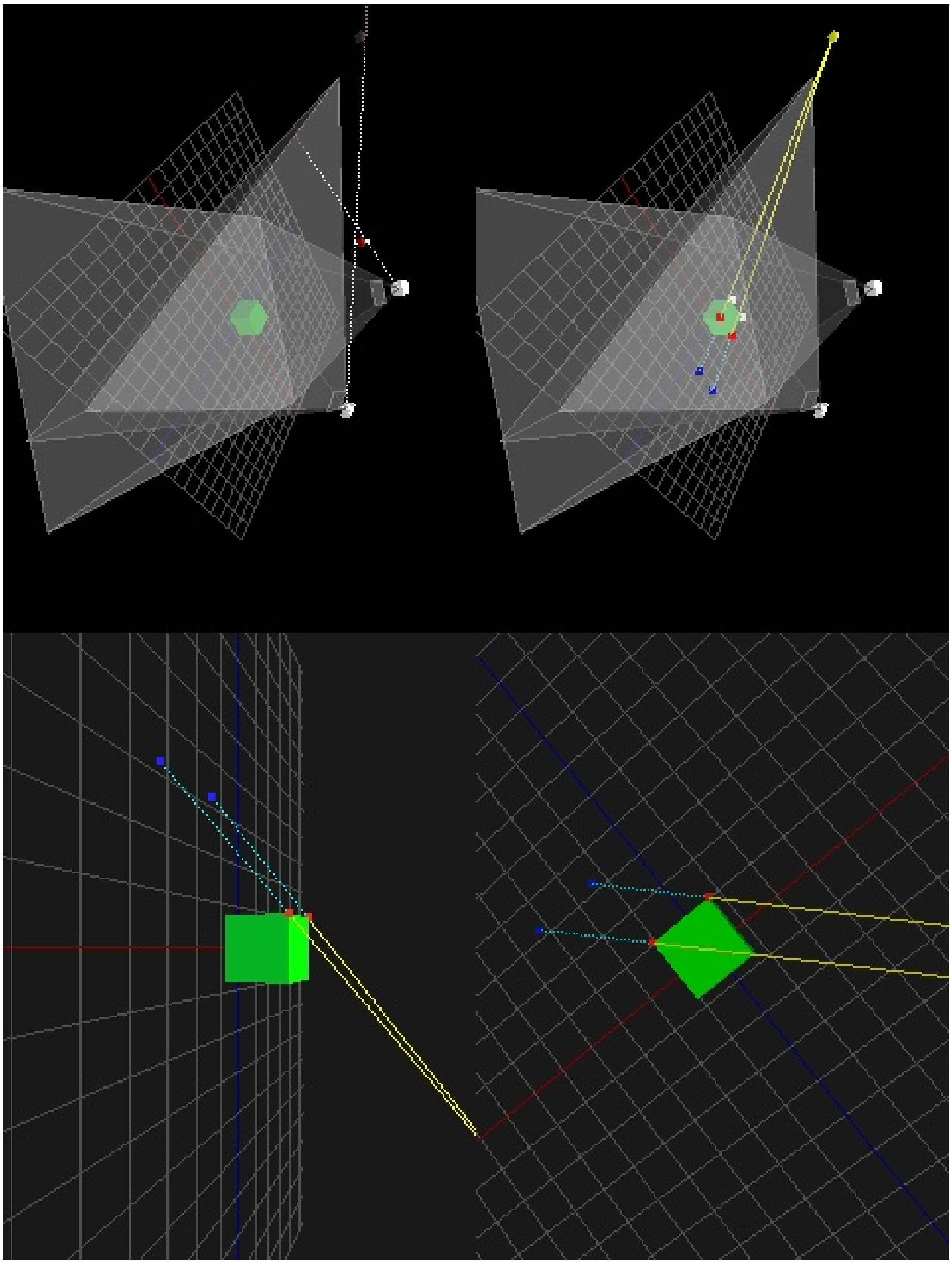


Figure 5.7: Parallel Line Problem

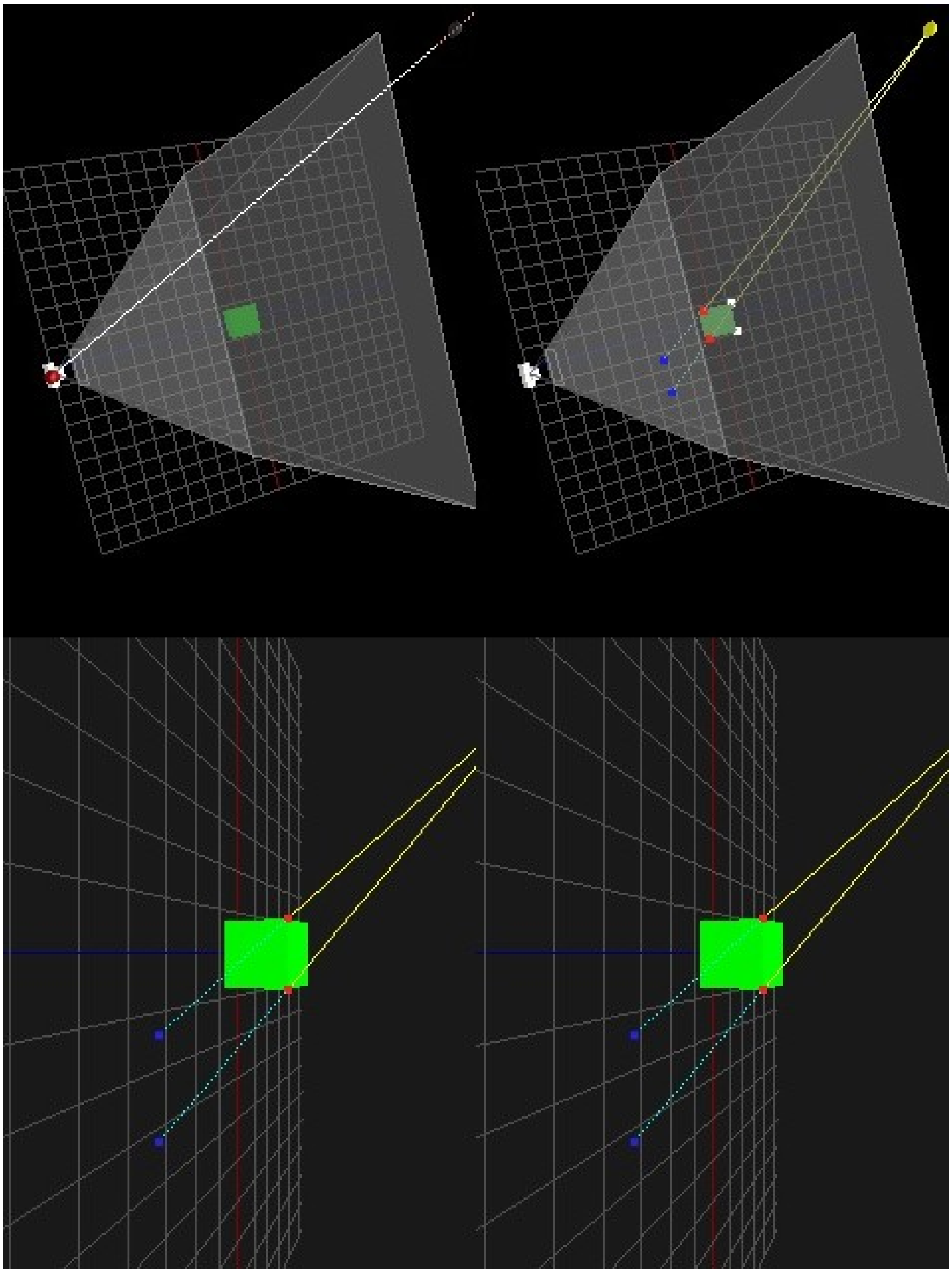


Figure 5.8: Same Image Problem

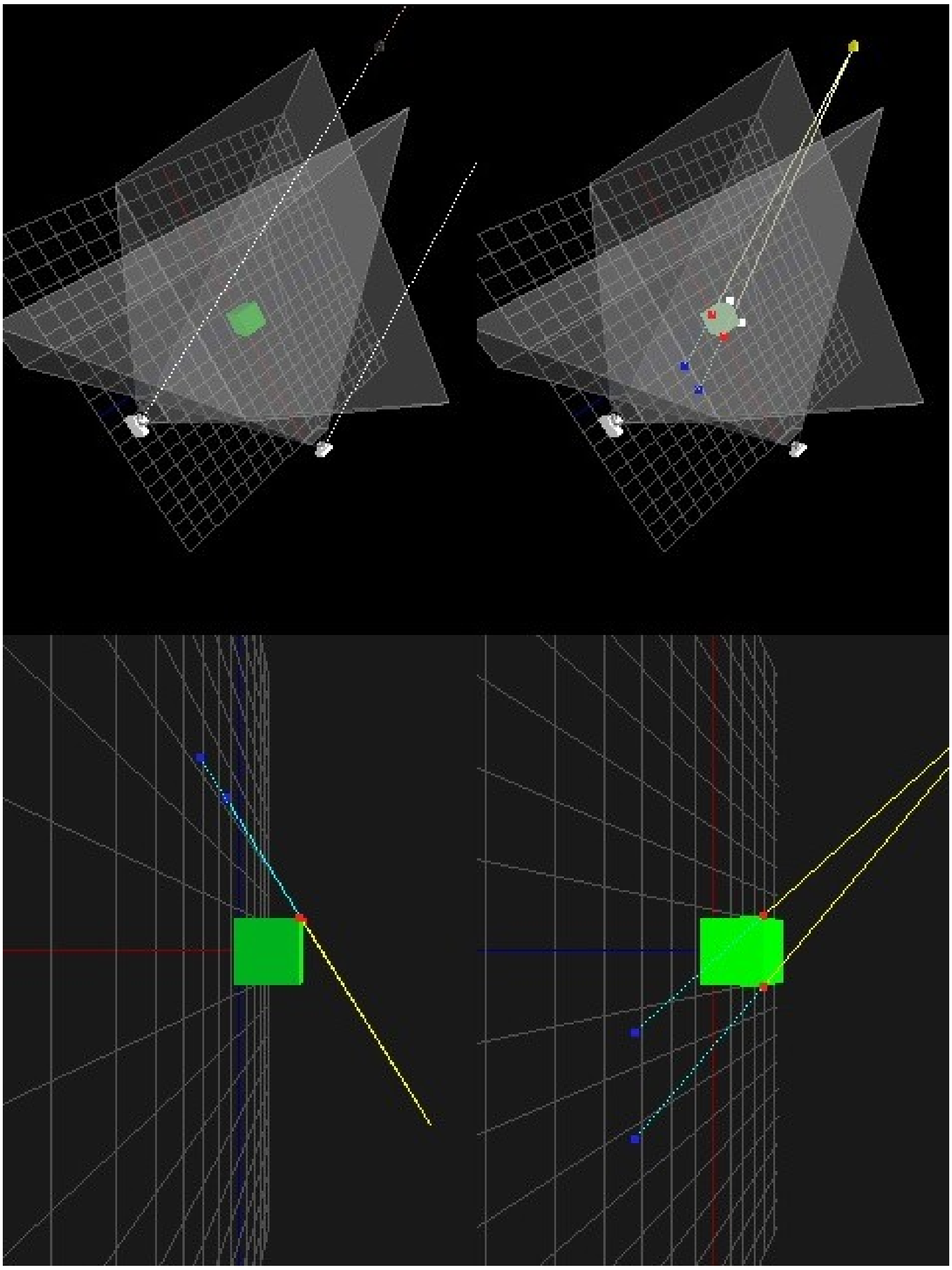


Figure 5.9: Same Line Problem

Chapter 6

Evaluation

6.1 Technique Verification

The proposed technique has been extensively evaluated, with a focus on functionality and performance. The results obtained from experimentation and simulation are comparable with other photometric registration techniques as was discussed in section 3.1. Technique functionality is confirmed visually, numerically and graphically and environmental changes are considered in order to establish the effect of variant operational conditions. An extensive data set has been generated as a result of tests that simulate the response of the prototype to changes in both environment and system configuration such data includes imagery similar to as shown in figures 6.2, 6.3 and 6.4. Mathematical models were used for the majority of experimentation, however a large quantity of input images were created via 3D rendering in addition to this. Images with conditions as shown in figures 6.2, 6.3 and 6.4 were considered and an extensive 8GB image set[*] was generated in order to carry out trials.

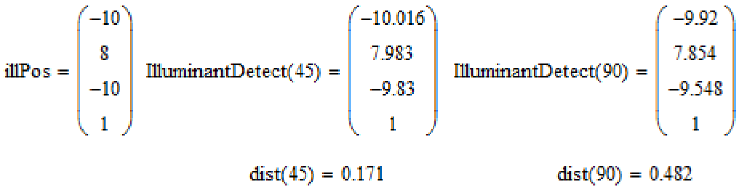
The observation and visual display of expected results facilitated the verification of technique's ability to augment reality and perform photometric simulation using correctly photometrically registered virtual illumination conditions. The numerical modeling and observation of results that are within a certain acceptable threshold of the real or pseudo-real[†] illuminant position has provided numeric technique verification. The graphing of output data has facilitated the determination of optimal and failure environmental conditions and configurations via analysis of high and low error regions. Graphical data is cross-referenced with visual results in order to gain a fuller understanding of a given simulation.

The mathematical model was produced in order to determine technique feasi-

[*] In the form of Autodesk 3DS Max renders

[†] The simulated illuminant position

bility and was used to simulate the complete process of illuminant discovery when using two input data sets as was discussed in section 5.1. Figure 6.1 shows the real illuminant position in numeric form, and two output results. This output was generated by the simulation as configured to make use of simulated shadow and object interest points that are produced by the simulated geometry of a simple cube object.



$$\mathrm{illPos} = \begin{pmatrix} -10 \\ 8 \\ -10 \\ 1 \end{pmatrix} \quad \mathrm{IlluminantDetect}(45) = \begin{pmatrix} -10.016 \\ 7.983 \\ -9.83 \\ 1 \end{pmatrix} \quad \mathrm{IlluminantDetect}(90) = \begin{pmatrix} -9.92 \\ 7.854 \\ -9.548 \\ 1 \end{pmatrix}$$

$$\mathrm{dist}(45) = 0.171 \qquad \mathrm{dist}(90) = 0.482$$

Figure 6.1: MathCAD Initial Results

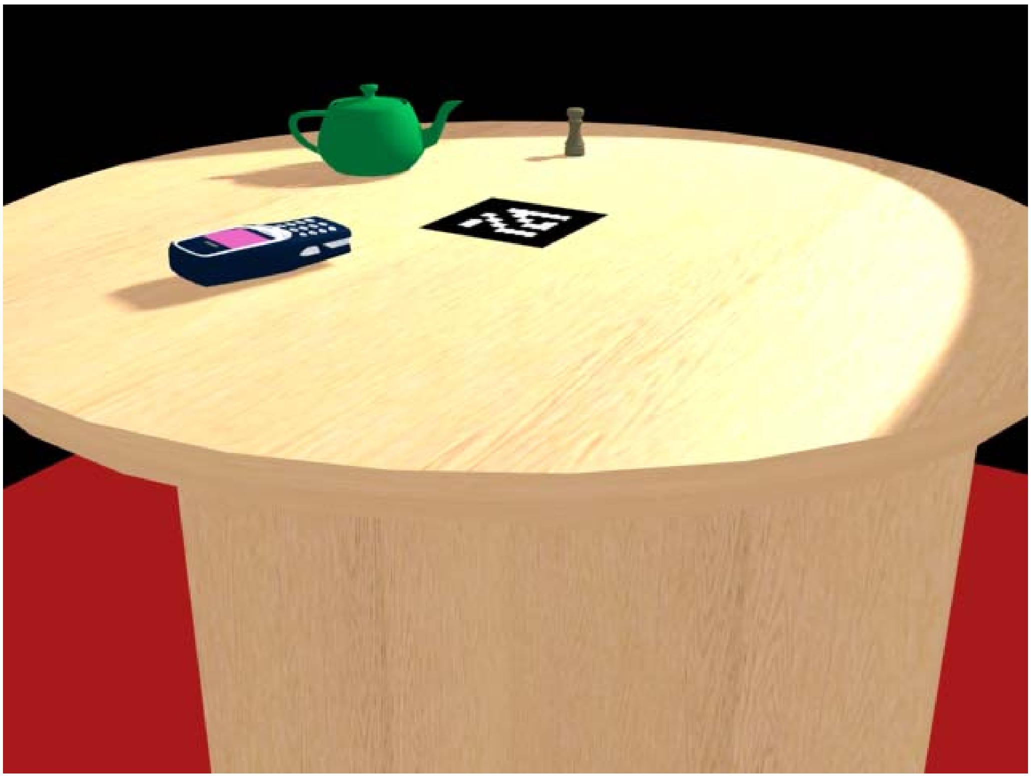


Figure 6.2: Sample Input Data 1

The parameter given to the two functions *Illuminant Detect* and *Dist* represent the difference in angle between the two simulated camera devices. The exact

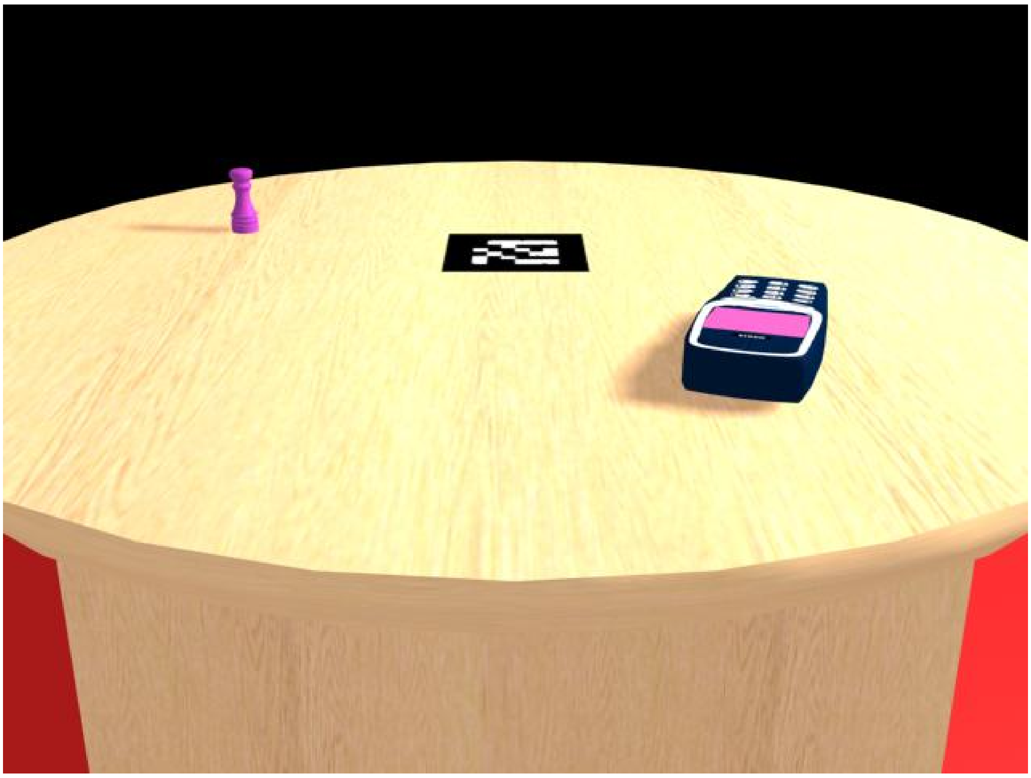


Figure 6.3: Sample Input Data 2

results with the same configuration can be expected to vary depending on the angle of observation and are subject to a very slight rounding error, the magnitude of which varies slightly between implementation and underlying computing architecture. The overall error is calculated in the form of distance between the real and detected illuminant positions.

