EXTRACTION AND INTEGRATION OF PHYSICAL ILLUMINATION IN DYNAMIC AUGMENTED REALITY ENVIRONMENTS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

A'aeshah A. Alhakamy

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2020

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Mihran Tuceryan, Academic Advisor

Department of Computer and Information Science

Dr. Shiaofen Fang, Department Chair

Department of Computer and Information Science

Dr. Jiang Ya Zheng

Department of Computer and Information Science

Dr. Snehasis Mukhopadhyay

Department of Computer and Information Science

Approved by:

Dr. Shiaofen Fang

Head of the Graduate Program

I dedicate this dissertation to my children

Eyad, Renad, and Nehad

for keeping me grounded and reminding me of how special and important life can be.

ACKNOWLEDGMENTS

First of all, I would like to express the deepest gratitude to my advisor, Professor *Mihran Tuceryan* for his continuous support, constructive guidance, and valuable motivation throughout the production and development of my study and research. Professor *Tuceryan* is an outstanding advisor, and I consider myself extremely fortunate to have him as my mentor. I would also like to thank my committee members, Professor *Shiaofen Fang*, Professor *Jiang Yu Zheng*, and Professor *Snehasis Mukhopadhyay* for agreeing to serve on my committee.

Sincere thanks go to my parents, for being eminently supportive at every stage of my life and at pursuing higher education overseas. The fundamental role of my parent inputs and sacrifices from my earliest days of school enabled me to reach my goals. I deeply thanks my father, *Abdullah* for his continuous encouragement that keeps me motivated and eager to pursue a Ph.D. degree. My dedicated mother, *Bushra* has never stopped believing in me and I will always be grateful to have her in my life. My dream is to see them both on my graduation despite their long-time divorce and differences.

I am deeply indebted to my family, my husband *Majed* and my sweet three children *Eyad, Renad, and Nehad*, for their unconditional love and support throughout this journey. Thank you, my angels for being patient and getting busy while I spent several hours "deducted from your joy-filled family times" to be at work.

Many thanks to my brother *Ayman* for his inspirational involvement and invaluable support over the years of my study. I also would like to thank my sister *Ashjan* for her words of motivation during my hardest times. I am very grateful for my sister *Asma*, your life story turns my weakness into a strength.

I am very honored to be one of the recipients of the *Saudi Arabian Cultural Mission (SACM)* and the *University of Tabuk* generous sponsorship. The scholarship gives

me an opportunity to earn skills that will serve higher education in Saudi Arabia following graduation. Thank you personally for your generosity: Without scholarship patrons willing to support graduate studies, students such as myself would be unable to pursue advanced education abroad.

I extend my gratitude to my undergraduate Professor *Iman Alansari*, and my colleagues at the University of Tabuk Professor *Rabab Ramadan*, Professor *Mohammed Alwakeel*, Professor *Zaid Bassfar*, in addition to my master Professor *Mihran Tuceryan*, Professor *John Gersting* and Professor *Rajeev Raje* for providing letters of recommendation to earn an entry into graduate studies at this prestigious university.

I would like to thank my friends, *Abeer, Ghada, Aljohara, and Felix.* I would also like to thank the *Saudi student clubs* around the United States for the assistance given during my academic life abroad. Last but not least, The assistance of *Advanced Visualization Lab (AVL)* and *Visual Information Sensing and Computing Center* in providing a 360° camera, Kinect camera, and other gadgets, is much appreciated.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

SYMBOLS

Pb	basic pixel color
Pp	polarized pixel color
(x, y)	screen coordinates
(θ, ϕ)	spherical coordinates
$\|\uparrow\rangle \langle\uparrow\|$	vertical polarizing filter
$\|\rightarrow\rangle \langle\rightarrow\|$	horizontal polarizing filter
$\|\nearrow\rangle \langle\nearrow\|$	angular polarizing filter
$\|v\rangle$	vertical vector
$\|h\rangle$	horizontal vector
$\|\Theta\rangle$	angular vector, set of $\|v\rangle$, $\|h\rangle$
$\hat{V}$	vertical polarizing filter matrix
$\hat{H}$	horizontal polarizing filter matrix
$\hat{\Theta}$	angular polarizing filter matrix
i	light velocity in vacuum
r	light velocity in the medium
(n_u, n_v)	width/height of tangent map

$\overrightarrow{N}$	normal vector/dir
$\overrightarrow{L}$	light vector/dir
$\overrightarrow{h}$	halfway vector/dir
$\overrightarrow{V}$	viewpoint vector/dir
$\overrightarrow{T}$	tangent vector/dir
$\overrightarrow{B}$	BiTangent vector/dir
L	light source
P	feature point
ρ	transformed point
c	light color
f	diffuse factor
a	attenuation
S_c	specular color
S_f	specular factor
S_p	specular power

ABBREVIATIONS

AR	Augmented Reality
VR	Virtual Reality
MR	Mixed Reality
GPU	Graphics Processing Unit
BRDF	Bidirectional Reflectance Distribution Function
SfM	Structure from Motion
CNN	Convolution Neural Networks
SH	Spherical Harmonics
ICP	Iterated Closest Point
COM	Concurrent Odometry and Mapping
SLAM	Simultaneous Localization and Mapping
VPLs	Virtual Point Lights
IBL	Image-Based Lighting
DOF	Degree-of-Freedom
API	Application Programming Interface
HDR	High Dynamic Range
LDR	Low Dynamic Range
FoVs	Field-of-Views
HMDs	Head-Mounted Displays
SDF	Signed Distance Function
BSP	Binary Space Partitioning tree
PCA	Principal Component Analysis
MST	Minimal Spanning Tree
IMU	Inertial Measurement Unit

ABSTRACT

Alhakamy, A'aeshah A. Ph.D., Purdue University, December 2020. Extraction and Integration of Physical Illumination in Dynamic Augmented Reality Environments. Major Professor: Mihran Tuceryan.

Although current augmented, virtual, and mixed reality (AR/VR/MR) systems are facing advanced and immersive experience in the entertainment industry with countless media forms. Theses systems suffer a lack of correct direct and indirect illumination modeling where the virtual objects render with the same lighting condition as the real environment. Some systems are using baked GI, pre-recorded textures, and light probes that are mostly accomplished offline to compensate for precomputed real-time global illumination (GI). Thus, illumination information can be extracted from the physical scene for interactively rendering the virtual objects into the real world which produces a more realistic final scene in real-time. This work approaches the problem of visual coherence in AR by proposing a system that detects the real-world lighting conditions in dynamic scenes, then uses the extracted illumination information to render the objects added to the scene. The system covers several major components to achieve a more realistic augmented reality outcome. First, the detection of the incident light (direct illumination) from the physical scene with the use of computer vision techniques based on the topological structural analysis of 2D images using a live-feed 360° camera instrumented on an AR device that captures the entire radiance map. Also, the physics-based light polarization eliminates or reduces false-positive lights such as white surfaces, reflections, or glare which negatively affect the light detection process. Second, the simulation of the reflected light (indirect illumination) that bounce between the real-world surfaces to be rendered into the virtual objects and reflect their existence in the virtual world. Third, defining the

shading characteristic/properties of the virtual object to depict the correct lighting assets with a suitable shadow casting. Fourth, the geometric properties of real-scene including plane detection, 3D surface reconstruction, and simple meshing are incorporated with the virtual scene for more realistic depth interactions between the real and virtual objects. These components are developed methods which assumed to be working simultaneously in real-time for photo-realistic AR. The system is tested with several lighting conditions to evaluate the accuracy of the results based on the error incurred between the real/virtual objects casting shadow and interactions. For system efficiency, the rendering time is compared with previous works and research. Further evaluation of human perception is conducted through a user study. The overall performance of the system is investigated to reduce the cost to a minimum.

1 INTRODUCTION

Integrating virtual objects into dynamic augmented reality scenes in a photo-realistic manner that is indistinguishable from the real world, has been one of the major goals in computer graphics and computer vision. Image/video composition to achieve such a task is challenging due to visual coherence between the virtual and real objects in the final scene.

The extraction of correct and precise illumination information from the physical scene can provide the means to realistically render virtual objects in the final scene. Acquiring an illumination model featuring accurate light source information that would capture the whole real environment also can be challenging under limited experimental assumptions. A photo-realistic and dynamic scene in augmented reality that integrates the various illumination models requires addressing several aspects in each frame to be viable.