The results above show that the technique is able to approximate the real illuminant position with reasonable accuracy when a difference of 45° and 90° exists between the two camera angles. Figure 6.1 was obtained by performing the simulation using a pixel resolution of 640x480. The process was repeated a number of times in order to determine the accuracy at different simulated image resolutions. Tables 6.1 and 6.2 show the results, where the real-illuminant position is $[-10 \quad 8 \quad -10]$, for the camera angle differences of 45° and 90° respectively.

The visual simulation was then configured as above in order to display technique behavior under these conditions; this resulted in figures 6.5 and 6.6. These figures represent input scenes when dealing with a camera angle difference of 45° and 90° respectively.

The results show that in this instance the technique was able to achieve less

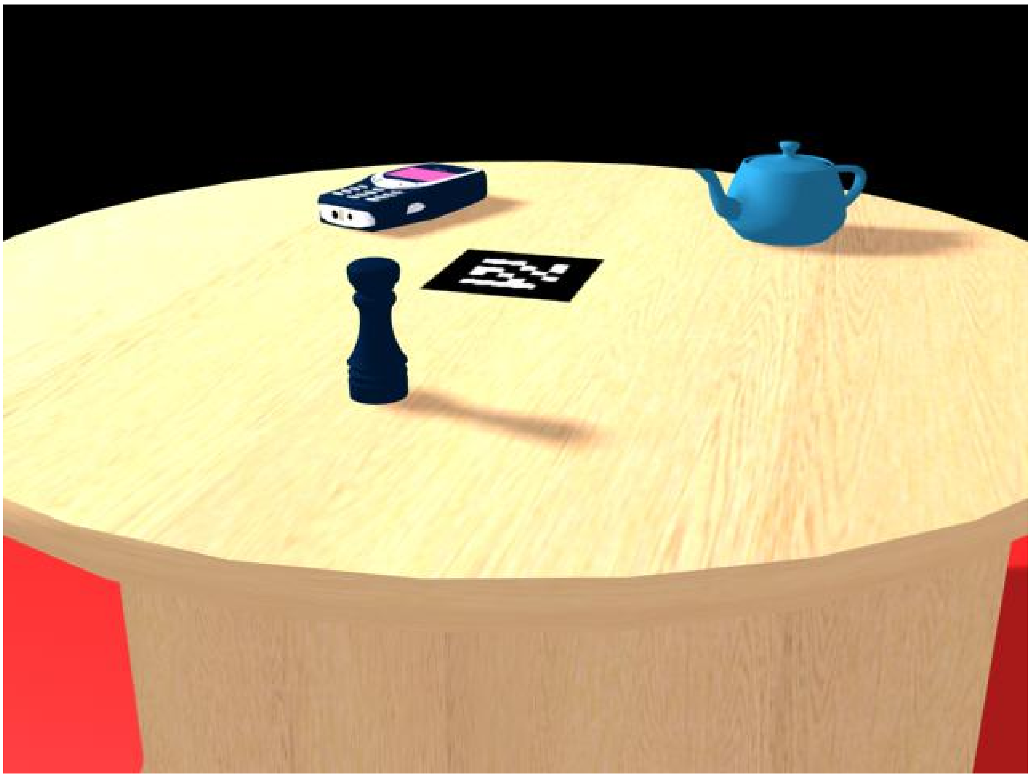


Figure 6.4: Sample Input Data 3

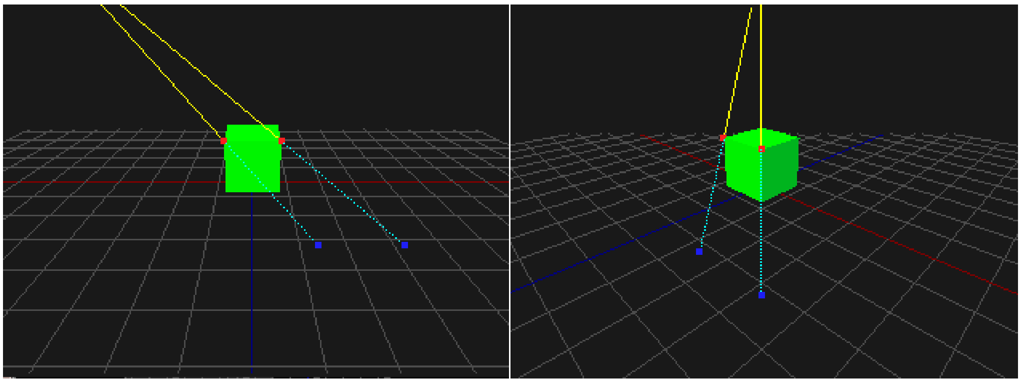


Figure 6.5: Visual Input (45°Δ)

error when observing the pseudo-real scene geometry at angles that differ by 45° than when differing by 90°, converse to as was expected. As it can be expected that results depend on the scene being observed and from viewing the input images is it possible to theorize as to why this is the case. When viewing the 45° input set it is apparent that the angle of the correspondence lines, in both images,

Resolution	Aspect	Detected Position			Error
320x240	1.333	[−10.554	8.269	− 10.276]	0.675
640x480	1.333	[−10.000	7.914	− 9.813]	0.206
800x600	1.333	[−10.503	8.190	− 10.231]	0.585
1024x768	1.333	[−9.585	7.776	− 9.469]	0.710
1152x864	1.333	[−10.216	8.072	− 9.995]	0.228
1280x960	1.333	[−10.243	8.067	− 10.018]	0.253
1600x1200	1.333	[−10.062	8.020	− 9.871]	0.145
2048x1536	1.333	[−10.016	7.983	− 9.830]	0.171
3200x2400	1.333	[−9.922	7.908	− 9.748]	0.279
4000x3000	1.333	[−10.015	7.949	− 9.828]	0.180
6400x4800	1.333	[−10.089	7.991	− 9.889]	0.143
852x480	1.777	[−10.000	7.914	− 9.813]	0.206
1280x720	1.777	[−10.187	8.059	− 9.970]	0.199
1365x768	1.777	[−10.372	8.136	− 10.139]	0.420
1600x900	1.777	[−9.963	7.951	− 9.786]	0.223
1920x1080	1.777	[−10.262	8.104	− 10.034]	0.284
1440x900	1.6	[−9.963	7.951	− 9.786]	0.223
1680x1050	1.6	[−10.422	8.114	− 10.163]	0.466
1920x1200	1.6	[−10.062	8.02	− 9.871]	0.145
2560x1600	1.6	[−10.166	8.022	− 9.953]	0.174
3840x2400	1.6	[−9.922	7.908	− 9.748]	0.279
7680x4800	1.6	[−10.089	7.991	− 9.889]	0.143

Table 6.1: Results: Variant Resolution (45°Δ)

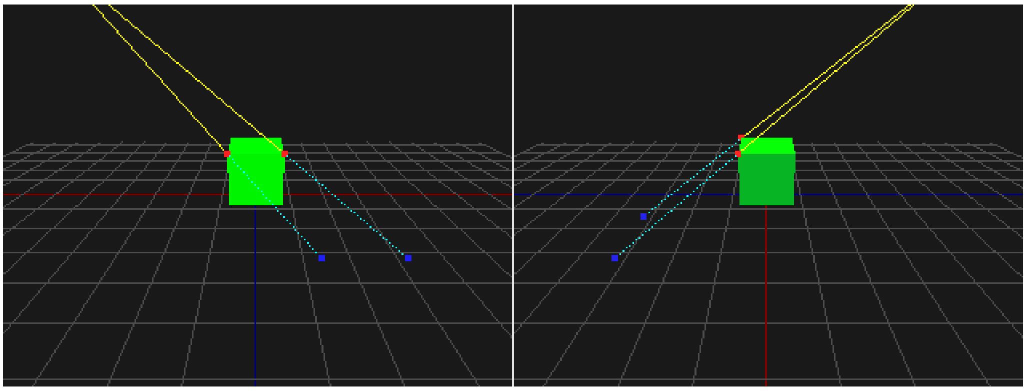


Figure 6.6: Visual Input (90°Δ)

is such that intersection is not as easily determinable. With the 90° image set it can be seen that the line gradients are more subtle and therefore the intersection location is less clear. This is especially true for lower resolution input images,

Resolution	Aspect	Detected Position	Error
320x240	1.333	[−13.737 10.625 − 13.814]	5.950
640x480	1.333	[−9.163 7.405 − 8.819]	1.565
800x600	1.333	[−12.029 9.147 − 11.610]	2.833
1024x768	1.333	[−11.162 8.723 − 11.083]	1.745
1152x864	1.333	[−10.704 8.305 − 10.274]	0.815
1280x960	1.333	[−10.317 8.088 − 9.866]	0.355
1600x1200	1.333	[−9.898 7.852 − 9.514]	0.518
2048x1536	1.333	[−9.920 7.854 − 9.548]	0.482
3200x2400	1.333	[−10.536 8.206 − 10.209]	0.611
4000x3000	1.333	[−10.288 8.066 − 9.916]	0.307
6400x4800	1.333	[−10.069 7.923 − 9.667]	0.348
852x480	1.777	[−9.163 7.405 − 8.819]	1.565
1280x720	1.777	[−11.394 8.787 − 11.043]	1.910
1365x768	1.777	[−10.690 8.276 − 10.207]	0.771
1600x900	1.777	[−9.798 7.768 − 9.442]	0.637
1920x1080	1.777	[−10.068 7.923 − 9.612]	0.401
1440x900	1.6	[−9.798 7.768 − 9.442]	0.637
1680x1050	1.6	[−10.150 7.979 − 9.634]	0.396
1920x1200	1.6	[−9.898 7.852 − 9.514]	0.518
2560x1600	1.6	[−9.793 7.765 − 9.372]	0.702
3840x2400	1.6	[−10.536 8.206 − 10.209]	0.611
7680x4800	1.6	[−10.069 7.923 − 9.667]	0.348

Table 6.2: Results: Variant Resolution (90°Δ)

or when scene geometry is far away. In these situations the technique finds it difficult to differentiate between slightly angled and parallel correspondence lines. The results show that this is less of a problem when dealing with higher resolution imagery. In real situations where a higher availability of corresponding interest points exist such issues can be mitigated. For example, should this occur, the technique may make use of other correspondence lines, or factor in additional lines and calculate the average intersection point.

It is feasible to threshold a CL so that short lines containing less reliable information are omitted. Figure 6.7 shows the result of augmenting an image using such an input data-set. The input here is of low resolution and observes the scene with a 90° difference in angle. The image shows that once geometrically registered, the detected coordinates can be placed in context and are able to mimic real-world light environments. Therefore it can be observed that the technique is able to facilitate an improvement to augmented reality realism even despite some inaccuracies that are induced by low resolution input. It should also be noted that in some infrequent circumstances, increasing the resolution or moving closes to scene geometry does not reduce error. For example when the new configuration causes parallel or otherwise undesirable correspondence pairs for one or more input scenes. Once again, this can be mitigated using the steps described above. The average error is calculated from the data above to give the vector magnitude of 0.288 represented in unregistered coordinate units for the angle difference of 45°.

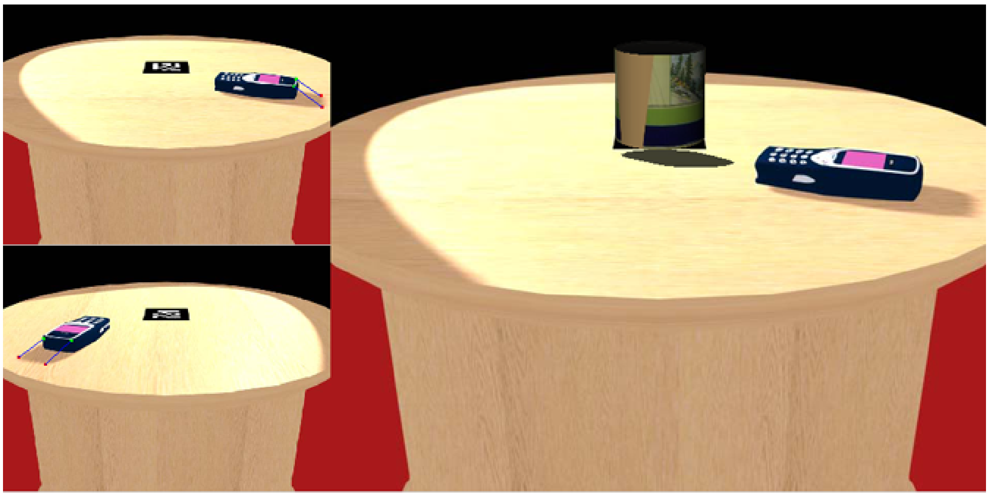


Figure 6.7: Example output

6.2 Optimal Configuration and Operating Conditions

An investigation into the difference that modifying the angle between input cameras makes to scene accuracy was undertaken using the mathematical model which was configured to generate output for a range of angle values. This value was the only variable and all other values remained constant. The graphs shown in figures 6.8, 6.9, 6.10, 6.11 and 6.12 were obtained from this experiment. They show the difference between camera angles versus error for a number of input resolutions. Again, error is defined as the distance between the real-world illuminant position and the detected virtual-world illuminant position. These figures show graphs produced from data obtained from imagery at commonly used image resolutions with the aspect ratio of 1.333. Figure 6.8 shows the worst results obtained under this configuration. It shows how low resolution imagery, such as images with the dimensions of 320 pixels by 240 pixels, are unsuitable for processing in this manner and proves the theory that such images do not contain sufficient information to allow for proper illuminant position estimation. All experiments discussed within this chapter have been conducted with such extreme resolutions providing results equally as erratic, therefore with the exception of figure 6.8 these superfluous graphs have been omitted. It would appear that low resolutions imagery increase the probability of special conditions arising that result in greater error within the illuminant detection process. If the imagery contains geometry that produces a perfect set of correspondence lines then low resolution may yield acceptable results, however such scenes occur rarely therefore it is worth omitting low resolution imagery from consideration in order to better achieve technique robustness.

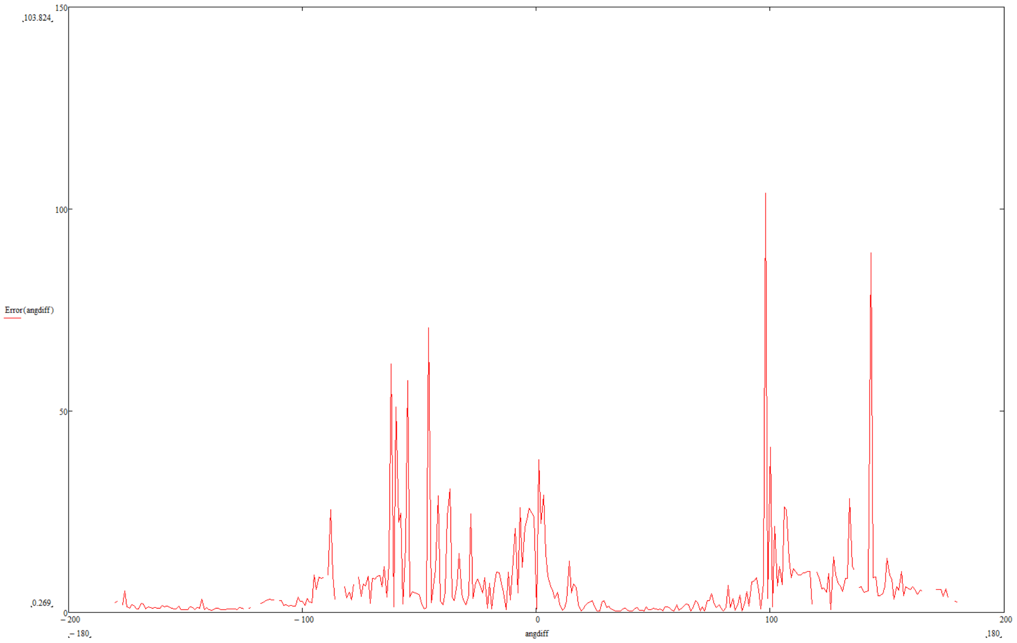


Figure 6.8: Angle Difference vs Error (320x240)

Figure 6.9 shows that greatly improved results can be obtained by increasing the resolution. From this figure it is easy to determine both the regions in which error is lowest and where error is most prevalent. In order to confirm these regions the data was sampled at a number of higher resolutions. Figures 6.10, 6.11 and 6.12 show the results of the same process using the same constants, sampled using images at resolutions of 1024x768, 2048x1536 and 6400x4800 respectively. By truncating the peaks at which error is greatest it is possible to view this data on the same scale as shown in figure 6.15. This figure shows the significant improvement of detection robustness. Fewer and thinner peaks of error are observed when error conditions are encountered when considering higher resolution input.

To verify that aspect ratio did not effect performance experimentation was repeated for a variety of aspect ratios. Figures 6.13 and 6.14 show the results of experimentation using the resolutions 1920x1080 and 7680x4800 which have the aspect ratio of 1.777 and 1.6 respectively.

An investigation into how the field of view of the camera effects the end

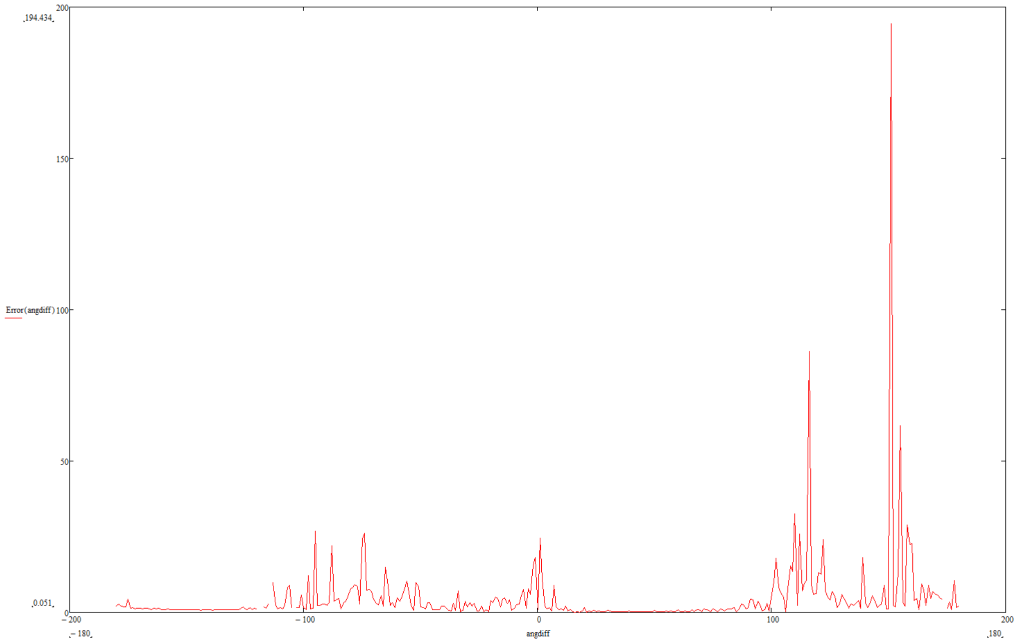


Figure 6.9: Angle Difference vs Error (640x480)

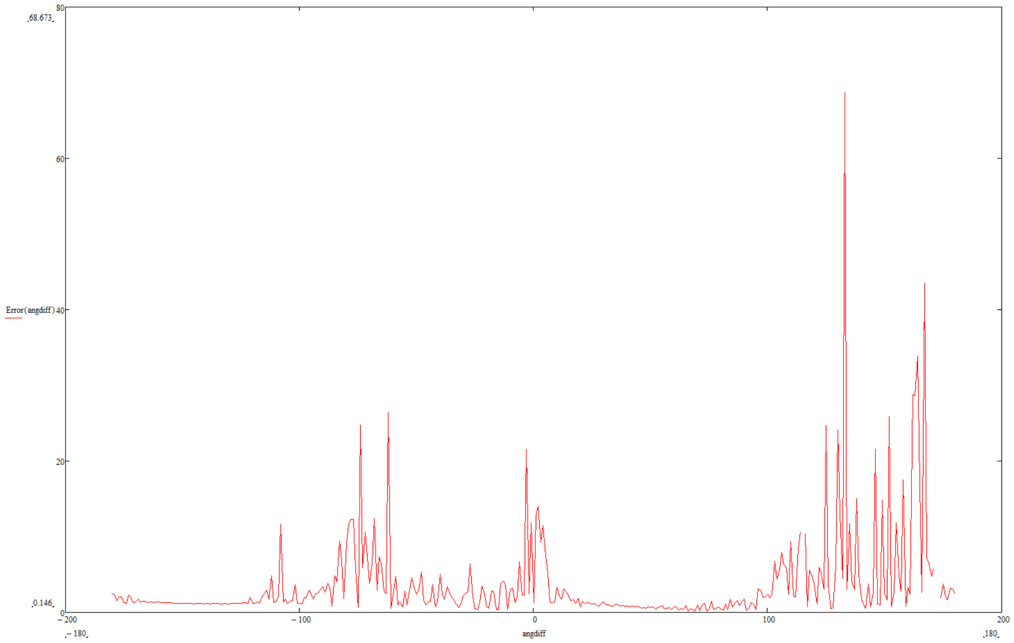


Figure 6.10: Angle Difference vs Error (1024x768)

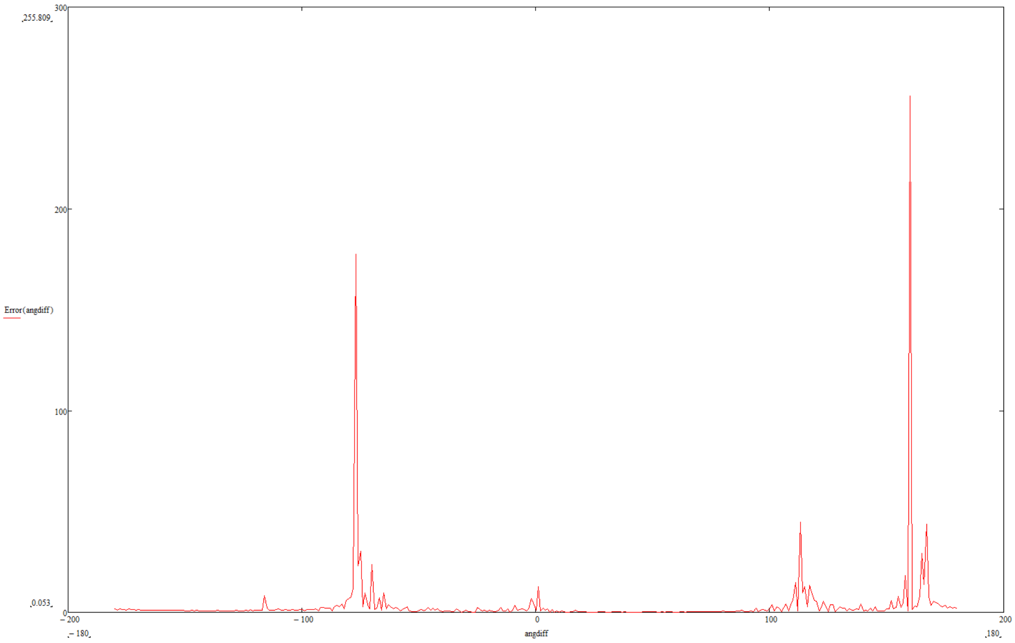


Figure 6.11: Angle Difference vs Error (2048x1536)

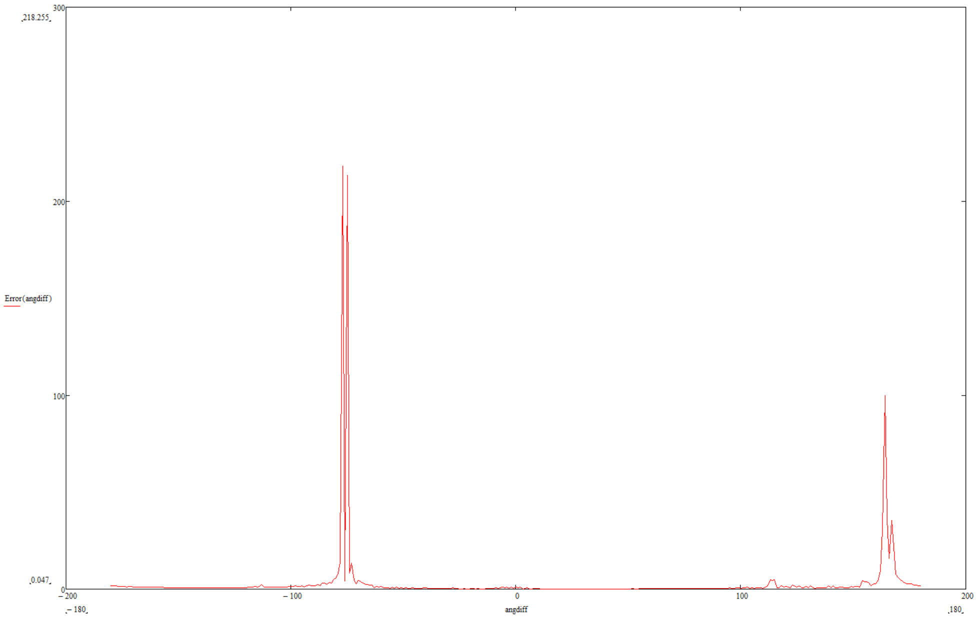


Figure 6.12: Angle Difference vs Error (6400x4800)

result was conducted. The investigation aimed to determine the levels of error associated with different fields of view (FoV), or if infact this was sufficiently

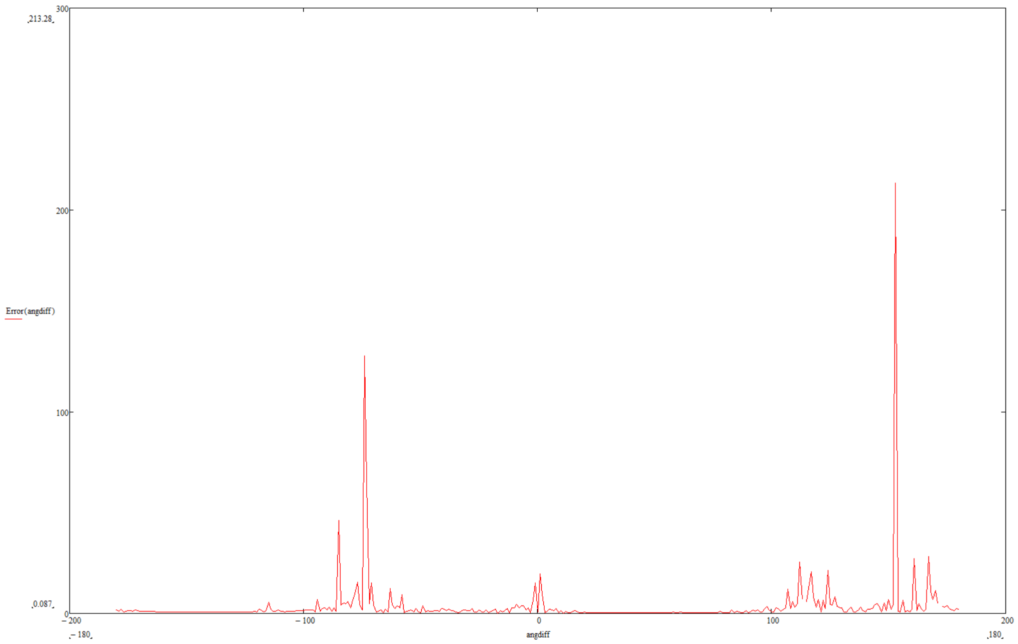


Figure 6.13: Angle Difference vs Error (1920x1080)

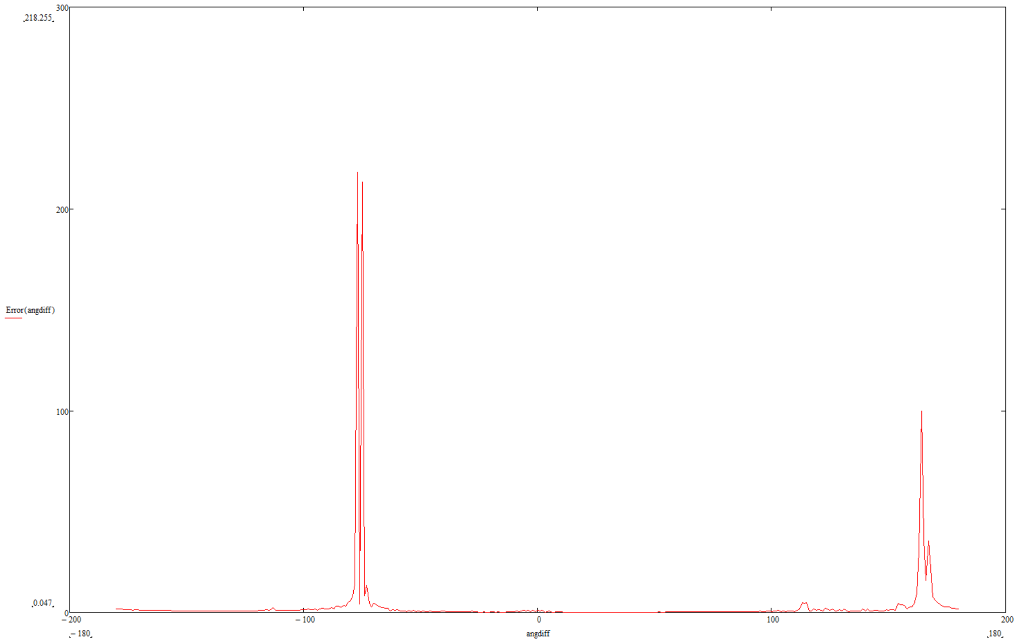


Figure 6.14: Angle Difference vs Error (7680x4800)

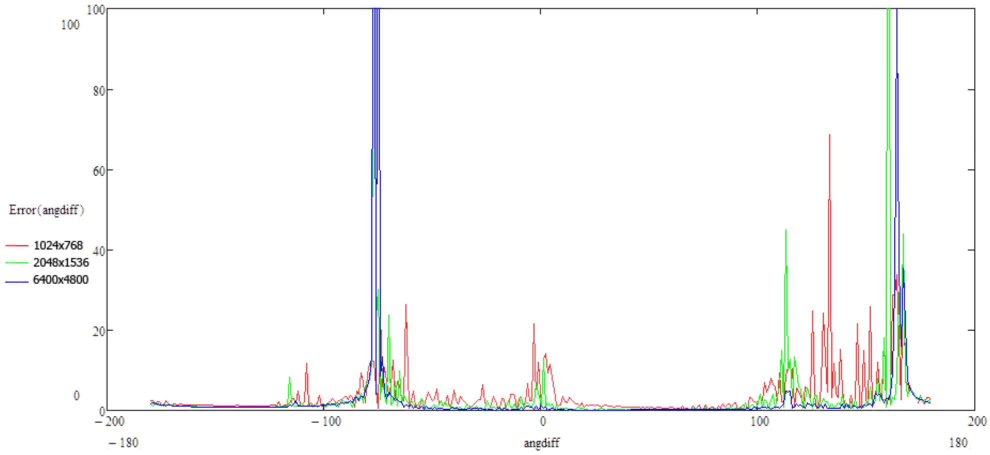


Figure 6.15: Angle Difference vs Error (Combined)

compensated for in the reverse projection calculations. A number of graphs were created using results generated by the mathematical model, which was configured with field of view as the only variable. Figure 6.16 shows FoV versus error for a relatively low resolution input set sampled at 640x480 pixels and figure 6.17 shows the same at the higher resolution of 7680x4800. Figures 6.18 and 6.19 show the surface plots as generated by varying the field of view of both cameras independently for the low resolution of 640x480 and the higher resolution of 7680x4800 respectively.

In extension of the above, the two concepts were combined in an experiment to determine how varying both the difference in camera angle and the field of view affected the results. The results of this experiment are shown in figure 6.20 which shows difference in angle and field of view versus error for the resolution of 640x480 and figure 6.21 which shows the same for the resolution 7680x4800. These results showed that at all resolutions field of view configurations exist that yield undesired correspondence lines or cause the occlusion or clipping of interest points. At certain fields of view the image is optically distorted as to cause parallel, short or same correspondent line conditions. In reality, the field of view is dependant on the focal length of the lens and optical properties of the physical camera. The specific application and goals of the augmented reality system will influence the type of lens used, however the above results should be considered when deciding which lens to use in order to avoid causing one of the above undesirable conditions.

Graphs 6.22, 6.23 and 6.24 were produced by varying the resolution of both cameras simultaneously and show how varying the resolution in such a manner effects error. These graphs are for the aspect ratios of 1.333, 1.777 and 1.6 respectively. It was observed that as with varying the resolution of both cameras

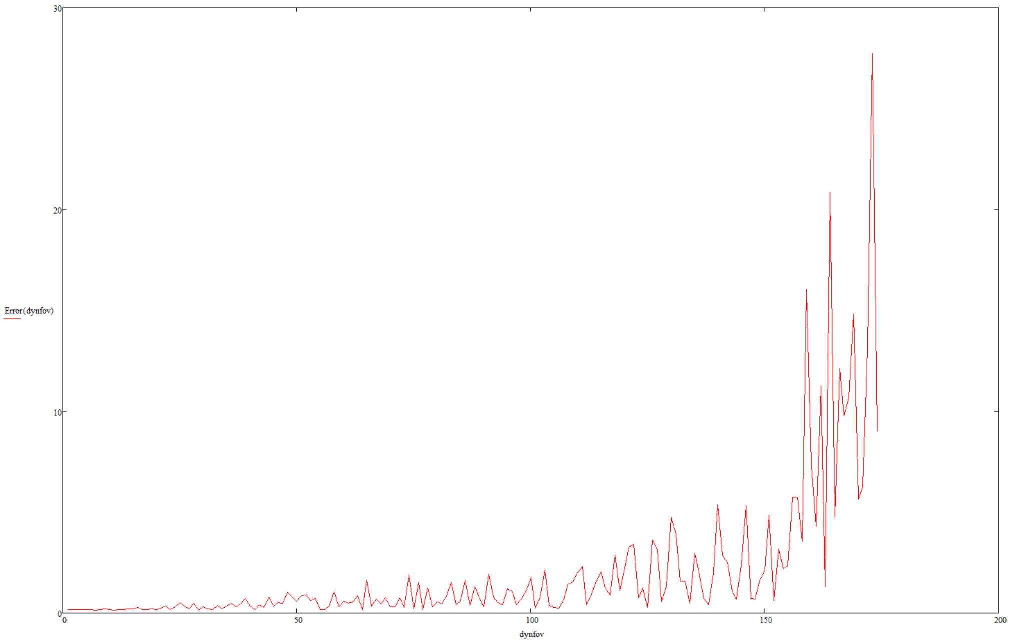


Figure 6.16: FoV vs Error (640x480)