Dynamic illumination is a problem that studies light transportation from the light sources and the light reflected from other objects under varying scenes conditions. The two types of light transport contribute to the final image which is produced by the rendering process. Usually, the real-time algorithms for illumination are developed to allow fast computation with a minimum cost of error. This Thesis investigates the visual coherence problem in augmented reality by developing a system that provides real-time dynamic illumination for interactive augmented reality based on the virtual objects appearance in association with the real objects and other criteria in the scene.

The proposed algorithm estimates and detects the incident light (direct illumination) in the physical scene through live-feed 360° camera that is instrumented on any Augmented reality (AR) device (e.g., a handheld mobile device, a head-mounted display, or webcam camera) to capture the entire environment map using computer vision techniques. A physics-based light polarization technique is utilized to reduce

or better absorb any false positive lights from white surfaces, reflections, or glare to detect the incident lights with fewer errors. The developed system simulates the reflected light (indirect illumination) from surfaces in the rendering of virtual objects through investing two separate sub-methods which provides totally different outcomes for the internal interaction among the real and virtual objects, see Figure 1.1

The system computes various shading properties for each virtual object in every frame. These properties include material textures, shadows, specular and diffuse illumination, reflection, refraction, and Fresnel. Moreover, an improvement of the physics-based illumination through polarization is presented which allows more accurate light source estimation for the virtual objects. In order to provide a credible sense of the physical environment in AR, the geometric properties of real-scene including plane detection, 3D surfaces reconstruction and simple meshing can be incorporated into the virtual scene in real-time for realistic interaction between the virtual and real objects without the use of image-based markers.

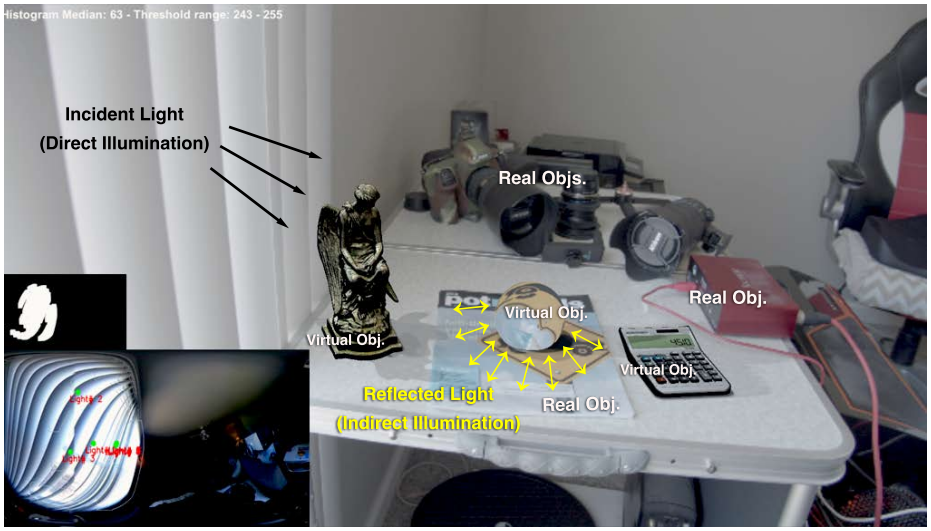

Figure 1.1. The difference between the direct illumination where the incident light from the source hits the objects directly, while the indirect illumination is a reflection of lights that is bouncing among the objects virtual or real.

Performance in real-time is a crucial requirement in AR systems. The interaction with the virtual objects, rendering, and registration methods must work in real-time. The challenge of fully estimating the direction of incident light with the simulation of reflected light, in addition to defining the shading properties demand high-performance cost. Thus, real-time and dynamic illumination models are of high interest in computer vision and graphics. This thesis evaluates and improves multiple methods to provide an effective illumination model for AR systems and applications.

1.1 Thesis Statement

This research investigates and develops a system that provides an illumination model for visual coherence in augmented reality with a dynamic environment in real-time where virtual objects appear as a part of the real world, using computer vision and graphics methods for estimating the incident light (direct illumination) after applying physics-based light polarization technique, simulating the reflected light (indirect illumination), defining the shading properties of the virtual object as if it actually exists in the physical scene, and include some geometric properties of the real scene such as plane detection, 3D surfaces reconstruction, and simple meshing into the virtual scene while reducing the overall performance cost.

1.2 Hypothesis

It is possible for a realistic virtual object to simulate an accurate and precise dynamic illumination by only instrumenting a live-feed 360° camera on an AR device (e.g., a handheld mobile device, a head-mounted display, or webcam camera) with the use of computer vision and computer graphics techniques, even when lighting conditions and environment setting are changing dynamically. By separating the system into several parts: (1) incident light polarization and estimation, (2) reflected light simulation, (3) shading properties definition, (4) geometric properties of physical-scene, including plane detection, 3D surfaces reconstruction and simple meshing which

must work simultaneously, a more accurate dynamic illumination model could be achieved.

1.3 Contributions

This dissertation formulates and explores real-time methods to augment a live-feed camera video with a virtual object that depicts the real lighting conditions in the scene while at the same time influences the final image background and surfaces in a dynamic environment. In the subsequent chapters, the following contributions of this work are presented. Figure 1.2 illustrates the system structure and how it evolved to produce the final image.

- Estimate and detect the incident light (direct illumination) direction and angle from a live-feed 360° panoramic image view, with unknown scene geometry and light position.
- Present a novel physics-based lighting in the augmented reality system using polarization properties of light, in order to eliminate/reduce false-positive lights from reflections, white surfaces, or glare.
- Investigate the Simulation of the reflected light (indirect illumination) methods in a dynamic environment and evaluates the overall performance of the rendering process in order to reduce the cost.
- Define specific shading properties for each virtual object that is affected by the previous methods of light source estimation and passing reflecting lights from the surroundings for realistic final perception.
- Explore the geometric properties of real-scene including plane detection, 3D surfaces reconstruction and simple meshing so the virtual objects can recognize the depth information providing far more realistic interactions.

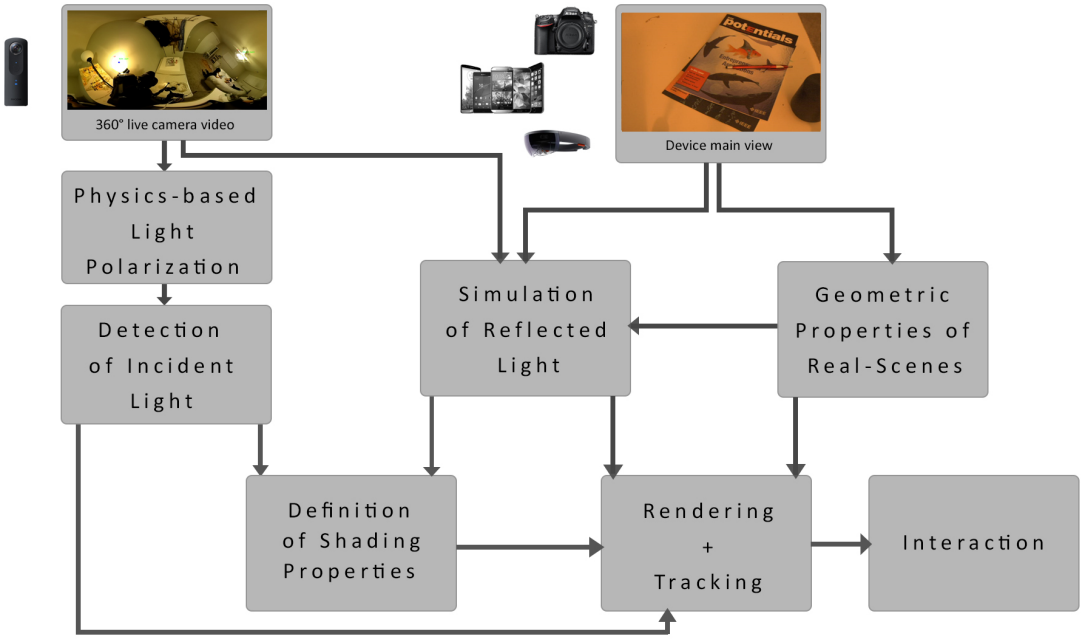

Figure 1.2. General overview of the system components which present the evolution of the entire structure: (1) light polarization, (2) incident light estimation, (3) reflected light simulation, (4) shading properties definition, (5) physical geometric properties of real-scenes

1.4 Scientific Publications

Key results of this dissertation are published in conference proceedings and journals. A reprint with minimal edit was incorporated with the following direct relevant publications. As stated on the copyright and consent form, the "Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgment is given to the original source of publication."