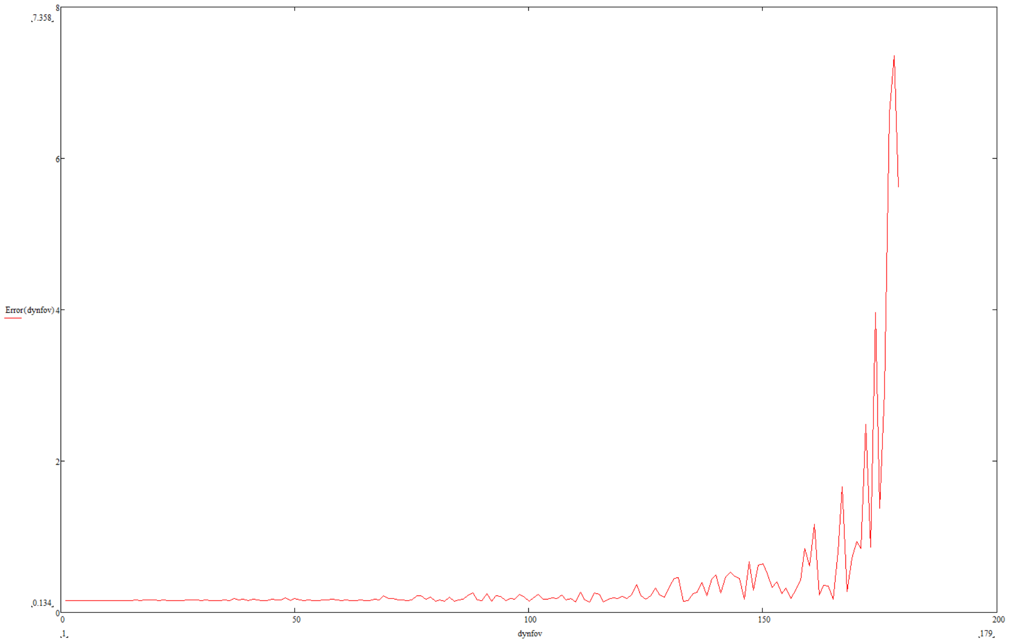


Figure 6.17: FoV vs Error (7680x4800)

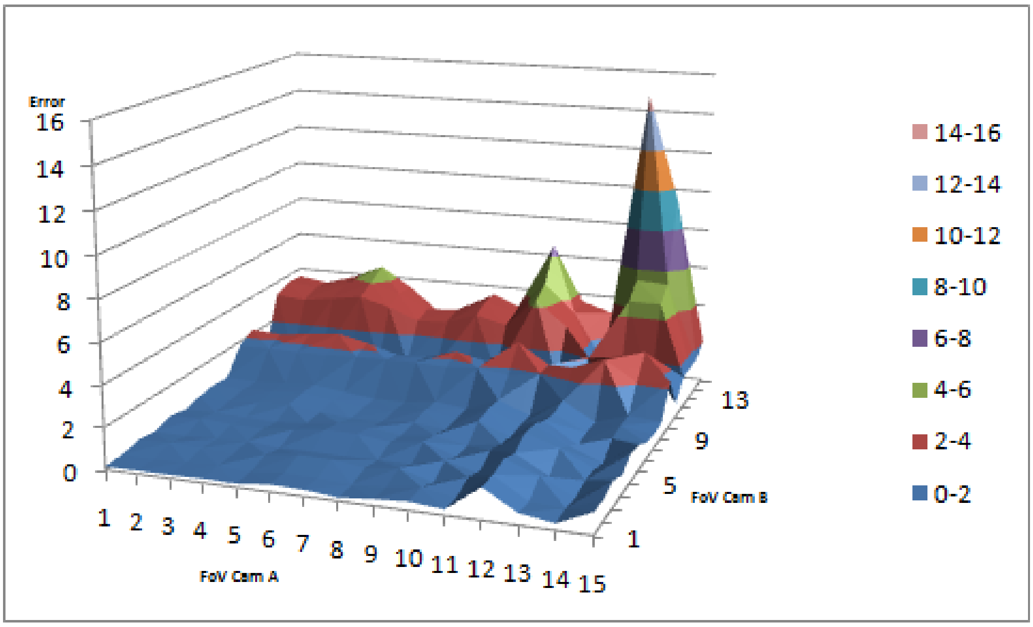


Figure 6.18: FoV CamA, FoV CamB vs Error (640x480)

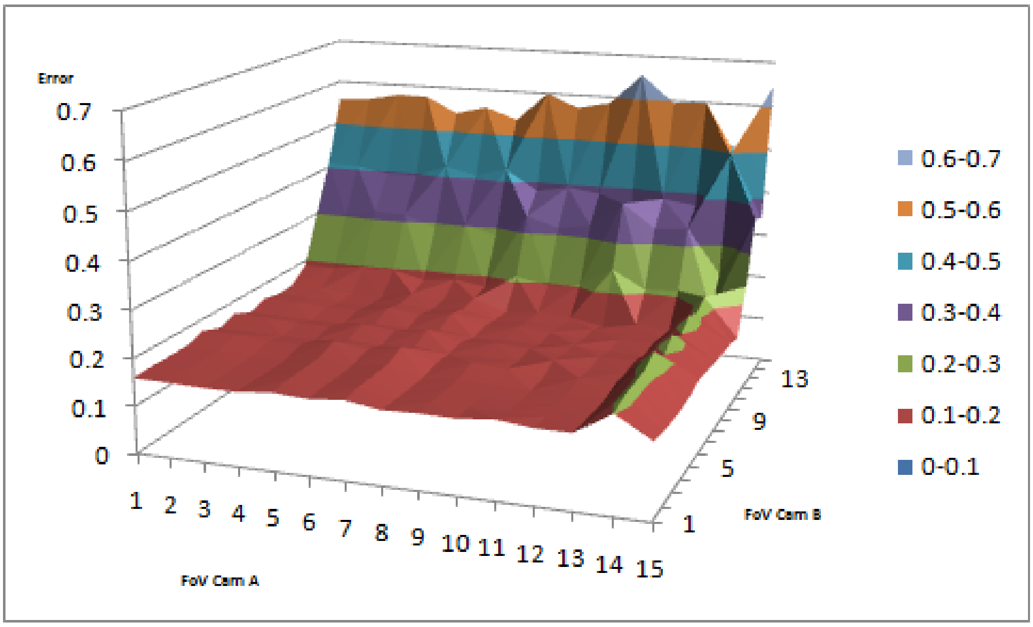


Figure 6.19: FoV CamA, FoV CamB vs Error (7680x4800)

simultaneously some geometry can fall in such a way that undesirable CLs are produced, the difference being that the majority of error is induced on the plane

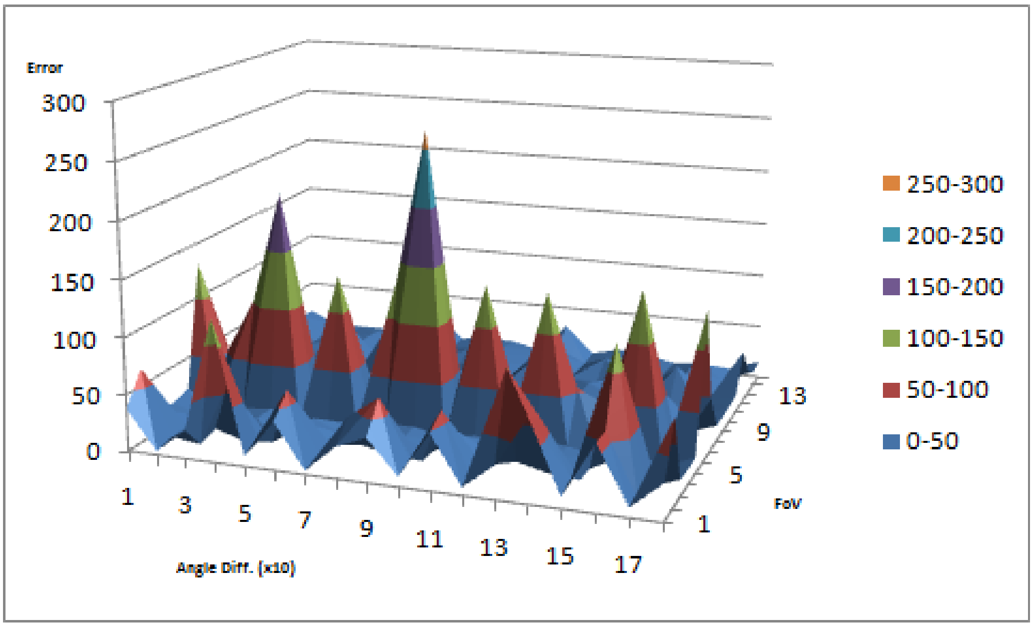


Figure 6.20: Angle Difference, FoV vs Error (640x480)

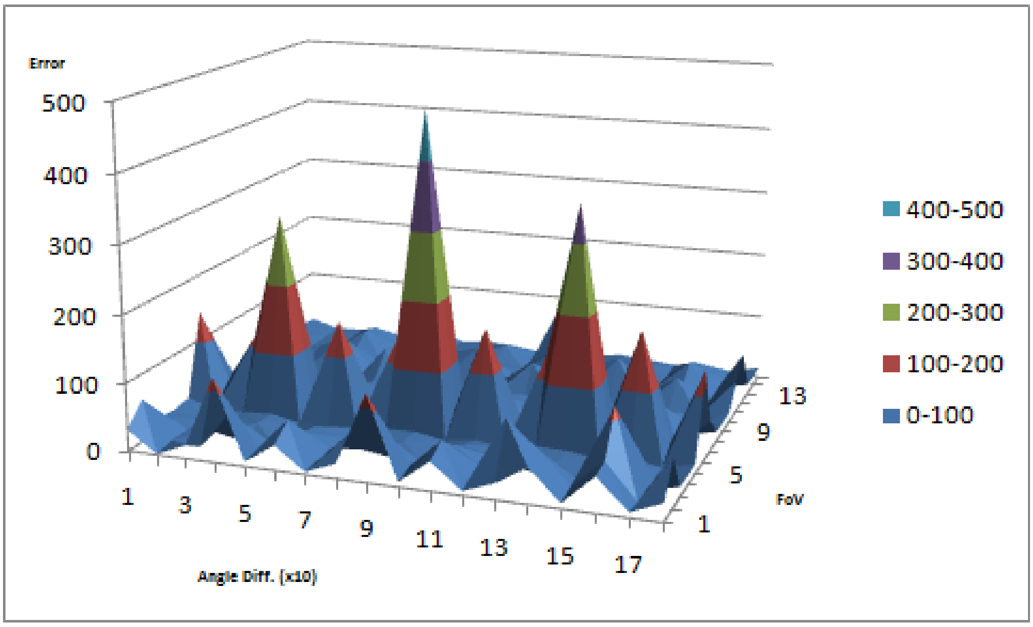


Figure 6.21: Angle Difference, FoV vs Error (7680x4800)

on which that camera projects, post geometric registration. It can be seen that when using consistent input scene geometry, a similar pattern of error exists between the aspect ratios. Once again, on average, higher resolution imagery produces results containing less error.

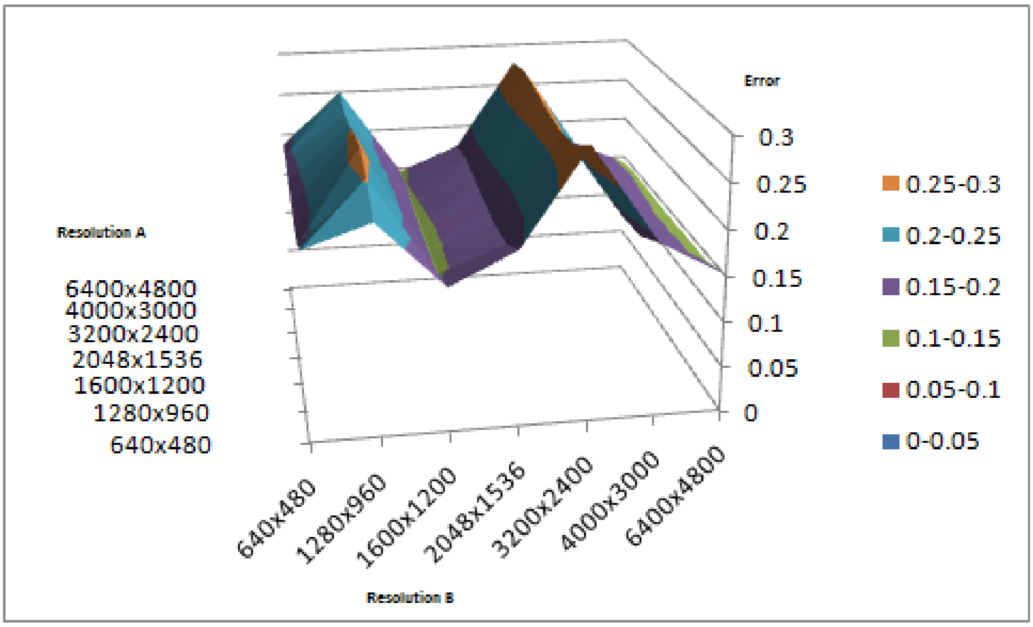


Figure 6.22: Variable Resolution Both Cameras (1.333 aspect)

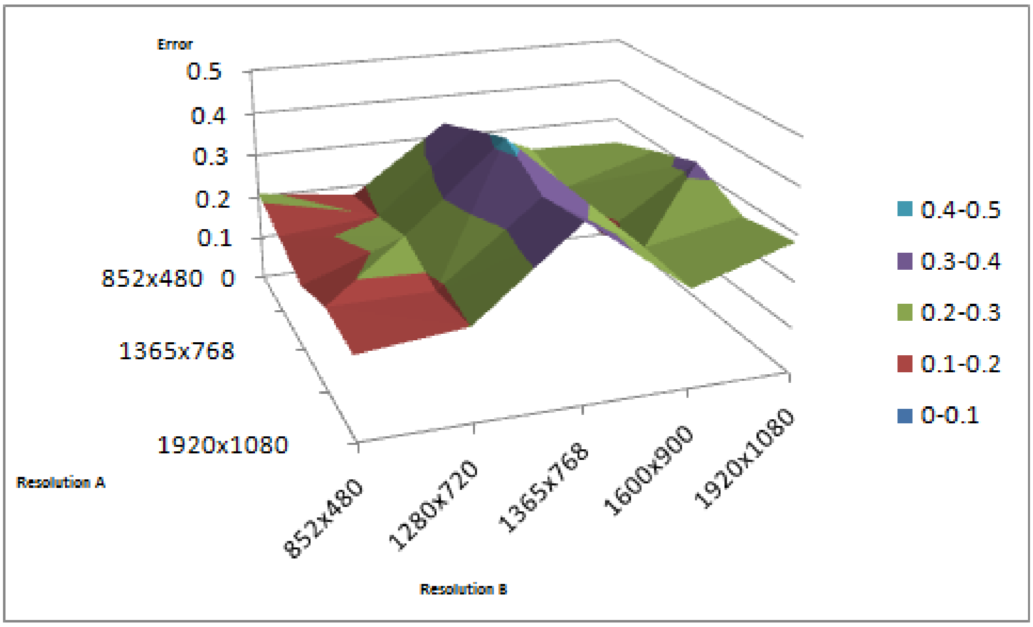


Figure 6.23: Variable Resolution Both Cameras (1.777 aspect)

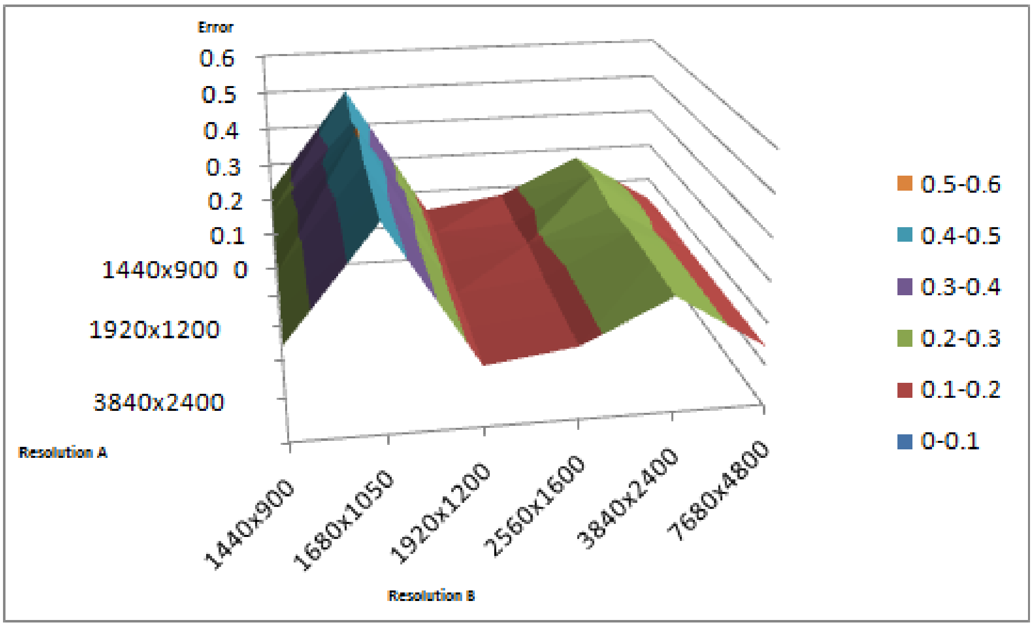


Figure 6.24: Variable Resolution Both Cameras (1.6 aspect)

6.3 Susceptibility to Pixel Error

The technique is unable to identify or compensate for pixel error, therefore it is important to be aware of the detection problems that such error can create. To this end, a number of experiments were conducted in order to establish the effect that any induced inaccuracy would cause. This error was induced on the pixel level, as this is the most significant cause of error as induced by the inaccurate detection of image interest points as shown in figure 6.25.