1. A'aeshah Alhakamy and Mihran Tuceryan. 2019. AR360: Dynamic Illumination for Augmented Reality with Real-Time Interaction. In 2019 IEEE 2nd

International Conference on Information and Computer Technologies (ICICT). 170-174. https://doi.org/10.1109/INFOCT.2019.8710982 [1].

2. A'aeshah Alhakamy and Mihran Tuceryan. 2019. CubeMap360: Interactive Global Illumination for Augmented Reality in Dynamic Environment. In 2019 SoutheastCon. IEEE, IEEE Press, 1-8. https://doi.org/10.1109/SoutheastCon42311.2019.9020588 [2].

3. A'aeshah Alhakamy and Mihran Tuceryan. 2019. Polarization-Based Illumination Detection for Coherent Augmented Reality Scene Rendering in Dynamic Environments. In Proceedings of the 36th Computer Graphics International (CGI19) in cooperation with ACM SIGGRAPH and Eurographics, chapter in (Advances in Computer Graphics). Springer International Publishing, Cham, 3-14. https://doi.org/10.1007/978-3-030-22514-8_1 [3].

4. A'aeshah Alhakamy and Mihran Tuceryan. 2019. An Empirical Evaluation of the Performance of Real-Time Illumination Approaches: Realistic Scenes in Augmented Reality. In Augmented Reality, Virtual Reality, and Computer Graphics (AVR'19). Springer International Publishing, Cham, 179-195. https://doi.org/10.1007/978-3-030-25999-0_16 [4].

5. A'aeshah Alhakamy and Mihran Tuceryan. 2020. Physical Environment Reconstruction Beyond Light Polarization For Coherent Augmented Reality Scene On Mobile Devices. In Transactions on Computational Science XXXVII: Special Issue on Computer Graphics. Springer Berlin Heidelberg, 19-38. https://doi.org/10.1007/978-3-662-61983-4_2 [5].

6. A'aeshah Alhakamy and Mihran Tuceryan. 2020. Real-Time Illumination and Visual Coherence for Photorealistic Augmented/Mixed Reality. ACM Comput. Surv. 53, 3, Article 49 (May 2020), 34 pages. https://doi.org/10.1145/3386496 [6].

Additional Publication:

7. Abeer Alsaiari, Ridhi Rustagi, A'aeshah Alhakamy, Manu Mathew Thomas, Angus G Forbes, et al.2019. Image Denoising Using A Generative Adversarial Network. In 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT). IEEE, 126-132. https://doi.org/10.1109/INFOCT.2019.8710893 [7].

8. Francesco Cafaro, Milka Trajkova, and A'aeshah Alhakamy. 2019. Designing Embodied Interactions for Informal Learning: Two Open ResearchChallenges. In Proceedings of the 8th ACM International Symposium on Pervasive Displays (PerDis 19). ACM, New York, NY, USA, Article 28, 2 pages. https://doi.org/10.1145/3321335.3329688 [8].

9. Milka Trajkova, A'aeshah Alhakamy, Francesco Cafaro, Rashmi Mallappa, and Sreekanth R. Kankara. 2020. Move Your Body: Engaging Museum Visitors with Human-Data Interaction. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI 20). Association for Computing Machinery, New York, NY, USA, 1-13. https://doi.org/10.1145/3313831.3376186 [9].

10. A'aeshah Alhakamy, Milka Trajkova, Francesco Cafaro, Rashmi Mallappa, Sreekanth R. Kankara, and Vedak Sanika. 2020. Design strategiesand optimizations for human-data interaction systems in museums. In Proceedings of the 2020 International Conference on Advanced Learning Technologies (ICALT 20). IEEE. Presented, in print. https://doi.org/10.1109/ICALT49669.2020.00081 [10]

1.5 Thesis Outline

The chapters of this dissertation are organized as follows:

- *Chapter 2* discusses the fundamentals of visual coherence, light models, and related work on real-time illumination algorithms, setups of augmented and mixed reality, the real scene physical objects tracking, and reconstruction.

- *Chapter 3* presents a general overview of the research design and methodology structure of algorithms and concepts used for extracting and integrating the physical illumination in dynamic augmented reality environments.

- *Chapter 4* investigates the methods and techniques for detecting the incident lights directly from the physical scene using radiance map captured through a live-feed 360° camera. Also, the use of light polarization is discussed in this chapter and show how it support reducing or eliminating the false-positive lights.

- *Chapter 5* shows the algorithms for estimating the reflected lights that is bouncing between the virtual and real objects which are extracted from the local regions surrounding each object of concern, then is applied through the image-based lighting mode.

- *Chapter 6* covers how the virtual objects materials and features are defined to create shading properties where specific shading programs are created to meet the system requirements.

- *Chapter 7* discusses the geometric properties of real-scene including plane detection, 3D surfaces reconstruction and simple meshing in order to enable the depth data available in the physical world so the virtual objects can interact with the real object without AR markers.

- In *Chapter 8* the whole system is evaluated in multiple lighting conditions the accuracy of results based on the error incurred between the real/virtual objects casting shadow and interactions. The rendering time is considered in order to develop more efficient algorithms. Further evaluation of human perception is conducted through a user study. The overall cost of the system is reduced to maximize system performance.

- *Chapter 9* concludes the work in a precise summary of the research development and adaptation for our illumination extraction and integration model in

augmented reality. Also, the future work is discussed briefly for potential development.

2 THEORETICAL BACKGROUND AND LITERATURE REVIEW

A photo-realistic final scene in augmented, mixed, or virtual realities would not be possible without a correct illumination model. This chapter is dedicated for informing the reader with the important background and terminology of realistic rendering and illumination notions for a comprehensive review. Visual coherence and computer graphics cues for the scene structure in augmented reality is discussed in section 2.1. Rendering equation and light model are covered in section 2.2, while rendering paths are explained in section 2.3. Section 2.4 highlights the composition of the augmented scene with the real world to produce the final image. A quick overview of the light polarization is explained in section 2.5. This chapter is a reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

2.1 Visual Coherence

The techniques used to combine the real and virtual objects are investigated to know how the virtual objects are blended seamlessly into the physical environment to the extent where they are immersed in the real world. Although many AR applications do not focus on the visual coherence as the ultimate goal, it is significant to achieve seamlessness in entertainment, commerce, and education. The issue related to appearance, generally depending on techniques from photo-realistic computer graphics in real-time.

The ability to embed a three-dimensional object into an image of the real scene means that objects should be rendered from a virtual camera with internal and external parameters that correspond to the physical camera. Fundamental depth cues can be obtained with a calibrated camera. Information about the depth provides

an interpretation of a three-dimensional structure from the viewpoint of the camera. Depth cues can be categorized as monocular or binocular where there are around 15-20 different depth cues. Cues from a single image are known as monocular, see Figure 2.1), while cues depicted in a pair of images are called binocular. [11, 12]

At present, several AR displays use a monocular video see-through mode. In general, for AR, the most important clue is the depth that can be produced by computer graphics software. Also, there are other essential depth cues such as:

- Relative size: the distance between the objects and the observer. The further it is, the smaller it appears.
- Relative height: how far is the object from the other objects and where their base is higher in the image.
- Perspective: the converging of the parallel lines when drifting away from the observer.
- Surface detail: objects closer to the observer has more texture gradient and fine-grained surface detail.
- Atmospheric attenuation: while closer objects appear clearer, most-distant objects can be blurred based on the atmospheric effects.
- Occlusion: in the screen space, closer objects obscure the further ones.
- Shading: The illumination fall on the objects based on the direction and location of the light sources.
- Shadow: objects blocking the light cast shadow on other objects.