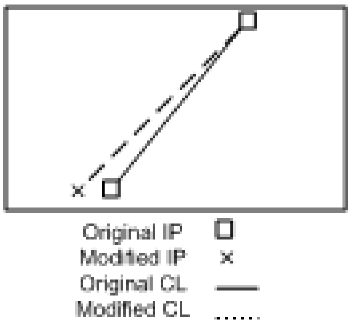


Figure 6.25: Artificially Inducing Error

The artificial pixel error was plotted against overall illuminant error in order to determine the existence of any correlation. Pixel error was artificially introduced

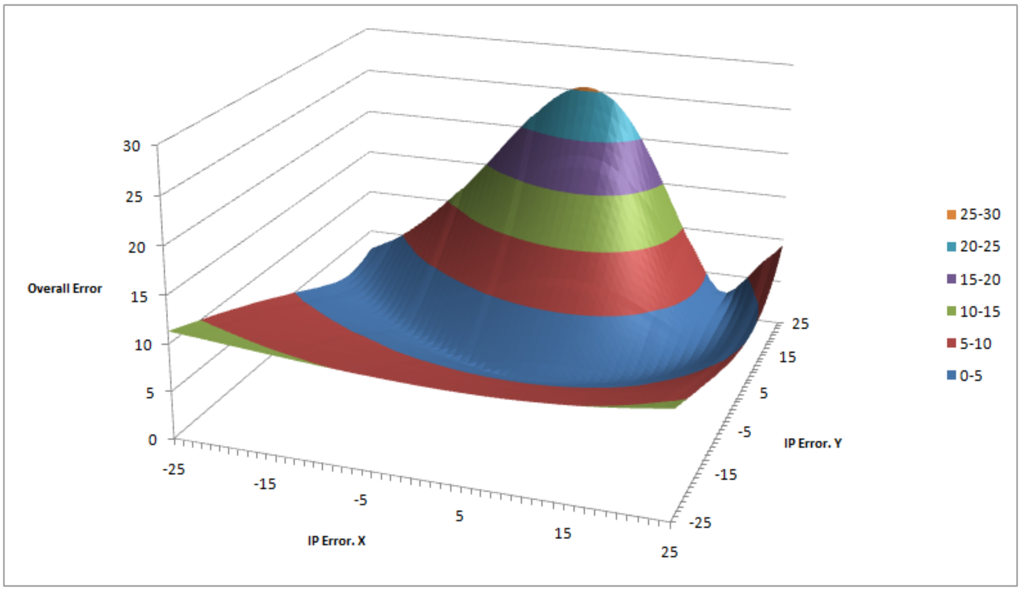


Figure 6.26: Induced IP Err. vs Overall Err.: Single Cam (640x480)

into both shadow and object interest point location vectors on both the x and the y axis for images of varying resolution. One and two camera inputs were considered separately in order to take into account the pixel inaccuracy caused by a single input, whilst none experienced through the second and also pixel error caused by both camera input devices simultaneously. Figure 6.26 contains a surface plot that shows the result of artificially inducing error within an input image with the resolution of 640x480. This figure shows error as induced into data obtained from just one camera input whereas figure 6.27 shows the result of inducing error into data obtained from both camera inputs.

The results show that pixel error in a detected IP has a proportional effect on the overall error. The majority of the impact being on the plane on which the camera projection lies. As expected, figure 6.26 shows that inducing IP error on the Y axis causes more dramatic changes in overall error than when inducing error on the X axis. This is due to IP error on the Y axis more greatly effecting the gradient of the associated CL lines than on the X axis. When both cameras induce pixel error the overall error is more universal and is less easily mitigated by the results from the other camera. Figure 6.27 shows that when error is induced into both camera inputs simultaneously error is greater and also both IP error axis more evenly effect the levels of error in the output, this is because error is now variable on both input planes.

Introducing additional camera inputs would potentially mitigate IP error by providing an additional line to intersect against during the final stages of the technique. This would be especially practical in augmented reality systems with

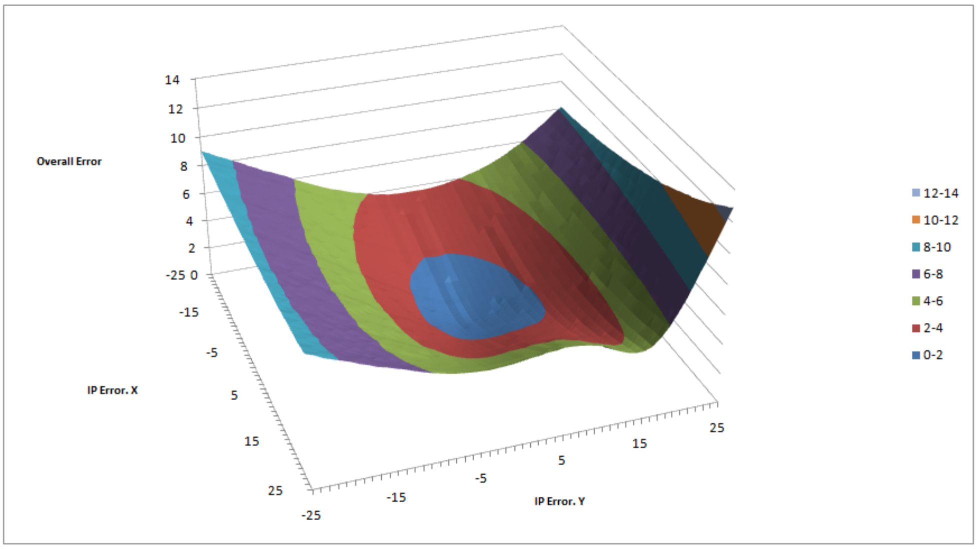


Figure 6.27: Induced IP Err. vs Overall Err.: Both Cam (640x480)

multiple participants. The same error induction experiment was repeated again in its entirety, except this time the position of the real illuminant was considered at variable distance from the origin. This resulted in error induced into a single camera input as is shown in figure 6.28 for error induced on the X axis and figure 6.31 on the Y axis. These figures show that a level effect occurs whereby the overall error resulting from introducing IP error is dramatically increased for illuminants that are of greater distance away as opposed to those illuminants that are closer.

The result of inducing error into both camera inputs is shown in figures 6.29 and 6.30 for the X axis and figures 6.33 and 6.34 for the Y axis. These results again show that the above mentioned lever effect occurs. The further away the illuminant, the greater the effect of incorrect IP detection has on the illuminant position. This experiment has graphically identified the relationship between illuminant distance from scene geometry and overall error for a given level of IP error. Additionally it can be seen in figure 6.29 that the distance at which the technique loses robustness to IP pixel error is less when inducing error into both cameras simultaneously. The result of using higher resolution imagery does mitigate this issue as can be seen in figure 6.30. This figure shows that the technique remains more robust for illuminants of a greater distance than when lower resolution imagery was used. The smoother gradient indicates a gradual, rather than sudden failure to accurately detect illuminants. When inducing IP error on the Y axis the gradient of the associated CL line is effected more than when inducing error on the Y axis as the relatively erratic results shown in figure 6.31 shows. This is an indication that the technique may cope less with error occurring on the

Y axis than on the X axis. This is also mitigated with higher resolution imagery as the smoother results in figure 6.32 show. It should also be noted that error is always significantly less, and the technique more robust to IP error with higher resolution imagery. When the illuminant is extremely close anomalies can sometimes be seen. The reason for this is that in this configuration the illuminant is actually positioned within the scene geometry, causing conditions that the technique is not designed, or even intended to handle. Therefore these anomalies, as seen within figures 6.32 and 6.34, can be safely ignored. The graphs 6.33 and 6.34 verify that the technique has improved robustness to the illuminant error lever effect with higher resolution input imagery. The peak where the illuminant distance is low, at the left of figure 6.34 is caused by the special condition as explained above.

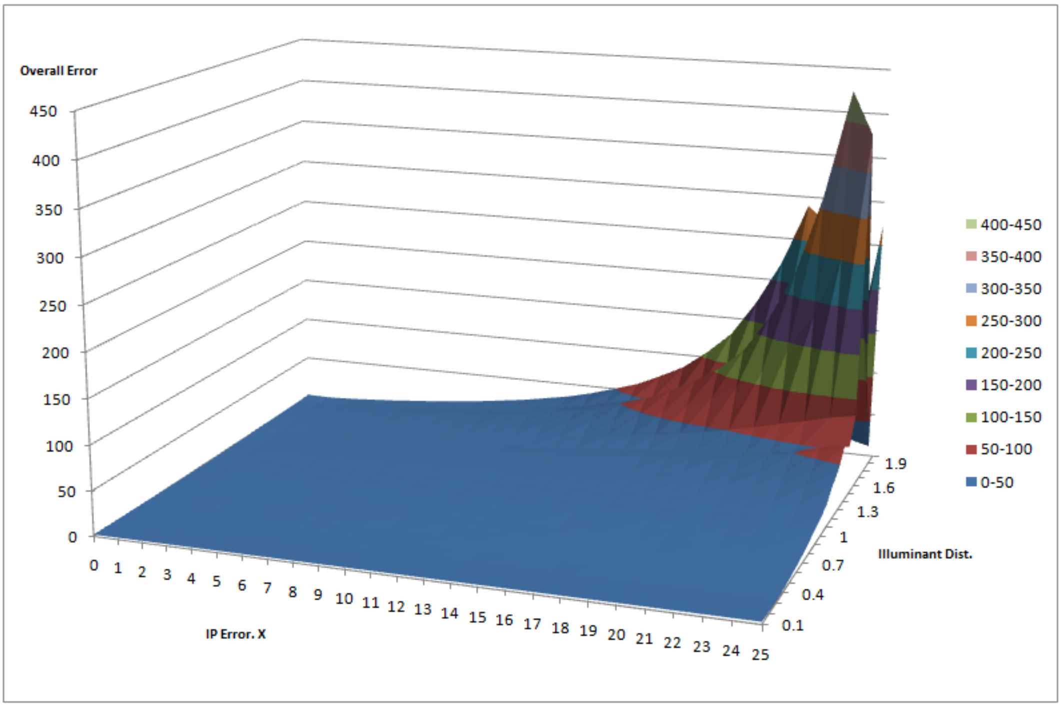


Figure 6.28: Illum. Dist., Ind. IP Err. X vs Overall Err.: Single Cam (640x480)

6.4 Technique Performance

The speed of the proposed technique was measured using the precision timer exposed by the Windows API and complexity is determined using the application profiler built into the Microsoft Visual Studio 2010 debugger. The technique, on average, executed from start to finish in 0.082 seconds. It should be noted that

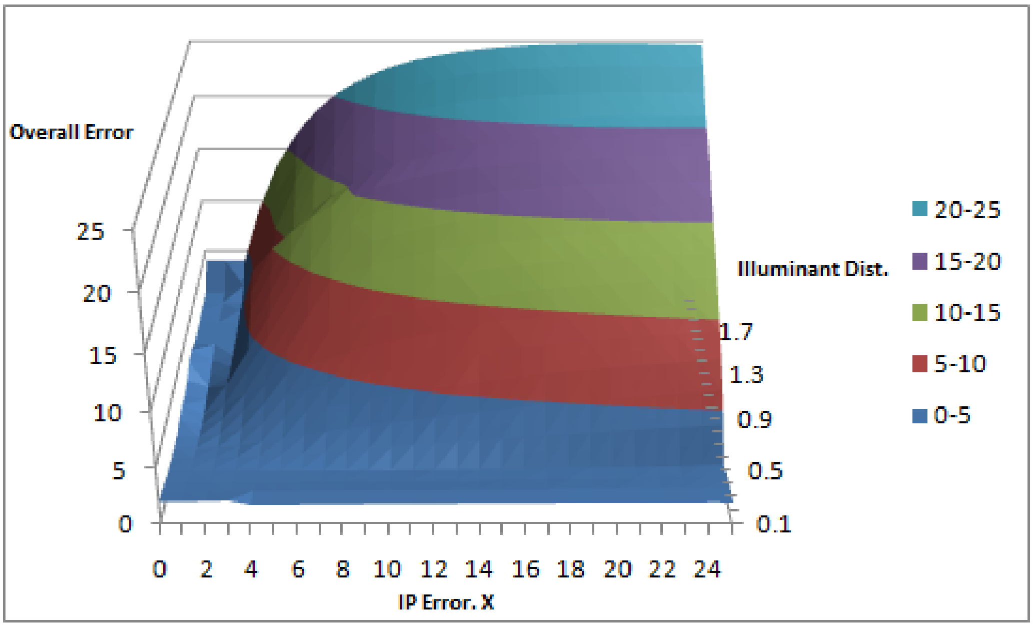


Figure 6.29: Illum. Dist., Ind. IP Err. X vs Overall Err.: Both Cam (640x480)

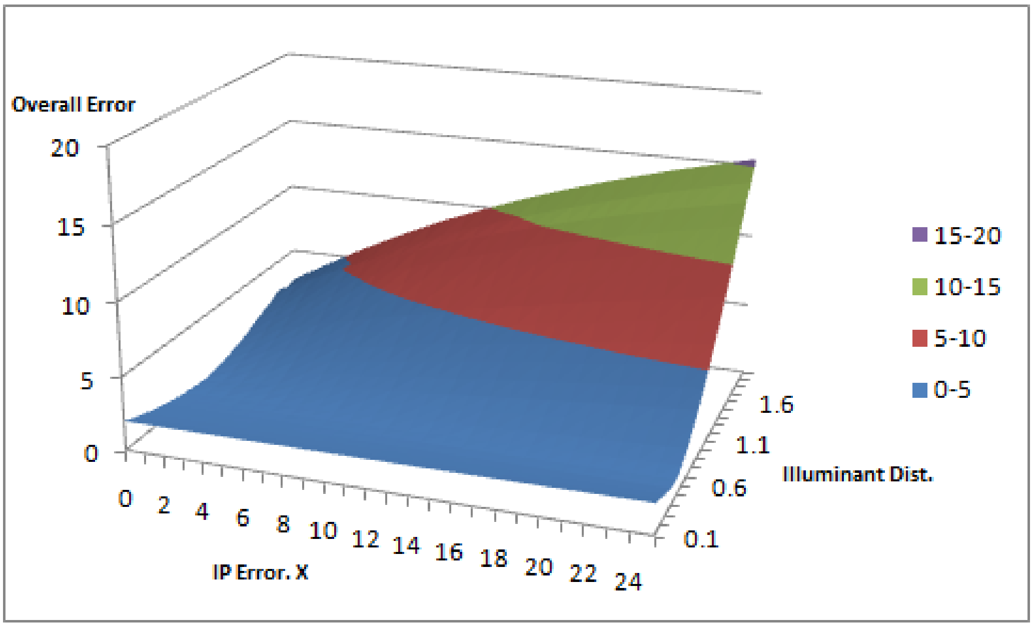


Figure 6.30: Illum. Dist., Ind. IP Err. X vs Overall Err.: Both Cam (7680x4800)

the Windows API is only able to show timings at intervals of around 18.2/sec, depending on which functions are available to the programmer through the Windows

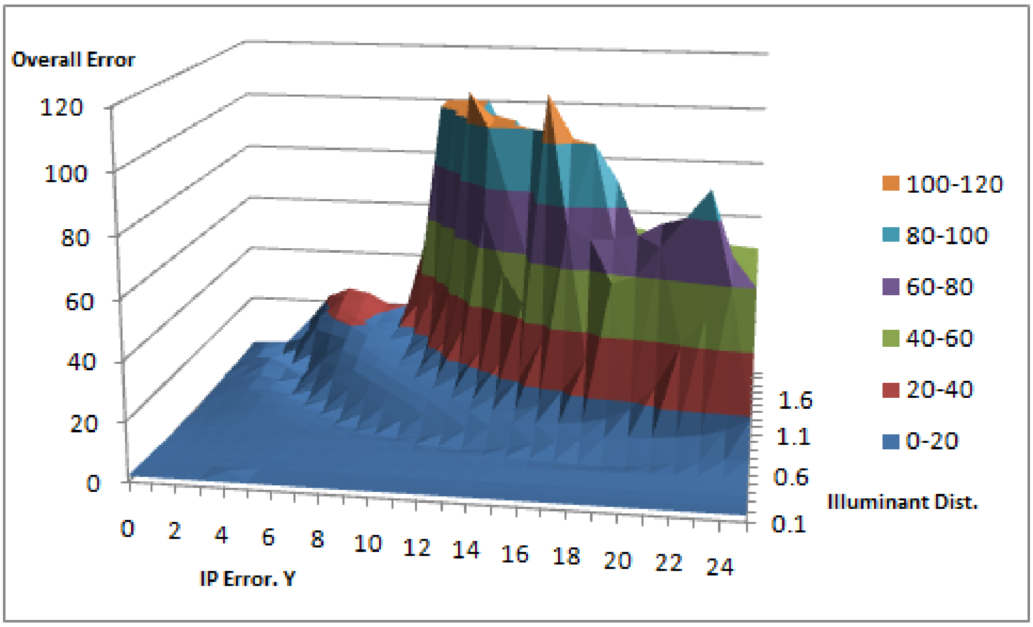


Figure 6.31: Illum. Dist., Ind. IP Err. Y vs Overall Err.: Single Cam (640x480)

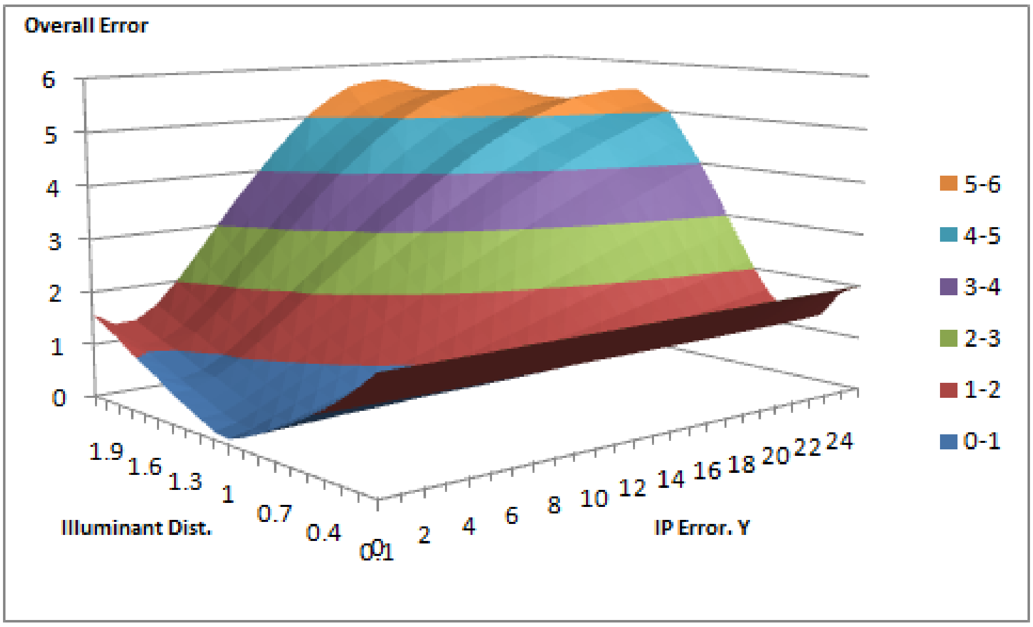


Figure 6.32: Illum. Dist., Ind. IP Err. Y vs Overall Err.: Single Cam (7680x4800)

SDK. As the graphical rendering thread was fully decoupled from the photometric registration update process it was possible to obtain the maximum capped