These cues are delivered by the well-equipped three-dimensional computer graphics tools. Some of them are more straightforward to produce by a virtual camera registered geometrically with a real one, such as size, perspective, height, and surface details. Atmospheric attenuation concerns far-field outdoor AR. However, the

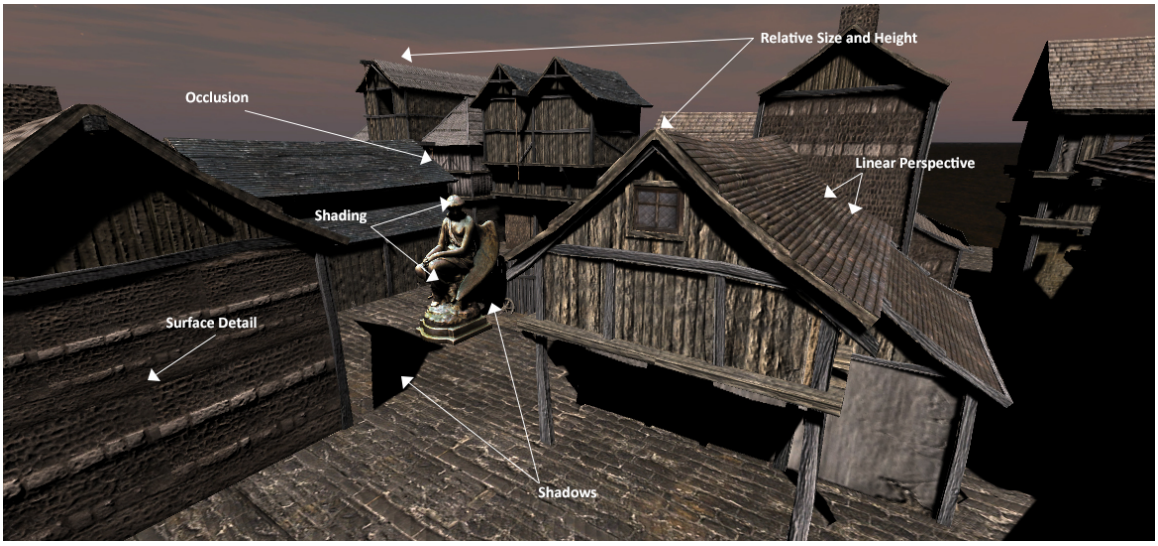

Figure 2.1. Interpretation of the scene structure from a single image through the monocular depth cues [6]

other cues, especially occlusion, shading, and shadows demand attention in the AR rendering process.

Combining the real and virtual worlds in AR/MR extends the conventional rendering process in the computer graphics pipeline to involve more steps. The video see-through pipeline is better suited in this research than an optical see-through pipeline, which consists of the following stages:

1. **Acquisition.** obtain a model or a set of data from the real scene such as geometry, materials, and illumination.

2. **Registration.** transform the obtained sets of data which is the standard photometric and geometric properties of one coordinate system of the real and virtual scenes.

3. **Compositing.** merge the virtual objects and the real physical environment into one single image scene.

4. **Display.** provide the user with the composited image.

The rendering process in AR/MR obviously is more complicated than the standard pipeline in computer graphics. The AR rendering pipeline involves with a virtual scene and a real scene simultaneously to provide geometric and photo-metric registration for both scenes [11].

2.2 Rendering Equation and Light Model

The concept of visual coherence and light transport required a mathematical representation for better understanding. The exploration of three-dimensional rendering engines makes one familiar with the lighting models, and general notions such as albedo, ambient lights, specular reflections, and diffuse colors. The reflection model of the diffuse surface is the most straightforward lighting model also known as dot product lighting. The intensity of each light source which is the RGB color that is represented in the following equation:

$$I = \text{surfaceColor} \times \sum_{i=1}^{nlights} \text{LightIntensity}_i \times (N \cdot L_i) \tag{2.1}$$

The first fragment calculates the total of the incoming lights from every direction, scale it by the angle cosine between N surface normal and L light source. Then, multiply the outcome by the reflection function for the diffuse surface which is a constant color. [13]. This is the purest form of the rendering equation based on physics which is only for producing images in computer graphics. It is a meaningful standard by which the entire realistic lighting must be measured. The following formula is a more general form of rendering equation which represents an integral over a directions hemisphere where L is the function of the light intensity, see Figure 2.2 [14, 15].

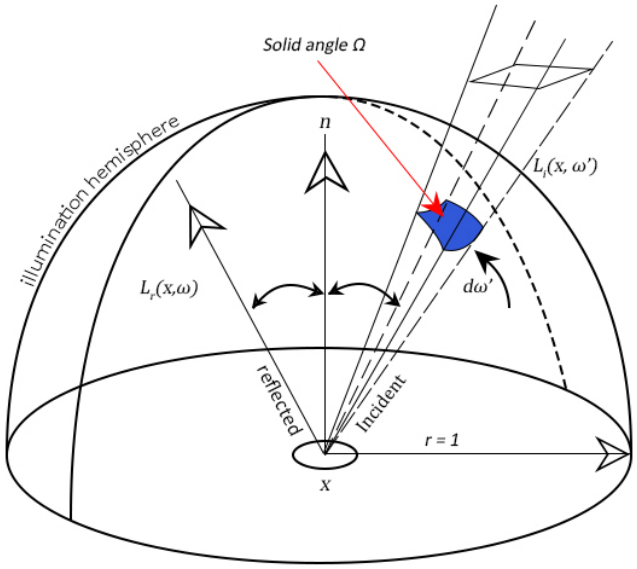

Figure 2.2. A visual representation for the rendering equation where a solid angle Ω can be calculated by the projection of surface area under the hemisphere [6]

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w'}, \vec{w}) L_i(x, \vec{w'}) |\vec{n}.\vec{w'}| dw' \tag{2.2}$$

Where:

- $L_o(x, \vec{w})$ refers to the outgoing radiance from position x into the direction $\vec{w}$.
- $L_e(x, \vec{w})$ represents the radiance emitted from position x to the direction $\vec{w}$.
- $\int_\Omega f_r(x, \vec{w'}, \vec{w})$ is the bidirectional reflectance distribution function (BRDF) of the surface at point x that defines how light is reflected at that surface which transfer the direction of incoming radiance $\vec{w'}$ to the direction of reflected lights $\vec{w}$ on the hemisphere Ω.
- $L_i(x, \vec{w'})$ is the incident light or the incoming radiance from the direction $\vec{w'}$.
- $\vec{n}.\vec{w'}$ refers to the dot product which is the cosine of the angle between the direction of incoming lights $\vec{w'}$ and surface normal $\vec{n}$.

- dw' represents the differential angle of incoming radiance [16].

This object-light interaction is based on two main characteristics: the light properties, and object material properties, which is known as the lighting model. There are multiple lighting models that have been developed over the years and the most popular one is the basic lighting model. In this model, the color of the object surface is the sum of four properties which will be mentioned in several sections through this work.

$$\text{Surface Color} = \text{Emissive} + \text{Ambient} + \text{Diffuse} + \text{Specular} \tag{2.3}$$

- *Emissive* - the light source which have the power to emit matter and energy.
- *Ambient* - uniform distribution of several times bouncing lights in all directions, in other words, the minimal light intensity or no light in the overall scene.
- *Diffuse* - a surface subset that has irregular micro-facets where lights are reflected in different directions.
- *Specular* - a surface subset with aligned micro-facets where the lights are reflected in a similar and few directions [17].

The lighting calculation can be achieved in two ways through the shader language which is a simulation made on a GPU code at the surface level that makes the final image look realistic to the human eyes.

- Per Vertex Lighting: the calculation is done in the vertex shader.
- Per Fragment Lighting: the calculation is done in the fragment shader.

2.2.1 Illumination Types

The calculation of light in the scene can be one or more of the following types:

- **Direct Illumination** - Also known as the incident light, where the light hits the surfaces from the source and is reflected to the observer/camera directly. The calculation of the light emitted and reflected from the surfaces towards the camera includes the shadow. The contribution of light energy makes direct illumination a major component in the rendering process.

- **Indirect Illumination** - refers to the reflected light, where the light bouncing from one object surface hits another object after traveling from the light source. So, it consists of multiple bounces which result in higher computational cost. However, it is considered an important part of light transport and shadowing regions in the scene which provide more realistic images. Figure 2.3 shows how the same scene is rendered with (a) direct light only, and (b) both direct and indirect lights.

- **Local Illumination** - This type of illumination depends on the local objects and light source (direct illumination) only. It is usually used in real-time applications due to the reduced calculation cost. It can be sufficient when the lights have a small consideration in the application but because the light is calculated on a local surface point and the interaction with the other surfaces cannot be included in the scene, this lighting model can be very limited with discreet light source.

- **Global Illumination** - For achieving a realistic rendering, global illumination is essential where at any point the illumination can depend on any other point in the scene. This lighting model includes both direct and indirect illumination as seen in Figure 2.3 where the reflections of the light on all the surfaces in the scene are calculated.

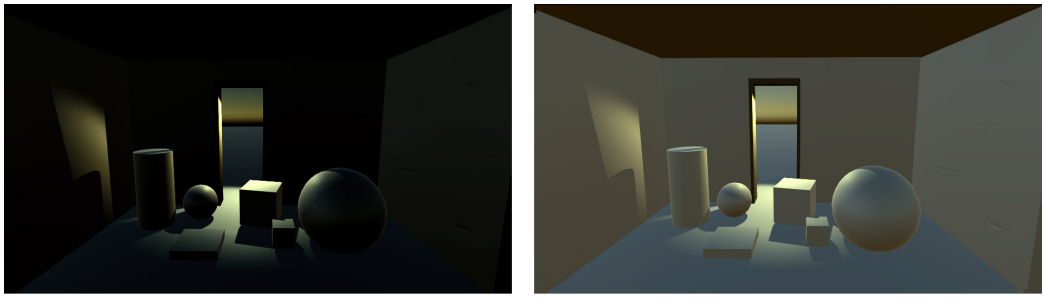

Figure 2.3. The same scene rendered under the consideration of (a) direct illumination only and (b) both direct and indirect illumination.

2.2.2 Forms of Light

In the physical world, the light is emitted from a 3D surface. In the digital world, an approximation of the real-life light is used to reduce the computational cost, see Figure 2.4. Listed below are the most known approximations and forms of light.

- **Directional Light** - This is the simplest type of light which is defined by a direction vector. The sun is a perfect example of the directional light where the direction of the sunlight is the same for all the objects in the scene. This effect is obtained by assuming the sun is a point light source infinitely far away. In many modelers, the directional light might appear as a local point that can be moved around the scene, but the only value that is being manipulated is actually the direction of the light and not its location. There is a limit to each color channel intensity which is represented based on the system used in one's application. Some systems have an intensity limit range of [0-1] others have a range [1-11] or it could be represented as a negative value which indicates that the light sucks photons from the surface.

- **Point Light** - Define a position in space that emits light off in all directions. A night lamp is a good example of this type of light. The light color and intensity

can be set easily as the directional light. These lights are different from the real-life lights in that by default the distance from the light does not affect its brightness. Attenuation is not considered because rendering is faster as long the light distance has a small impact on this type of light. However, a different equation can be used to attenuate the light based on distance. For instance, divide the intensity by the distance or the distance squared, which is how real light works. Some systems use entirely non-physical methods but are considered useful for artistic control of the lighting. The point light can be added in several local areas in the scene without affecting every other lighting area.

- **Spot Light** - This type of light is used in order to make an object or part of the scene the focus of attention. In real life, a theater stage light is one example of the spotlight. This emitter relies on the position just like the point light. However, the additional feature is controlling the cone of light that has been formed through the angle where the spotlight ends. Also, controlling the fall off exponent which is similar to how specular highlighting works.

- **Area Light** - The best form of the natural light is this form of light, but it comes with an expensive computational cost. The real emitter could be three-dimensional which has an area. Some systems do not have area light for real-time because of the complexity with the rendering computations but can be performed only using baked light-maps.

2.3 Rendering Paths

The technique, workflow, or path that used to render a lit scene are covered in this section.

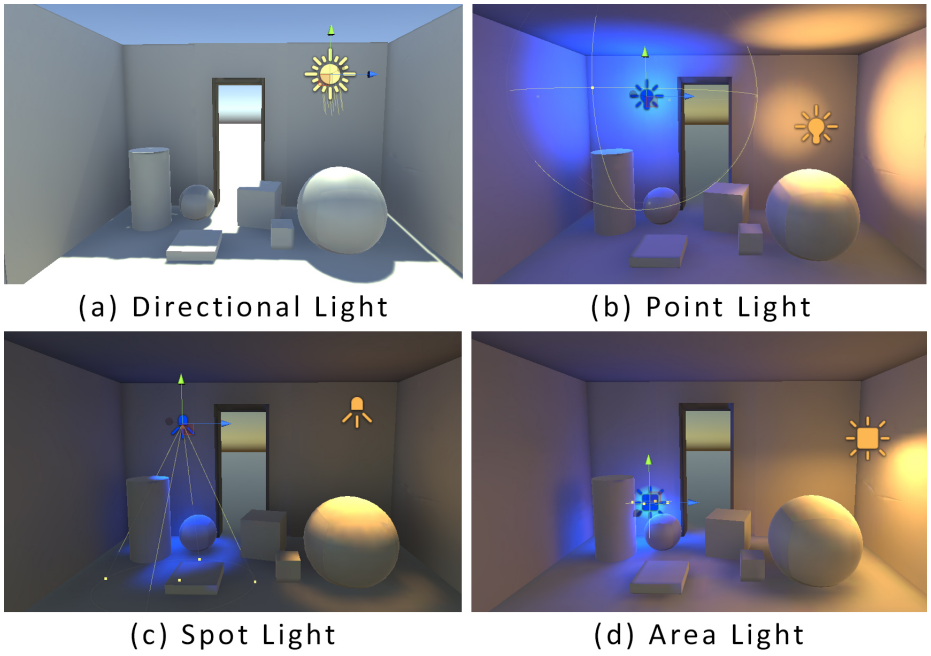

Figure 2.4. The approximations or form of nature-world lighting in the pixel-world of 3D systems.

2.3.1 Forward rendering

The idea of this path is to draw every object's mesh once per light source. Then combine each draw call by adding the color contribution of the light to the final lit image or scene. Thus, it renders a surface where the fragment color for each pixel is stored for the closest visible object. Figure 2.5 demonstrates how the forward rendering works in example with four objects and four lights. In the example, (1) the first object is rendered with the blue light on it as it influences only one object, (2) the yellow light color contribution is influencing the first three objects, (3) the green light, based on its characteristics, is influencing the second and third objects, (4) the pink light is influencing the last two objects, In ordinary condition the forward rendering will cause 16 draw calls but current engines optimization techniques are implemented to reduce performance cost.

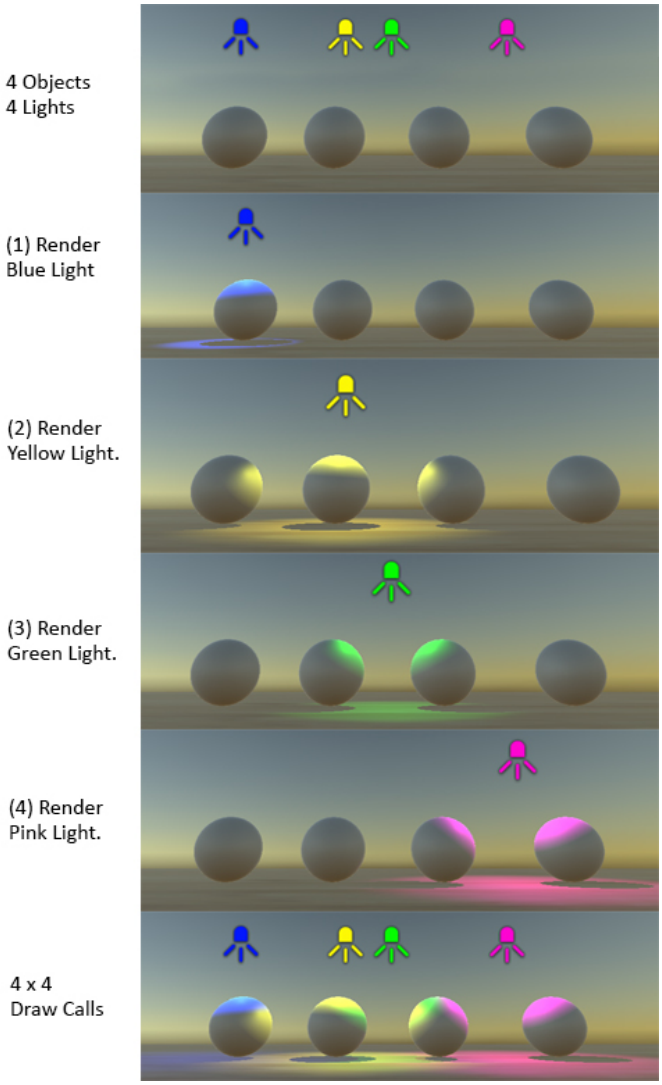

Figure 2.5. A Demonstration of forward rendering where four objects have been rendered by four different lights once at time.