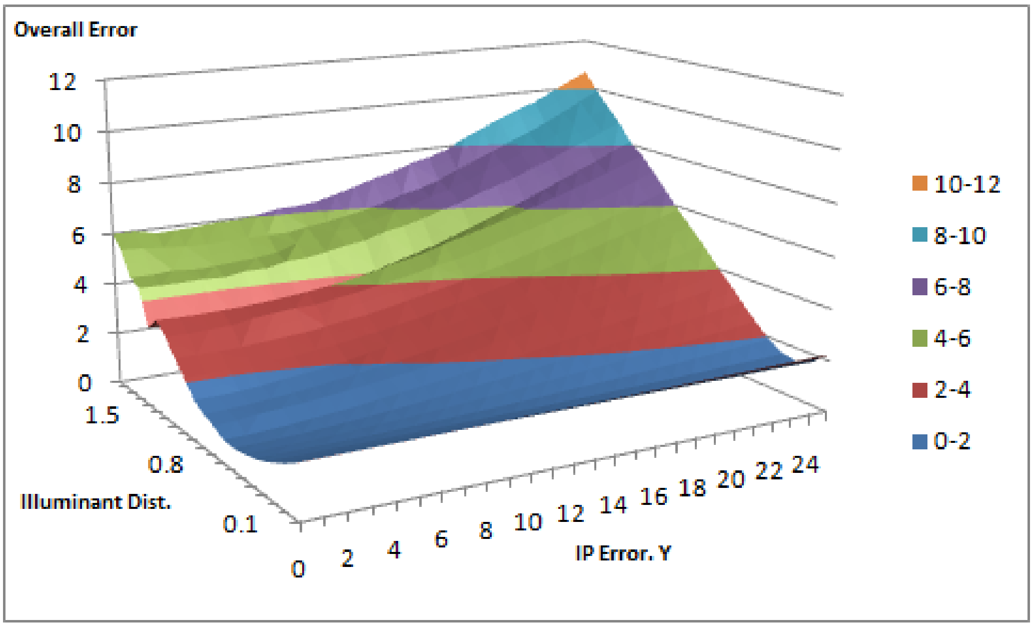


Figure 6.33: Illum. Dist., Ind. IP Err. Y vs Overall Err.: Both Cam (640x480)

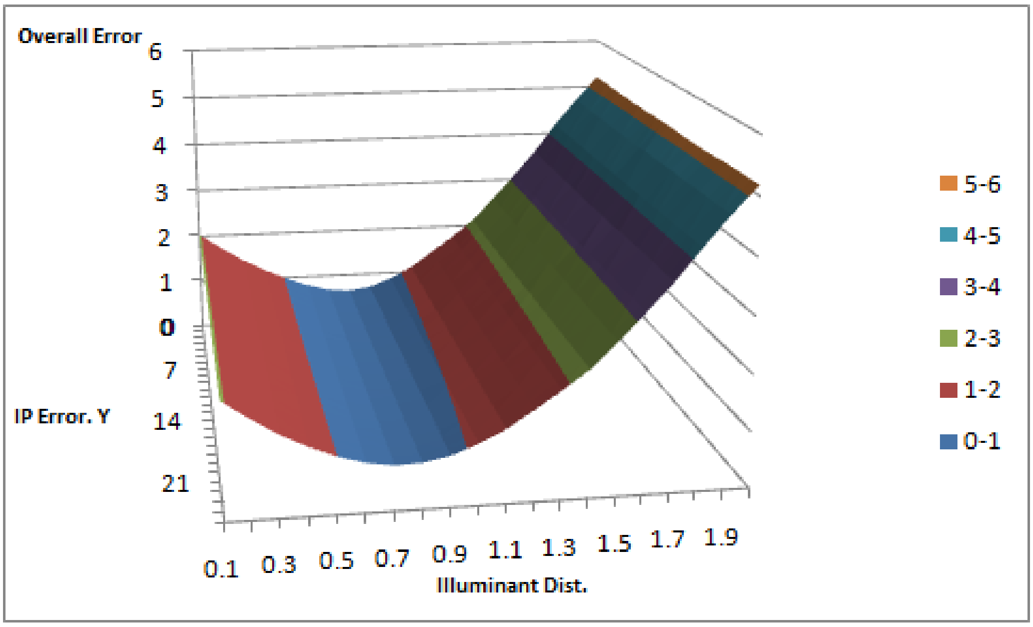


Figure 6.34: Illum. Dist., Ind. IP Err. Y vs Overall Err.: Both Cam (7680x4800)

render speeds of 60 frames per second under all operational conditions. It can therefore be said that the technique did not cause any graphical bottleneck issues.

Update speeds of 12.2Hz were obtained on a medium to high-end home computer with the following specification:

- 3.2GHz Quad-Core CPU
- 8GB DDR 3 RAM
- Windows 7 x64 (64bit)
- NVIDIA Geforce GTX 295
    - 1.75GB DDR3 Video RAM
    - Dual GPU with 1242MHz clock-speed
    - RAM Clock at 2GHz

Due to the threaded decoupling of graphical rendering and illuminant positional updated changes in resolution did not effect this update speed whatsoever. This was possible because the mathematical calculations within the illuminant update thread are not of high enough computational complexity to cause slowdown within any other thread on the available hardware. This may not be true of legacy systems that have very low computational resources, in which case lower framerates may be achieved. It is assumed that the complexity of underlying third party techniques is taken into consideration when deciding which is to be used. For example, if the augmented reality system requires that a convolution be performed on low-quality input imagery prior to registration then update rate will be inversely proportional to the resolution of that input. Additionally, as the proposed technique requires geometric registration, the overall update rate is also dependant on the complexity of the chosen geometric method. The complexity and operational speed of such third party techniques is an important consideration as was discussed in chapter 3.

The overall error does vary between camera configuration, resolution and even scene geometry as the correspondence lines are determined by these factors. A number of problem scenarios do arise when scene geometry is translated into CL lines with undesirable properties, these may be mitigated should additional information be available. Although failure conditions exist at certain regions with this scene layout, it should be noted that these will change depending on the geometry used. It was observed that no matter what configuration is used, the accuracy deteriorates rapidly when the camera angle difference approaches $0°$ and $180°$.

The most optimal angle for all non-erroneous scene configurations is observed to be approximately $90°$. The margin surrounding the extreme angles within which accuracy starts to fall becomes wider for lower resolutions. Therefore higher resolution imagery provides greater flexibility when physically positioning the cameras. Additionally, the rate at which error increases as these extreme angles are approached is greater for the lower resolution imagery as shown in the *Angle Difference vs Error* graphs above. In order to illustrate how scenes appear at with different angle parameters, figures 6.35, 6.36, 6.37, 6.38 and 6.39 show visualizations of the configuration and associated results for the angle differences of 0, 90, 110, 160, and 270 degrees respectively.

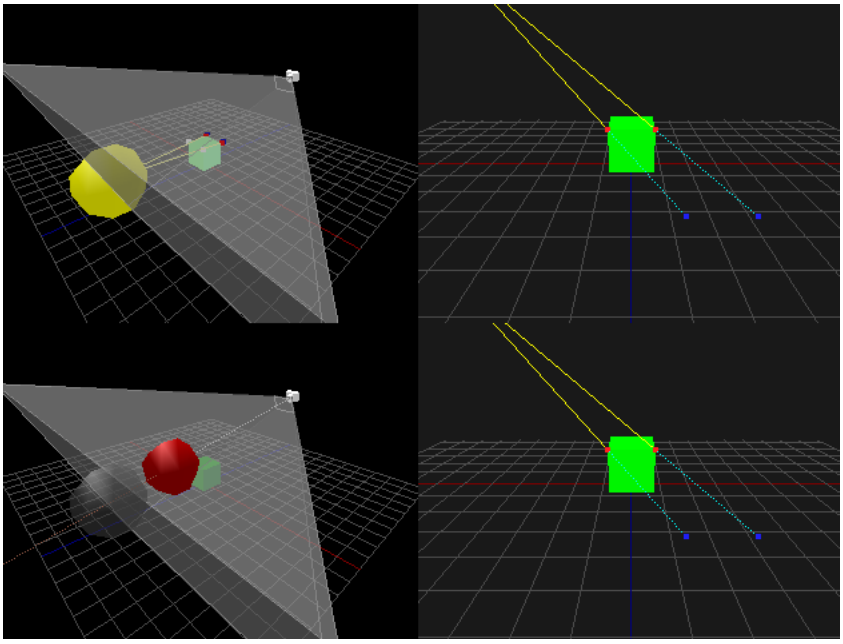


Figure 6.35: 0° Angle Difference Visualization

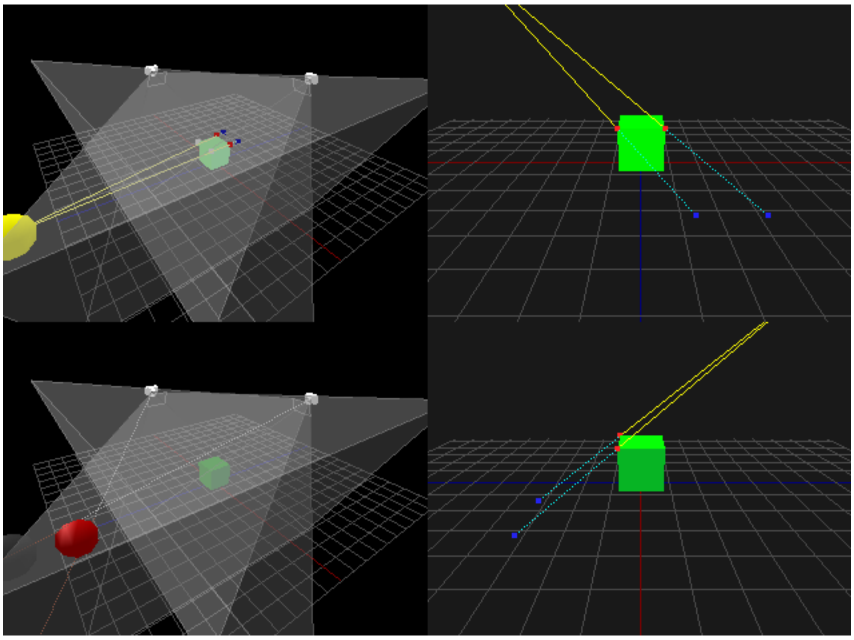


Figure 6.36: 90° Angle Difference Visualization

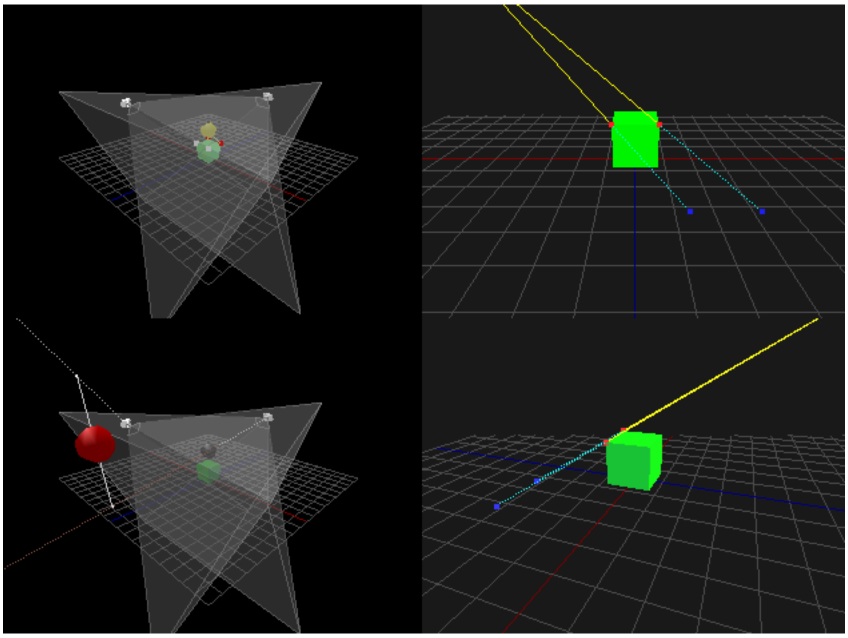


Figure 6.37: 110° Angle Difference Visualization

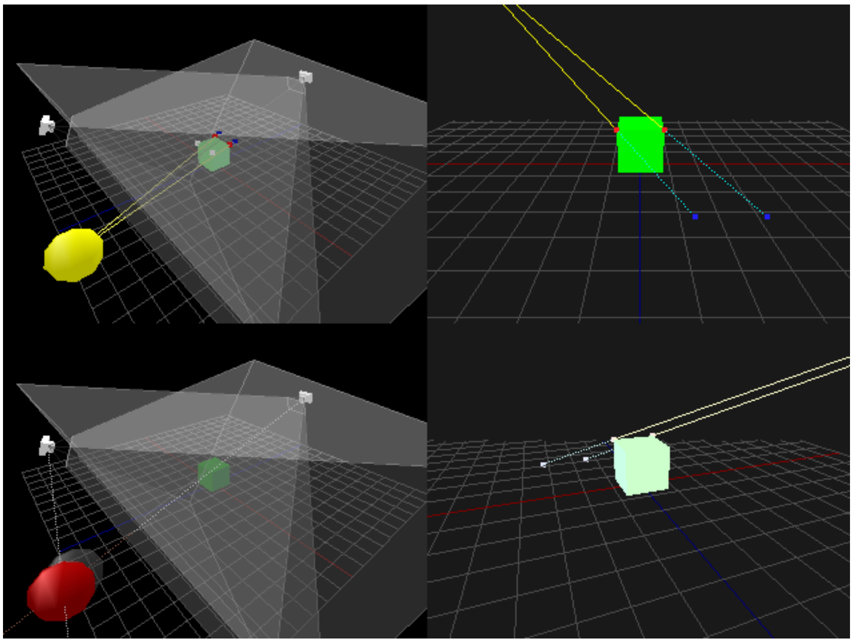


Figure 6.38: 160° Angle Difference Visualization

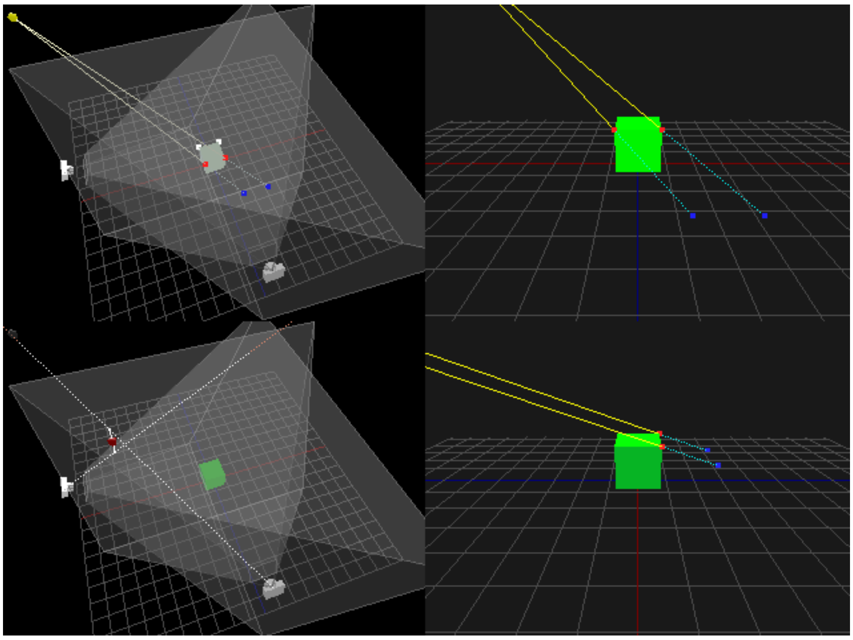


Figure 6.39: 270° Angle Difference Visualization

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Geometric and photometric registration are both important for any realistic augmented reality application. As geometric techniques become more reliable, research focus is turning to photometric registration. A number of photometric registration techniques have been proposed; however they are often computationally complex and although they may work well for static images, real-time methods are required for live video based systems. Existing real-time techniques require either pre-calibration or continuous calibration using known objects. Such techniques either introduce additional artificial components to the scene or place constraints on the system. Techniques that provide realistic real-time photometric registration have not yet been made available. The research discussed here has succeeded in furthering such realism when considering the presence of a single illuminant.

As the new technique is interest point based it is possible to achieve greatly reduced operational complexity compared to alternative approaches. Although some assumptions are made, the technique does not massively constrain the operational environment beyond the need for sufficiently visible cast shadows and scene geometry. The proposed technique is robust to occlusions, partially robust to noise within input imagery and does not disrupt scene realism by requiring the presence of artificial or pre-known objects within the environment. Features from shadow and object scene elements are expected to be extractable within a certain degree of accuracy.

The research progressed after first reviewing the state-of-the-art in photometric registration techniques. A new approach was then designed, proposed and validated. The approach includes steps to determine the geometrically registered 3D position of a single illuminant when given simultaneously captured images of the same scene as observed form at least two different angles; thus achieving photometric registration. No other research has been published that makes use of interest points to locate an absolute illuminant position, for the purpose of AR photometric registration or otherwise; additionally, the proposed technique displays characteristics that are superior to those discussed elsewhere that are either inflexible, slow, require pre-calibration or disrupt scene realism. High frame-rates are obtainable due to the speed at which the technique operates. By using this approach it is possible for augmented reality systems to produce more convincing augmentations; thus allowing for greater user immersion. This achievement is through the resultant capability to correctly and consistently match real and virtual illuminant conditions.

A number of proof-of-concept implementations have facilitated the verification, evaluation and refinement of the technique. Such implementations have delivered the insight required to iteratively further the development of the technique, improving robustness, accuracy and reducing operational complexity. Implementations take the form of a number of a mathematical models, a reusable photometric registration library and a number of simulation applications.