This is a familiar concept in any 3D compositing background which renders different passes for different lights then composites them to produce the final image. The number of draw calls increases with every single light in the scene, therefore, in the mobile application, the number of lights is minimal with one light or baked into textures. The forward rendering has two main passes: **base pass** and **additional pass** where the lighting mode is defined based on the number of lights in the scene.

The forward base pass can render one-directional light per pixel. If there are several directional lights in the scene, the brightest one will use the base pass. However, if the scene has no directional light, these calculations cannot be done. In addition, it can render Spherical Harmonics (SH) lights such as light probes, global illumination, and sky ambient. The forward additional pass also can render one directional light per pixel that affects the object. for instance, if the scene has one directional light and other point or spotlight, the base pass will perform the calculation for the directional light, and the additional pass will be devoted to render the additional lights influencing the object [18, 19].

2.3.2 Legacy Deferred Lighting

The problem with adding more and more lights to the scene is performance cost. A light evaluation for the surfaces must be performed for every light added. Deferred rendering is one solution to this obstacle. In deferred rendering instead of storing the fragment color for each pixel of the closest visible object, other data for each pixel is stored such as position, normal, material.

Every light point in the scene has an upper limit for the light range which forms a sphere that can affect a volume in space, see Figure 2.6. Therefore, the surfaces found inside the sphere are affected by this light. Each light affects a small number of pixels on the screen which means a massive number of lights with a limited radius can be evaluated each time. The sphere drawing informs the GPU which pixels on the screen are covered by the light and should be evaluated. While there are variant shapes that can be drawn such as circle, aligned rectangle, the purpose of geometry is to test only the limited set of pixels that are potentially inside the light range.

Deferred lighting has three stages as demonstrated in (Figure. 2.7): (1) The first stage is to render the scene into the "Geometry Buffer" (G-Buffer) which contains the Depth (Z-Buffer), Specular power, and Normal for each visible pixel. (2) the second stage is to find the affected pixels for each light source in the scene, and read

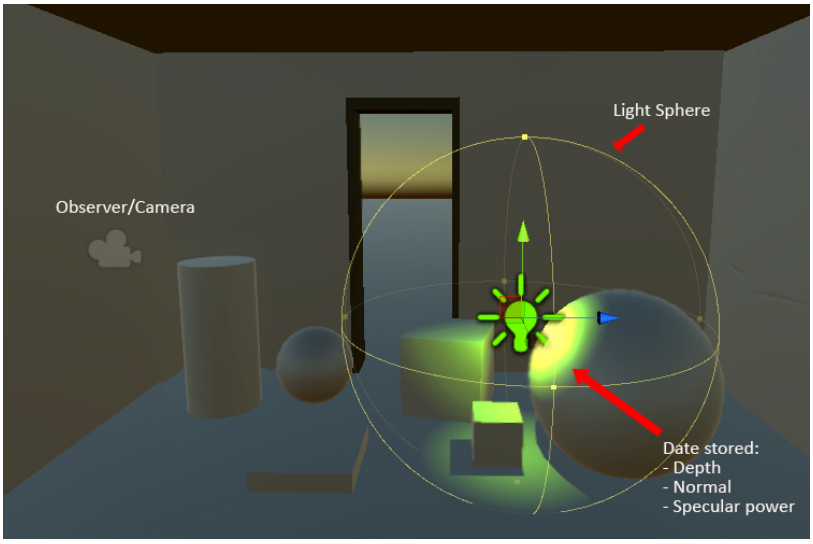

Figure 2.6. The light range in shape of sphere.

the corresponding G-buffer data, then calculate the light value, and finally store it in the "Light-Accumulation Buffer". (3) the third stage is to render the scene for the second time for each visible pixel while combining the accumulated light values with the mesh color which is defined in the material of the object to produce the final color, then add any ambient or incident light [19].

The deferred rendering was invented to overcome the performance latency in the forward rendering. This type of rendering would defer the shading of the scene to the last moment available. However, it has some drawbacks, such as difficulty rendering transparent objects. Also, it is not flexible to have all the information passed beforehand onto the last stage while the renderer has been developed [17]. The Legacy deferred lighting is different than the deferred shading discussed below.

2.3.3 Deferred Shading

Deferred shading has two stages: (1) the object in the scene is rendered into the G-buffer while storing the object depth, diffuse color, normal in the world space,

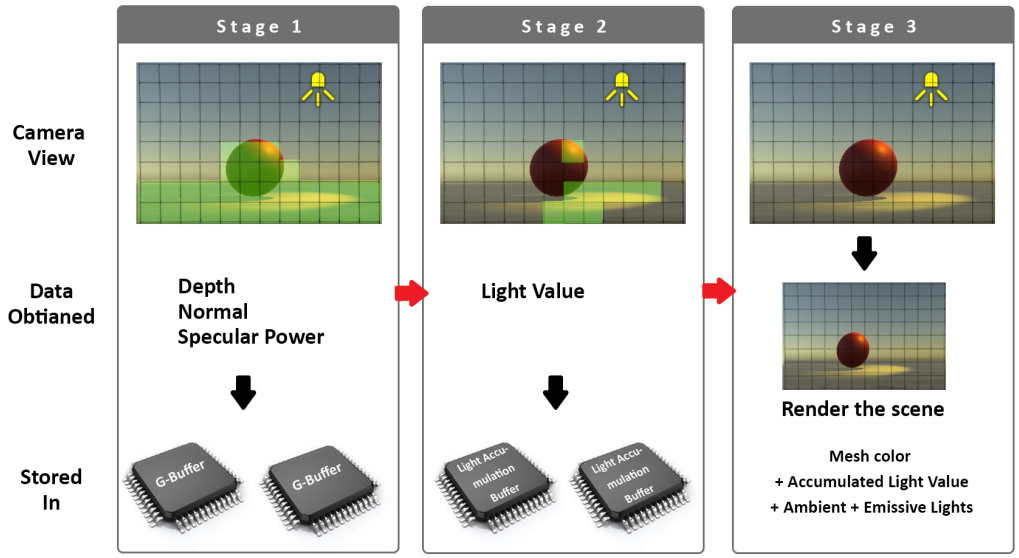

Figure 2.7. A visual representation of the Legacy Deferred Lighting three stages where two meshes are available at the scene: sphere and plane floor and how they affected by one light source.

specular color, smoothness, and emission, see Figure 2.8. (2) Find the affected pixels for each light source, then read the corresponding data from the G-buffer to calculate the lighting value, and store them in the light accumulation buffer. Finally, read the G-buffer to find the mesh color based on the data stored in the first stage, then add the accumulated light value to produce the final color of each pixel.

The basic difference between the deferred lighting and deferred shading is that the latter does not require rendering the scene for the second time as long it is done in the first stage where the color and other properties were not stored in the G-buffer. Deferred shading required a graphics card with multiple render targets, and the hardware must support shader model 3 or later and support for depth-render texture.

These are the different types of rendering techniques or render paths that can be used to render lighting in scenes.

Figure 2.8. An example of different passes or data that is collected in deferred shading and deferred lighting.

2.3.4 Ray Tracing

The ray-tracing algorithm was introduced by Whitted in 1979 [20]. For each pixel, shoot a "visibility ray" based on the camera direction and center of projection, see Figure 2.9. This ray traverses the scene until it hits an object surface, which can be represented using equations of a non-linear system. Thus, instead of tracing the ray from the light source, it is traced backward from the viewer. The light source, material properties, and BRDF are the elements that evaluate the lighting at the hit point. A ray is shot from a point toward the light source direction to determine whether the point is located in the shadow or in the light. In order to simulate the light transport, i.e., reflection, refraction, and inter-reflection, additional rays are shot based on the BRDF. The procedure will stop when a threshold or a stable value is reached.