The photometric registration library delivers the capability of real-time illuminant consistency and is reusable between multiple applications. This library takes the form of a C++ API. It allows a host application to obtain a single 3D illuminant position from two or more input images and associated data. This API may be statically or dynamically integrated into any application that requires the functionality of the proposed technique. The applications that have resulted from this research include:

- Core API library
- C++ code to augment reality and simulate manual light conditions
- A software proof of concept application for technique and scenario visualization
- An application to simulate the end result after processing image data using the photometric registration API
- An application to generate input imagery dynamically and simulate variable environmental conditions
- An application to process pseudo interest points in order to achieve illuminant position from user specified data
- A MathCAD application to numerically simulate and graph technique results

Whilst the API library and mathematical model represent the core functionality and technique theory, the above applications demonstrate capability and application specific solutions. The software proof of concept provides a user-friendly visualization of technique operation through its various functional stages. It allows for the simulation of of variable scenarios and configurations through a user-friendly graphical user interface. This application is able to dynamically generate input imagery for a wide-range of such scenarios and configurations and provides a thorough experimentation capability. The imagery and interest point data sets are generated on the fly, then analyzed. Another software prototype application performs full reality augmentation using output illuminant data as derived by analyzing interest points associated with two input frames. This application can process sequential image input, simulating and tracking dynamic scenes. Thus allowing for the acquisition and inclusion of real or pseudo-real imagery, and the observation of the reactions of the technique to changes in input and environmental conditions and system configurations.

A detailed investigation into technique robustness, including a number of operating conditions has been undertaken and the results reported. The susceptibility to pixel-based interest point error has also been investigated for a number of different illuminant configurations. Mitigation methodology has been proposed for conditions in which the technique fails or operates sub-optimally. Such mitigation methods include user configuration decisions and implementation specific automatic adaptations that would dynamically and automatically reject bad input data, making best use of the information available at the time.

Conditions under which the technique both operates optimally and fails have been discussed. Such conditions in which the technique functions optimally include:

- High resolution input imagery
- Angles with high image disparity
- Input that allows many correspondence lines to be detected
- Scenes with clean shadows and well-defined geometry

Conditions under which the technique fails or performs sub-optimally include:

- Low resolution input
- 0° difference in input angle
- 180° difference in input angle
- Scenes with multiple light sources
- Complex concave or convex scene geometry

This thesis has outlined the problem, and context, of photometric registration within the field of augmented reality. It has presented a new method that addresses this problem in ways that are superior to preceding solutions. The conditions under which the technique operates both optimally and sub-optimally are discussed. Failure conditions are also considered and mitigation strategies are proposed where appropriate. Such strategies can be used simultaneously or independently and include:

- Use of temporal meta-data
- Averaging input between multiple frames
- Tracking velocity of moving illuminant
- Omission of backward CL intersection
- Automatic avoidance of bad CLs
- Image noise reduction
- Detection of parallel and identical CLs
- Drop obvious anomalous data and last-known good as input
- Requirement of suitable camera angles; or otherwise ensure high resolution imagery
- Use of average 2D intersection point; factoring as many CLs as available
- Increasing the number of camera inputs to achieve better 3D accuracy

7.2 Contributions

In addition to this thesis, a number of research papers have been published that document the proposed technique, resultant mathematical models, proof-of-concept applications and the above mentioned investigations. These publications are as enumerated on page page iii. The evaluation and verification process has been outlined and the results documented and compared to those of other published approaches. The technique itself is proven numerically as per the mathematical model discussed in section 5.1, visually via the prototype AR simulation as seen in figure 5.4 and graphically as presented in chapter 6. The resulting technology will benefit society as it has direct application potential within the fields of realistic training, virtual simulation, entertainment and gaming. The novel aspects of the technique include the use of cast-shadow and geometry interest points to detect the location of a single illuminant.

7.3 Future work

Further work will aim to provide technique robustness under more environmental scenarios, including complex global illuminant conditions. Complex scenes will be analyzed and multiple light sources considered, as will different types of light source and shadow. Research into how the technique may be massively parallelized by use of GPGPU techniques may be undertaken. GPU assisted execution would potentially allow for the introduction and processing of many camera inputs simultaneously, improving both technique robustness and accuracy. Additionally, an investigation may be undertaken into whether a single camera input may provide sufficient information for the technique to succeed, provided that it is moving and the change in camera pose be detected through optical flow methods or through the geometric registration of two subsequent frames. It may be the case that the two or more camera requirement may be eliminated in such scenarios. The automatic identification and selection of pairs of interest points to use and how they correspond is also an area for investigation. In addition to the direct contribution to knowledge, this research project has opened up additional areas of exploration. These areas are likely to facilitate the enhancement of realistic augmented reality capability, and future interest point based methods may push the boundaries of photometric registration technology for a number of years to come.

Bibliography

[1] K. Agusanto, L. Li, Z. Chuangui, and N. Sing. Photorealistic rendering for augmented reality using environment illumination. *Mixed and Augmented Reality, IEEE / ACM International Symposium on*, 0:208, 2003. [cited at p. 46]

[2] S. Ahn, M. Choi, J. Choi, and W. Chung. Data association using visual object recognition for ekf-slam in home environment. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2588–2594, October 2006. [cited at p. 40]

[3] S. Ando. Consistent gradient operators. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(3):252 –265, mar 2000. [cited at p. 52]

[4] ARTag. Augmented reality api. `http://www.artag.net`, April 2010. [cited at p. vii, 35]

[5] M. Bailey and Dru Clark. Using chroma depth to obtain inexpensive single-image stereovision for scientific visualization. In *Journal of Graphics Tools*, pages 1–9, 1999. [cited at p. 22]

[6] P. Bao and D. Xu. Complex wavelet-based image mosaics using edge-preserving visual perception modelling. *Computers and Graphics*, pages 309–321, 1999. [cited at p. 53]

[7] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 218–233, 2003. [cited at p. 154]

[8] A. Basso, H. Graf, D. Gibbon, E. Cosatto, and S. Liu. Virtual light: digitally-generated lighting for video conferencing applications. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, pages 1085 –1088 vol.2, 7-10 2001. [cited at p. 45, 49]

[9] R. Behringer. Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. In *Proceedings of IEEE Virtual Reality*, page 244, 1999. [cited at p. 26, 38, 40]

[10] W.E. Bradley and F. van de Kop. A comparison of a mechanically stabilized gyrocompass and a gps-aided inertial navigation system. In *OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Century*, volume 2, pages 780–784, 1999. [cited at p. 27]

[11] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. [cited at p. 51]

[12] V. Coors, T. Huch, and U. Kretschmer. Matching buildings: pose estimation in an urban environment. *isar*, 00:89, 2000. [cited at p. 40]

[13] K. Cornelis, M. Pollefeys, M. Vergauwen, and L. Gool. Augmented reality using uncalibrated video sequences. *Lecture Notes in Computer Science*, 2018:144–160, 2001. [cited at p. 39]

[14] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Proceedings SIGGRAPH Computer Graphics*, pages 135–142, 1993. [cited at p. 23]

[15] A. Davison. Real-time simultaneous localisation and mapping with a single camera, 2003. [cited at p. 40]

[16] A. Dempster, N. Laird, and B. Rubin. Maximum likelihood from incomplete data via the em algorithm. In *Journal od the Royal Statistical Society*, pages 1–38, 1977. [cited at p. 54]

[17] D. Devarajan, R. Radke, and H. Chung. Distributed metric calibration of ad hoc camera networks. *Transactions on Sensor Networks*, 2(3):380–403, August 2006. [cited at p. 39]

[18] E. Downing, L. Hesselink, J. Ralston, and R. Macfarlane. Three-color, solid-state three-dimensional display. *Science 273*, pages 1185–1189, 1996. [cited at p. 24]

[19] D. Eberly. *3D Game Engine Design.* Morgan Kaufmann, 2001. [cited at p. 77, 158]

[20] Sparkfun Electronics. Devices. `http://www.sparkfun.com`, Feburary 2010. [cited at p. vii, 33]

[21] O. Faugeras. Three-dimensional computer vision: A geometric viewpoint. *Cambridge, MIT Press*, pages 1–43, 1993. [cited at p. 151]

[22] Y. Feng. Estimation of light source environment for illumination consistency of augmented reality. In *Proceedings of the 2008 IEEE Congress on Image and Signal Processing*, pages 771–775, 2008. [cited at p. viii, 47, 48, 160]

[23] M. Fiala. Artag, a fiducial marker system using digital techniques. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:590–596, 2005. [cited at p. 31]

[24] R. Freeman and A. Steed. Interactive modelling and tracking for mixed and augmented reality. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 61–64, New York, NY, USA, 2006. ACM. [cited at p. 31]

[25] H. Fuchs, S.Pizer, C. Tsai, and S. Bloombreg. Adding a true 3-d to a raster graphics system. In *IEEE Computer Graphics and Applications*, pages 73–78, 1982. [cited at p. 24]

[26] Duke Gledhill. *3D Panoramic Imaging for Virtual Environment Construction.* PhD in Informatics, Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdon, 2009. [cited at p. 52, 53]

[27] Duke Gledhill, Gui Yun Tian, D. Taylor, and David Clarke. Panoramic imaging - a review. *Computers and Graphics*, 27(3):435–445, June 2003. ©2003 Elsevier Science Ltd. [cited at p. 50]

[28] I. Gordon and D. Lowe. What and where: 3d object recognition with accurate pose. In *Toward Category-Level Object Recognition*, volume 41, pages 67–82, 2004. [cited at p. 39]

[29] I. Gordon and D. Lowe. What and where: 3d object recognition with accurate pose. In *Toward Category-Level Object Recognition*, volume 41, pages 67–82, 2006. [cited at p. vii, 39, 41]

[30] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions*, pages 623–629, 1971. [cited at p. 154]

[31] Erik Granum, Thomas B. Moeslund, Mortiz Strring, Wolfgang Broll, and Michael Wittkmper. Faciliating the presence of users and 3d models by the augmented round table. In *Proceedings of the 6th Annual International Workshop on Presence*, page 168178, 2003. [cited at p. vii, 4]

[32] M. Haller, S. Drab, and W. Hartmann. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 56 – 65, 2003. [cited at p. 26, 79, 157]

[33] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988. [cited at p. 40, 52]

[34] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. [cited at p. 152]

[35] Simon Haykin. Adaptive filters, 2001. [cited at p. 45]

[36] B. Heisele. Visual object recognition with supervised learning. *IEEE Computer Society Intelligent Systems*, pages 38–42, 2003. [cited at p. 54]

[37] M. Hirose, T. Ogi, S. Ishiwata, and T. Yamada. A development of immersive multiscreen displays (cabin). In *Proceedings of the VRSJ*, pages 137–140, 1997. [cited at p. 23]

[38] R. Hirose and H. Saito. A vision-based ar registration method utilizing edges and vertices of 3d model. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pages 187–194, December 2005. [cited at p. vii, 40, 41, 42]

[39] W. Hoff, T. Lyon, and K. Nguyen. Computer vision-based registration techniques for augmented reality, 1996. [cited at p. 39]

[40] Jeroen D. Hol, Thomas B. Schony, and Fredrik Gustafsson. Relative pose calibration of a spherical camera and an imu. *Mixed and Augmented Reality, IEEE / ACM International Symposium on*, 0:21–24, 2008. [cited at p. 27]

[41] H. Hua, C. Gao, L. Brown, N. Ahuja, and J. Rolland. Using a head-mounted projective display in interactive augmented environments. In *In Proceedings of IEEE and ACM International Symposium on Augmented Reality*, pages 217–223, 2001. [cited at p. 21]

[42] M. Inami, N. Kawakami, D. Sekiguchi, Y. Yanagida, T. Maeda, and S. Tachi. Visio-haptic display using head-mounted projector. In *In Proceedings of IEEE Virtual Reality*, pages 233–240, 2000. [cited at p. 21]

[43] Sony Computer Entertainment Inc. The eye of judgement. `http://www.eyeofjudgment.com`, January 2010. [cited at p. vii, 12]

[44] A. Joshi, S. Atev, O. Masoud, and N. Papanikolopoulos. Moving shadow detection with low and mid-level reasoning. In *IEEE International Conference on Robotict and Automation*, pages 4827–4832, 2007. [cited at p. 54]

[45] S Julier, Y Baillot, M Lanzagorta, D Brown, and L Rosenblum. Bars: Battlefield augmented reality system. In *In NATO Symposium on Information Processing Techniques for Military Systems*, pages 9–11, 2000. [cited at p. vii, 4, 5]

[46] M. Kanbara and N. Yokoya. Real-time estimation of light source environment for photorealistic augmented reality. In *In Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, United Kingdom*, pages 911–914, 2004. [cited at p. 45, 49]

[47] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmentedreality conferencing system. In *Proceedings of the 2nd International Workshop on Augmented Reality*, pages 85–94, 1999. [cited at p. 39, 45]

[48] R. Kijama and T. Ojika. Transition between virtual environment and workstation environment with projective head-mounted display. In *In Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 130–137, 1997. [cited at p. 22]

[49] S. Kim, S. Diverdi, J. Chang, T. Kang, Ronald Iltis, and Tobias Hollerer. Implicit 3d modeling and tracking for anywhere augmentation. In *Proceedings of the 2007 Symposium on Virtual Reality Software and technology*, 2007. [cited at p. 40, 41]

[50] G. Klinker, S. Shafer, and T. Kanade. A physical approach to color image understanding. *Int. J. Comput. Vision*, 4(1):7–38, 1990. [cited at p. 43, 44]

[51] K. Knowlton. Computer displays optically superimposed on input devices. In *Bell Systems Technical Journal*, pages 36–383, 1977. [cited at p. 22]

[52] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *ACM VRST '97 Lausanne Switzerland*, pages 87–94, 1997. [cited at p. 39]

[53] J. Kollin. A retinal display for virtual-environment applications. In *In SID International Symposium Digest of Technical Papers*, pages 827–834, 1993. [cited at p. 21]

[54] A. Kosir and J. Tasic. Simple shape detection using pattern spectrum. *Ljubljana Electrical Engineering*, pages 58–65, 2001. [cited at p. 55]

[55] W. Krueger, C. Bohn, B. Frohlich, H. Schuth, W. Strauss, and G. Wesche. The responsive workbench: A virtual work environment. In *IEEE Computer Graphics and Applications*, pages 42–48, 1995. [cited at p. 23]

[56] W. Krueger and B. Frohlich. The responsive workbench. In *IEEE Computer Graphics and Applications*, pages 12–15, 1994. [cited at p. 23]

[57] J. Lee, S. You, and U. Neumann. Tracking with omni-directional vision for outdoor ar systems. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 47. Integrated Media Systems Center, 2002. [cited at p. 39]

[58] A. Leone and C. Distante. Shadow detection for moving objects based on texture analysis. In *Pattern Recognition*, pages 1222–1233, 2007. [cited at p. 54]

[59] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua. Fully automated and stable registration for augmented reality applications. In *Proceedings of the 2nd International Symposium on Mixed and Augmented Reality*, page 93, 2003. [cited at p. 41]

[60] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. *IEEE ICIP*, pages 24–35, 2002. [cited at p. 55]

[61] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. [cited at p. viii, 40, 52, 80]

[62] M. Lucente. Interactive three-dimensional holographic displays: Seeing the future in depth. In *Proceedings SIGGRAPH Computer Graphics*, pages 63–67, 1997. [cited at p. 24]

[63] J. Ma, Y. Zhou, Q. Hao, and Y. Zhang. Efficient estimation of multiple illuminant directions using c-means clustering and self-correction for augmented reality. In *ICIS '09: Proceedings of the 2009 Eigth IEEE/ACIS International Conference on Computer and Information Science*, pages 1106–1110, Washington, DC, USA, 2009. IEEE Computer Society. [cited at p. vii, 8, 48]

[64] E. MacDougall. Spatial filtering. In *Proceedings. International Geographical Union. Commission on Quantitative Methods*, pages 425–434, 1970. [cited at p. 50]

[65] N. Martel-Brisson and A. Zaccarin. Learning and removing cast shadows through a multidistribution approach. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1133–1146, 2007. [cited at p. 54]

[66] Mashable. Top 10 iphone apps. `http://www.mashable.com/2009/12/05/augmented-reality-iphone`, December 2009. [cited at p. vii, 13, 14, 15]

[67] S. McKay, S. Mason, L. Mair, P. Waddell, and M. Fraser. Membrane mirror-based display for viewing 2d and 3d images. In *Proceedings of SPIE 3634*, pages 144–155, 1999. [cited at p. 24]

[68] S. McKay, S. Mason, L. Mair, P. Waddell, and M. Fraser. Stereoscopic display using a 1.2-m diameter stretchable membrane mirror. In *Proceedings of SPIE 3639*, pages 122–131, 1999. [cited at p. 24]

[69] Y. Mukaigawa, S. Mihashi, and T. Shakunaga. Photometric image-based rendering for virtual lighting image synthesis. In *Proceedings of 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 115–124, 1999. [cited at p. 42, 43, 44, 49]

[70] J. Mutch and D. Lowe. Multiclass object recognition with sparse, localized features. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 58–65, 2006. [cited at p. 55]

[71] T. H. Myer and Ivan E. Sutherland. On the design of display processors. *Commun. ACM*, 11(6):410–414, 1968. [cited at p. 11]

[72] D. Pao, H. Li, and R. Jayakumar. Shapes recognition using the straight line hough transform: Theory and generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1076–1089, November 1992. [cited at p. 54]

[73] J. Parsons and J. Rolland. A non-intrusive display technique for providing real-time data within a surgeons critical area of interest. In *In Proceedings of Medicine Meets Virtual Reality*, pages 246–251, 1998. [cited at p. 21]

[74] G. Pekhteryev, Z. Sahinoglu, P. Orlik, and G. Bhatti. Image transmission over ieee 802.15.4 and zigbee networks. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 3539 – 3542 Vol. 4, may 2005. [cited at p. 28]

[75] S. Pessoa, G. Moura, J. Lima, V. Teichrieb, and J. Kelner. A global illumination and brdf solution applied to photorealistic augmented reality. *Virtual Reality Conference, IEEE*, 0:243–244, 2009. [cited at p. 48]

[76] B. Phong. Illumination for computer generated pictures. *Communications of the ACM*, pages 311–317, 1975. [cited at p. 154]

[77] W. Piekarski and B. Thomas. The tinmith system: demonstrating new techniques for mobile augmented reality modelling. *Aust. Comput. Sci. Commun.*, 24(4):61–70, 2002. [cited at p. vii, 4, 6, 26]

[78] T. Poston and L. Serra. The virtual workbench: Dextrous vr. In *Proceedings of Virtual Reality Software and Technology*, pages 111–121, 1994. [cited at p. 22]

[79] A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):918–923, 2003. [cited at p. 54]

[80] AR Wearable Projects. Mobile outdoor augmented reality. `http://wearables.unisa.edu.au/projects`, September 2009. [cited at p. vii, 19]

[81] H. Pryor, T. Furness, and E. Viirre. The virtual retinal display: A new display technology using scanned laser light. In *In Proceedings of 42nd Human Factors and Ergonomics Society Anual Meeting*, pages 1570–1574, 1998. [cited at p. 21]