The simulation of the physically-based realistic lighting represented in shadows, inter-reflections, refractions, etc. is the main advantage of ray tracing. However, accelerating the algorithm with the current hardware is challenging due to the update of data structures in a fully deformed geometry. Furthermore, traversal of non-coherent secondary rays in the scene cause misses in the cache memory and high-cost performance [21].

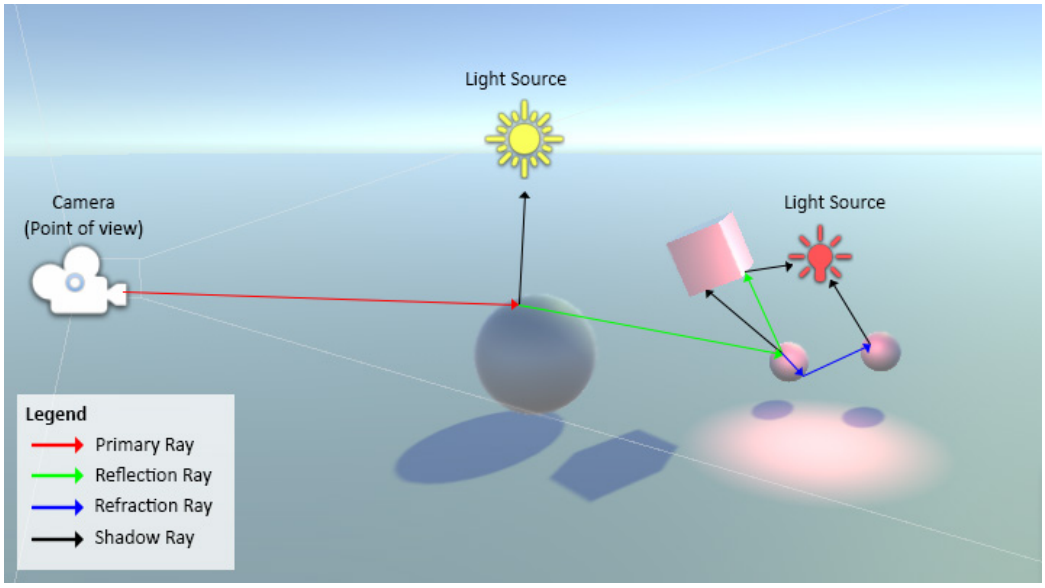

Figure 2.9. Ray tracing representation where a primary ray is shot for each pixel in the image from the camera view. When the ray hits a point on a primitive, the lighting is evaluated by shooting more rays for each effect: reflection, refraction, inter-reflection, shadow, and light source.

2.3.5 Path Tracing

The introduction of the path tracing algorithm and the rendering equation as numerical solution occurred in 1986 by Kajiya [14]. Similar to the ray tracing, rays are used for reflected light evaluation. In particular, A Monte Carlo method can be used to evaluate a reflection on any type of material. The incoming radiance is evaluated when reflected rays are randomly shot in several directions from the source. The following equation shows how to calculate the reflected radiance:

$$L_r(x,\vec{w}) \approx \frac{1}{N}\sum_{k=1}^{N}\frac{L_i, k(x,\vec{w'_k}) f_r(x,\vec{w'_K},\vec{w})|\vec{n}.\vec{w'_K}|dw'}{p(\vec{w'_K})} \tag{2.4}$$

Where L_r represents the reflected radiance, and the number of samples indicated as N, L_i, k is the k-th sample of the incoming radiance with ray in direction $\vec{w'_K}$. While $p(\vec{w'_K})$ is the probability of sampling a ray from the radiance contribution.

The algorithm calculates an impartial solution of the rendering equation while converging to the physical correct outcome. The high cost of performance is the main disadvantage of the path tracing algorithm. The increase of taken samples N decreases the error of the Monte Carlo estimator at a rate of $O(\sqrt{N})$. Also, the user can easily observe the noise caused by high variance errors. More improved sampling techniques led to faster convergence of the algorithm. Monte Carlo estimator can converge faster when the samples are distributed through $p(\vec{w'})$ that is round to the numerator in equation 2.4 which is one of variance reduction technique. The key feature of variance reduction is to transform the original integrand to a more constant function that is easier to integrate [22,23]. The variance reduced faster when the variables are sampled from a probability distribution with a similar shape to the function that is about to be integrated.

A more developed algorithm of the path tracing was proposed by Lafortune and Willems and is known as Bi-directional path tracing [24]. The procreation from the light source (forward) and from the camera (backward) are considered equally in the efficient version. Tracing rays is used to sample the paths from the camera and the light sources randomly. The hit rays and geometry intersections are stored while the shadow rays connect the light and camera paths, see Figure 2.10.

Path-tracing algorithm has been developed over the years. A Functional sampling and reconstruction to compute values was the focal point of several research [25–29]. Regulated path space to compute complex light paths is another presented approach [30]. When the light path is regulated, a biased sampling is achievable even in delta distributions [31].

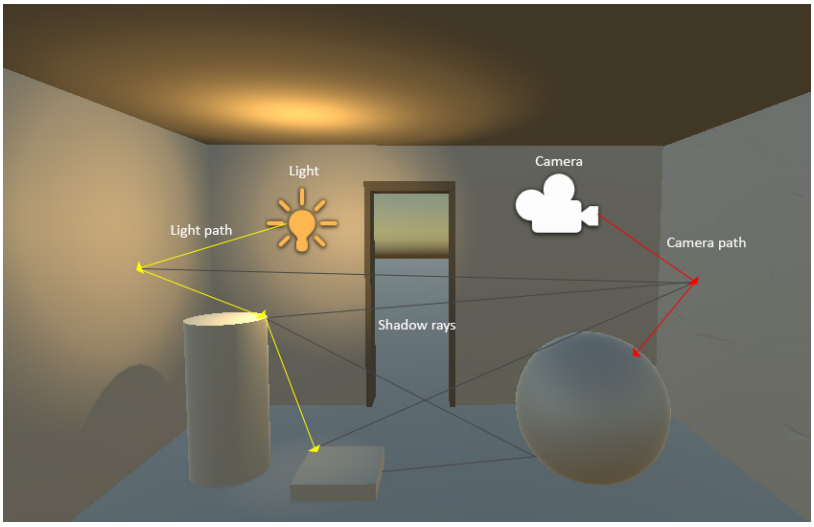

Figure 2.10. A representation of the Bi-directional path tracing algorithm

2.4 Differential Rendering and Compositing

The number of rendering passes that are required in the traditional differential rendering is two: one involved the local model of the real environment, and the other is merging both real and virtual objects. Doing the work twice without any change in many regions depicting any visual effect can be a questionable approach. The use of a single pass is a more efficient approach where the changes in lighting created by the virtual objects are simulated directly.

According to real-world lighting conditions, we can compute the common illumination between any kind of objects using a real scene model, virtual scene, and incident light. The light traveling directly from the source to an object and is reflected toward an observer is known as direct illumination. The light traveling from the source to an object and reflected toward another object is known as indirect illumination. Simulation of full global illumination can involve many light bounces between the objects before it eventually reaches the observer. The combinations of the interactions among objects could involve any of these four possibilities: from real objects to other real objects or to a virtual object, and from virtual objects to other

similar virtual objects or different real objects. The composition of real and virtual is based on differential rendering which contributes to visual realism.

Common illumination even with a precise photometric registration would not be perfect because it is not possible to fully interpret all light interactions in the scene. However, it would be efficient to preserve the subtle illumination effects which are present naturally in the real scene final image. The process of allowing the real-world illumination to be preserved is referred to as differential rendering. Fournier et al. [32] introduced the concept, then Debevec [33] developed the formula of differential rendering as follows.