[82] G. Reitmayr and T. Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *Proceedings of the 5th International Symposium on Mixed and Augmented Reality*, pages 109–118, 2006. [cited at p. 40]

[83] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005. [cited at p. 39]

[84] I. Sato, Y. Sato, and K. Ikeuchi. Illumination distribution from brightness in shadows: Adaptive estimation of illumination distribution with unknown reflectance properties in shadow regions. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 875–882, 1999. [cited at p. 44, 49]

[85] C. Schmandt. Spatial input/display correspondence in a stereoscopic computer graphics workstation. In *Proceedings SIGGRAPH Computer Graphics*, pages 253–261, 1983. [cited at p. 22]

[86] G. Sexton and X. Zhang. Suppression of shadows for improved object discrimination. In *IEEE Colloquium on Image Processing for Transport Applications*, pages 91–96, 1993. [cited at p. 54]

[87] I. Sexton and P. Surman. Stereoscopic and autostereoscopic display systems. *IEEE Signal Processing Magazine*, pages 85–99, 1999. [cited at p. 22]

[88] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proceedings of the International Symposium on Augmented Reality*, 2000. [cited at p. 39]

[89] D. Starkey and R. Morant. A technique for making realistic three-dimensional images of objects. *Behaviour Research Methods and Instrumentation*, pages 420–423, 1983. [cited at p. 24]

[90] A. State, G. Hirota, D. Chen, W. Garrett, and M. Livingston. Superiour augmented reality registration by integrating landmark tracking and magnetic tracking. In *Computer Graphics Forum*, pages 29–51, 2006. [cited at p. 79, 157]

[91] S. Teller. Distance methods. *Graphics Algorithms FAQ*, pages 40–58, 2000. [cited at p. 77]

[92] W. Thompson. Geometric reasoning for map-based localization, 1996. [cited at p. 40]

[93] A. Traub. Stereoscopic display using varifocal mirror oscillations. *Applied Optics*, pages 1085–1087, 1967. [cited at p. 24]

[94] Martin Vetterli, member Eurasip, and Henri J. Nussbaumer. Simple fft and dct algorithms with reduced number of operations. *Signal Processing*, 6(4):267 – 278, 1984. [cited at p. 50]

[95] D. Wagner and D. Schmalstieg. History and future of tracking for mobile phone augmented reality. In *International Symposium on Ubiquitous Virtual Reality*, pages 7 –10, 2009. [cited at p. 22]

[96] M. Walker. A look back at america's midnight spook shows. *Cool Hand Communications*, pages 3–15, 1994. [cited at p. 24]

[97] Y. Wang and D. Samaras. Estimation of multiple directional light sources for synthesis of augmented reality images. *Special issue on Pacific graphics 2003*, 65(4):185–205, 2003. [cited at p. 45, 49]

[98] Yang Wang and Dimitris Samaras. Estimation of multiple directional light sources for synthesis of mixed reality images. *Computer Graphics and Applications, Pacific Conference on*, 0:38, 2002. [cited at p. 46]

[99] T. Wiegand, D von Schloerb, and W. Sachtler. Virtual workbench: Near-field virtual environment system with applications. *Presence: Teleoperators and Virtual Environments*, pages 492–519, 1999. [cited at p. 22]

[100] L. Williams. Casting curved shadows on curved surfaces. In *SIGRAPH Computer Graphics*, pages 270–274, 1978. [cited at p. 79, 158]

[101] A. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983. [cited at p. 53]

[102] J. Yao and Z. Zhang. Systematic static shadow detection. In *Proceedings of the 17th International Conference on Pattern Regognition*, 2004. [cited at p. 54]

[103] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000. [cited at p. 152]

[104] L. Zhao and Y. Yang. Mosaic image method: A local and global method. *The IEICE Transactions Information and Systems*, pages 114–129, 1999. [cited at p. 53]

[105] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(7):680 –702, jul 1991. [cited at p. 42, 49]

[106] W. Zhou and C. Kambhamettu. Estimation of illuminant direction and intensity of multiple light sources. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 206–220, London, UK, 2002. Springer-Verlag. [cited at p. 46]

[107] W. Zhou and C. Kambhamettu. A unified framework for scene illuminant estimation. *Image Vision Comput.*, 26(3):415–429, 2008. [cited at p. vii, 46, 47]

Appendices

Appendix A

Camera Calibration

Camera calibration, or camera resectioning, is the process by which the parameters of a camera that captured an image frame are acquired. Camera calibration is possible if a three dimensional calibration pattern with spatially known features is placed within the camera view frustum. The correspondence between 2D and 3D can be established by extracting feature points from the two dimensional image. The result being a matrix that is based on the projection:

$$[su \quad sv \quad s]^T = C[XYZ1]^T \tag{A.1}$$

Often $[u \quad v \quad 1]^T$ are used to represent a 2D coordinate position and $[x_w \quad y_w \quad z_w \quad 1]^T$ represent a 3D point in virtual homogenous world coordinates. The matrix, C, allows for the mapping between a 3D point (x, y, z) and a 2D pixel (u, v). The value s is an arbitrary scale factor. C and s combined represent the intrinsic and extrinsic parameters of the camera. A pinhole camera model is assumed. The unknown parameters of C can be solved using the least squares method, providing that six or more correspondences between 2D and 3D points are found [21].

The mapping between world and pixel coordinates is fully expressed by:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[RT] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{A.2}$$

The intrinsic matrix contains five camera parameters that define the focal length, image format, and principle point:

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.3}$$

Focal length is represented by $\alpha_x = f \cdot m_x$ and $\alpha_y = f \cdot m_y$. m_x and m_y represent the scale factors that relate pixels to distance[34]. Some camera calibration implementations take into account any lens distortion effects by also estimating non-linear intrinsic parameters. The camera extrinsic parameters denote coordinate system transformations between 3D world and camera coordinates are and represented by R and T. The extrinsic parameters define the camera pose including the position of the camera and its heading. The camera projection matrix shown in equation (A.1) is derived from these intrinsic and extrinsic parameters. Methods of obtaining camera calibration are discussed by Zhang[103]

Appendix B

Lighting & Shading Calculation

Fixed function pipelines perform lighting and shading calculations on a per-vertex basis[*]. The colour of intermediate pixels of shaded surfaces are generated by interpolating between vertices during the rasterization process. Lighting and shading takes place as specified within the chosen shading model. The simplified OpenGL shading model is shown in figure B.1. It is comprised of the following components:

- Attenuation Factor
- Emission Term
- Diffuse Term
- Specular Term

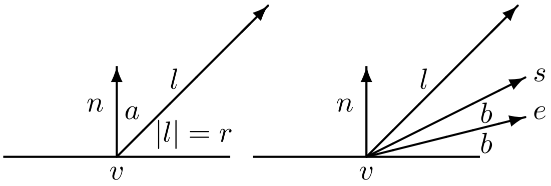


Figure B.1: OpenGL Lighting Model

The attenuation factor, F, selects the mode of attenuation to perform. The constants k_c, k_l, k_s in equation (B.1) are boolean values of either a 0 or 1. These values enable or disable the attenuation modes *constant*, *linear* and *square*

[*] Programmable pipelines are user-defined and beyond the scope of this text

distance respectively. The variable r is the distance between the vertex v and the light source l. Square distance attenuation is the most correct mode however the other modes are often used to mitigate rapid falloff for intensities at short distance to the light source. F is calculated as follows:

$$F = \frac{1}{k_c + k_l r + k_s r^2} \tag{B.1}$$

The emission term, E, describes the vertex ability to emit light. If this value is non zero then the vertex will create light emissions. The ambient term, A, represents ambient reflection of light. It is a multiplication of the ambient light property A_l and the ambient material property A_m as shown in the equation:

$$A = A_m A_l \tag{B.2}$$

The diffuse term, D, describes the diffuse reflection of light at the vertex v. It is dependent on the angle, a, between the direction to the light source, l, and the normal vector, n at v. Lambert's reflection law states $nl = cos(a)$[7]. D_m and D_l are constants that represent diffuse material and light properties respectively. Therefore D is defined as:

$$D = nlD_m D_l \tag{B.3}$$

The specular term, S, represents the reflective specular component. It calculates the half-vector $s = (l + e)/(|l + e|)$ between the vector pointing towards the camera, e, and the light vector, l. The exponent, i, controls the shininess of the surface. S_m and S_l are the specular and light properties respectively. The specular equation is therefore:

$$S = (ns)^i S_m S_l \tag{B.4}$$

Once all components are calculated the below lighting equation applies for n point light sources:

$$C = E + \sum_{i=0}^{n-1} F_i(A_i + D_i + S_i) \tag{B.5}$$

Where C is the resultant pixel colour vector at the given vertex, this calculation is repeated for every vertex being rendered. Once C has been obtained for all vertices, linear interpolation can be used in order to calculate pixel colour at intermediate pixel locations between vertices. There are three main interpolation and shading methods, *Gouraud*[30], *Flat* and *Phong*[76]. The Gouraud and flat shading models are supported by fixed function pipelines, including standard OpenGL. Phong shading requires per-pixel operations that are not supported by fixed function pipelines. Flat shading essentially copies the values from the vertex whereas Gouraud shading performs interpolation whereby vertex parameters such as lighting intensity, texture coordinates, normals and alpha values are linearly interpolated during rasterization. This provides approximate values for pixels within polygons. This avoids the calculating of all parameters for every single pixel. Phong shading requires programmable pipelines, such as offered by shader programming languages. The equation below shows the linear interpolation of a vertex property, $p_{u,v}$ at the *barycentric*[†] coordinates $u, v = [0, 1]$ between the constant values at the corners p_0, p_1 and p_2. In order to fill the entire triangle, its area is sampled at discreet u, v positions during rasterization as shown in figure B.2.

$$p(u, v) = (1 - u)(1 - v)p_0 + (1 - v)up_1 + (1 - u)vp_2 \tag{B.6}$$

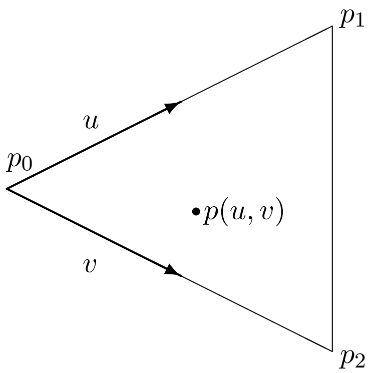


Figure B.2: Linear Interpolation of Vertex Properties Within a Polygon

[†] These are a form of homogeneous coordinates defined by the vertices of a simplex

Appendix C

Further Illumination Integration

Shadowing is a key component of any convincingly realistic three dimensional render. A number of techniques exist that allow an application to reproduce shadows providing that knowledge of both scene geometry and illumination information is available. Infact many shadowing techniques have been introduced since the field of computer graphics was born. This appendix discusses and compares some such techniques. The techniques discussed are summarized in table C.1. Those commonly adopted for AR use include that discussed by State et al[90] which proposes an AR system that favors the use of shadow maps whereas Haller[32] recommends the use of shadow volume techniques. Both methods allow AR applications to simulate shadows at low operational cost and do not inhibit realism when used within augmented reality systems. Plane projected shadowing techniques use two render passes in order to project a mesh onto a plane. The first pass renders the scene, including any geometry the light source and the plane on which to project the shadow. The second pass deals with the creation of shadows, projecting the geometry such that it looks like a shadow of the mesh. A suitable projection matrix can be created once the light position is defined. The plane on which to project is defined by three points. The shadow is then created by rendering all vertices on the ground plane. Figure C.1 shows how these points are projected onto the plane.

$\vec{L}$ represents the position of the light, $\vec{P}$ represents the position of a vertex that is to cast a shadow and $\vec{n}$ is the normal of the plane. $\vec{Q}$ is the resulting shadow point. A number of equations are evaluated in order to achieve the desired result. First a straight ray is cast from the light position and through the vertex position:

$$\vec{x} = \vec{L} + \lambda(\vec{P} - \vec{L}) \tag{C.1}$$

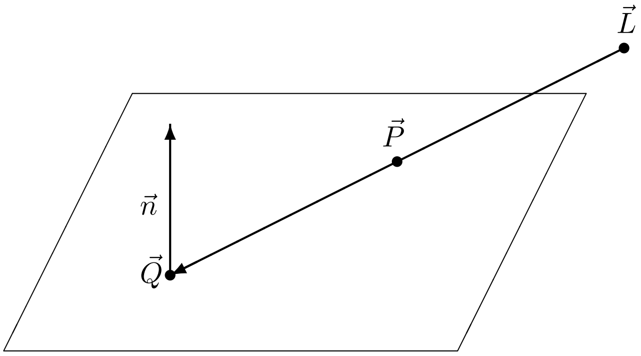


Figure C.1: Shadow Point Projection

Given the plane on which to cast the shadow:

$$(\vec{x} - \vec{E}) \cdot \vec{n} = 0 \tag{C.2}$$

Where $\vec{E}$ is a point on the plane. The point at which the ray intersects the plane can be calculated as below:

$$(\vec{L} + \lambda(\vec{P} - \vec{L}) - \vec{E}) \cdot \vec{n} = 0 \tag{C.3}$$

$$\lambda = \frac{\vec{E}\vec{n} - \vec{L}\vec{n}}{\vec{n}(\vec{P} - \vec{L})} \tag{C.4}$$

$$\vec{Q} = \vec{L} + \frac{\vec{E}\vec{n} - \vec{L}\vec{n}}{\vec{n}(\vec{P} - \vec{L})}(\vec{P} - \vec{L}) \tag{C.5}$$

The value Q is the shadow projected point, located on the plane, created by light source $\vec{L}$ and vertex position $\vec{P}$. This is repeated for each vertex that requires shadowing. The resulting geometry is then rendered last over the existing scene in a semi-transparent dark colour. The scene is then effectively shadowed. This technique assumes that shadows are projected only one single plane and not onto more complex geometry, self shadowing is not considered[19]. The projected shadow method[100] stores shadow data in a black and white texture that contains the shadows. This texture is created by rendering the scene from the point of view of the light source. When creating the texture, only shadow casting objects are rendered. The texture is then used instead of the flat projected mesh. This method allows shadows to be cast onto scene geometry that is located within the occluded region. Instead of projecting individual points it is the texture that is projected. The result is then rendered over the top of the scene to produce the desired effect. Objects need to be rendered in a hierarchial manner in order to render shadow receiving objects correctly. This requirement causes a resource overhead and significantly increases computational complexity greatly. As such, complex scenes may suffer from reduced frame-rates. Object self-shadowing is not possible with this technique. Shadow mapping techniques render the scene from the light source to the texture. A depth texture is generated and other data is rendered to the standard texture. By making use of the depth texture is it possible to self-shadow objects. Texture coordinates are created that are the same as the vertex coordinates and the desired result is obtained. These coordinates are then transformed into light coordinates. Shadow mapping techniques are intense when implemented on the CPU but recent advances in GPU technology allow for the rapid execution in hardware on the graphics card device. Vertex projection operates in a similar manor to plane projected shadows except that the shadow is calculated as follows:

$$s_i = \begin{bmatrix} p_x - (\frac{p_y}{l_y}) - l_x \\ h \\ p_z - (\frac{p_y}{l_y}) - l_z \end{bmatrix} \tag{C.6}$$

Where s is the projected vertex, p is the original vertex and l is the light position. This is repeated for each vertex that is being considered. The new x and z coordinates are calculated but the y value is simply set to the height value of the plane, h. With this technique shadows can only be projected onto planar surfaces. In order to improve efficiency, occlusion culling may be used to ensure that invisible vertices are not processed. Shadow volumes allow visually correct shadows to be rendered in real-time by employing the stencil buffer. They work by determining the volume of the generated shadow and creating a mesh representation of it. The base of this mesh is located at the point where the shadow is cast. From the camera position the lit and unlit points can be determined using the volume mesh and stencil buffer information. Usually, no additional computation is required in order to make use of the stencil buffer as it is enabled by default in most graphics implementations. This technique is very computational intense, especially when casting shadows of detailed meshes. This is due to the complexity of the shadow volume and the calculations that are required for its generation. Complex mesh objects are used when creating realistic scenes, however they are neither required or desired when generating realistic shadows via this approach. Infact such models would present a problem by vastly increasing the computational complexity of the shadow render pass. By creating a copy of the mesh and reducing its resolution a simpler, optimized shadow volume mesh can be calculated. Such a model is sufficient for use within the shadowing process. Models may be rendered at full resolution, then optimized prior to shadowing in order to improve performance. A tradeoff exists between computational complexity and shadow realism. The shadow volume method allows for self-shadowing where any region that falls within the 3D shadow mesh is to be shaded. Hybrid techniques that make use of combined shadow generation methods can be imple-

Plane Projected	Projected Shadows
Fast, low computational complexity High detail No self-shadowing No shadow receivers	Fast, very few calculations Detail is texture dependant No self-shadowing Shadow receivers
Vertex Projection	Shadow Volumes
Slow with high-resolution meshes High detail No self-shadowing No shadow receivers	Slow, many calculations High Detail Self-shadowing Shadow receivers
Depth Shadow Mapping	Hybrid Shadows
Very fast, very few calculations Detail depends on texture Self-shadowing Shadow receivers	Benefits of all chosen techniques Mitigates negative aspects of each technique

Table C.1: Comparison of 3D Shadow Emulation Techniques

mented in order to provide robustness to variant conditions. A tradeoff generally exists between performance and shadow realism. A technique may be adaptive, switching between methods depending on the situation at hand. One common example of a hybrid approach is shadow volume reconstruction which is a technique often used as a compromise between depth shadow mapping and shadow volumes. Here, a depth texture is rendered through the depth buffer and then a contour of the shadow is determined using depth values from this texture. Using this and the position of a light it is possible to construct a volume. This volume is used as a shadow volume. This technique offers a good medium between depth shadow mapping and shadow volume methods. The realism is an improvement on depth shadow mapping but not as accurate as with shadow volume, however operational complexity is much lower than with the shadow volume technique. Feng[22] identifies a number of illumination methods for augmented reality and classifies them into two categories. These are common illumination and relighting. Common illumination matching techniques attempt to simulate consistent lighting when artificial objects are inserted into a real context. Relighting techniques modify the real component in response to the insertion of a virtual object. This technique collects illumination parameters such as the Bidirectional Reflectance Distribution Function (BRDF) from the real scene for use within the virtual. The technique requires that approximate knowledge of real scene geometry be known prior to augmentation.

Copyright Notice