Given the scene geometry, scene material, camera parameters, and light source, we can compute a light simulation L_R that corresponds to the original scene without virtual objects. A second light simulation $L_{(R+V)}$ can be computed after inserting the virtual objects. Any pixels depicting virtual objects can be replaced by $L_{(R+V)}$. For all pixels depicting real objects the difference $L_{(R+V)} - L_R$ shows the changes that happened to the real objects after adding the virtual objects. Then, the difference can be added as a correction term to the camera image L_c, See Figure 2.11.

Therefore, for pixels with virtual objects $L_{final} = L_{(R+V)}$, and for pixels with real objects $L_{final} = L_C + L_{(R+V)} - L_R$ can be interpreted as an error term to simulate the result $L_{(R+V)}$ for correction of any inaccuracies in the modeling L_R of the original scene L_C. The pixels are brightened if the virtual objects indirectly illuminate them $L_{(R+V)} - L_{(R)}(positive)$. However, the pixels are darkened if the virtual objects cast a shadow $L_{(R+V)} - L_{(R)}(negative)$.

This rendering could be more challenging if the scene modifications provide re-lighting which changes the light sources and how that will affect the whole scene and not only the objects. Mainly, the idea is to remove the light source and cause the shadow to disappear from the scene. On the other hand, adding a new virtual light source would be applicable where the light can be linearly combined. Therefore, many methods were enhanced and developed to accommodate real-time global illumination.

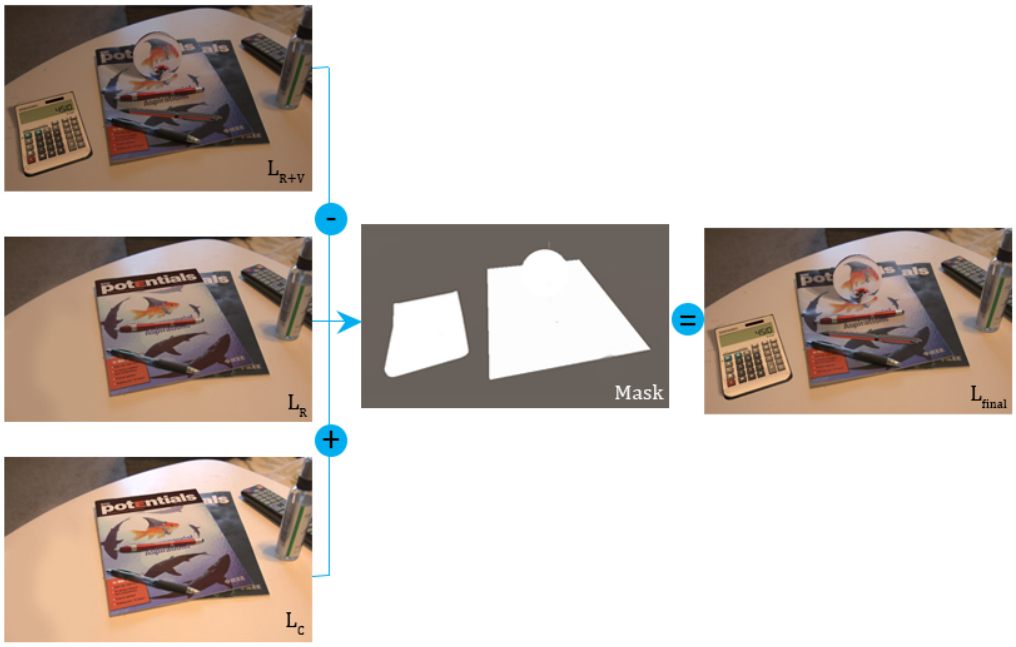

Figure 2.11. Differential rendering combines the light of the new virtual objects added to the scene against the physical scene representation. [3, 4, 6, 34]

Grosch et al. [35] modified photon mapping described in Jensen et al. [36] by using a differential photon mapping render in one pass for both real and virtual objects interaction. Every pixel of the environment map is representing a parallel light source. Thus, before the photons are shot towards the virtual object, First, they are uniformly distributed on a disk that has a radius perpendicular to the light direction. Then, If the virtual object was hit by a photon, the next intersection point is calculated using the real geometry and a negative flux is assign to that photon. Otherwise, if the virtual object does not intersect by a photon, it can be ignored, due to unchanged status to the light path.

Also, Grosch et al. [37] suggested a global illumination technique for indoor scenes in real-time using diffuse materials by light probes. The representation of near-field reflected light in the room is updated by using the direct light from outside and

a dynamic irradiance volume. The direct lighting also used sampling and shadow mapping. Knecht et al. [38] presented methods that combined instant radiosity utilizing differential rendering that only needs one rendering pass for achieving real-time performance in both diffuse and specular objects. Their method was extended for handling the reflective and refractive objects, and caustic effects by assuming the real object's geometry is static and given.

Kan et al. [39, 40] developed a method for interactive global illumination using photon mapping that allows caustic and reflective or refractive materials. Also, they developed a one-pass differential rendering method in real-time by utilizing irradiance caching. This irradiance separated real and virtual objects by analyzing both diverse ray types and intersection situations, which could be helpful in the computing process of differential irradiance. It is known that the real and differential irradiance are stored in the irradiance cache record which then can be utilized on the GPU for irradiance cache splatting. This method has some limitations regarding diffuse materials which required precomputation stages. However, the results were reasonable for multiple bounce global illumination [41].

Lensing et al. [42] solved the pre-computation stage of a one-bounce diffuse indirect lightning using reflective shadow mapping. Also, to overcome the errors of the depth image, the method used was pure image-based with some guided filtering. The development of differential rendering extended to mobile devices. Rohmer et al. [43, 44] reduced the computational cost for each light using tile-based rendering, in addition to frustum culling techniques tailored for AR systems and applications. Monroy et al. [45] presented a similar system which works in a dynamic environment with the ability to scan the real scene and then projected onto a two-dimensional environment map that contains RGB+Depth data.

2.5 Light Polarization

Any form of illumination whether natural (sunlight) or artificial (bulbs, lamps, flares, fluorescent, fireworks, LEDs, fire, etc) produces light waves. The electric field vectors of the light waves vibrate in every plane that is perpendicular to the propagation direction. The light is called linearly polarized (plane) if the electric field vectors are limited to a single plane with a filtered beam using specialized materials. The single plane receives all the waves vibrating which is referred to as plane-polarized (plane-parallel).

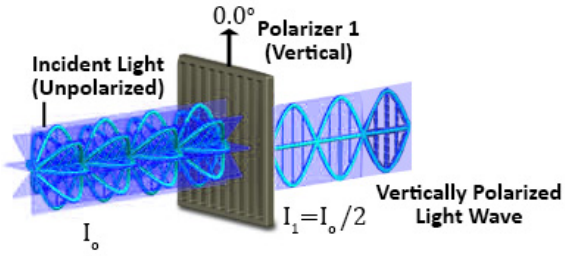

Figure 2.12. An illustration for the basic concept of polarized light [3].

The human eye cannot distinguish between (1) arbitrarily oriented and polarized light, and (2) a plane-polarized light; they can only be distinguished through the effect of color or intensity, such as wearing polarized sunglasses to reduce the glare. Figure 2.12 provides an illustration of the polarized light basic concept where an incident beam of light is polarized using one linear sheet filter. The non-polarized incident light has electric field vectors which are in all directions (360 degrees) as sinusoidal waves vibrating even though there are only six waves drawn in the figure at 60-degree intervals. In fact, the electric field vectors of the incident light before polarization are vibrating perpendicular to the propagation direction with an equal distribution in all the planes [46].

A standard physical process including absorption, reflection, refraction, scattering (diffraction), and birefringence (double refraction) can produce a polarized light by deviating the light beams. The reflected light from a flat insulating surface is often partially polarized with electric vectors that vibrate in a plane parallel to the insulating surface [46]. The electromagnetic waves can be explained using the consequence of Maxwell's equations and material equations. As it can be seen that the plane vector wave is represented with the following equation [47, 48].

$$\vec{E} = \vec{E_0} e^{i(\vec{k}.\vec{x} - wt)} \tag{2.5}$$

Where:

- $\vec{E}$ Real electric field vector.
- $\vec{k}$ Spatially and temporally constant wave vector which is normal to surface of constant phase.
- $|\vec{k}|$ Wave number.
- $\vec{x}$ Spatial location.
- w Angular frequency ($2\pi \times frequency$).
- t Time.
- $\vec{E_0}$ A complex vector in general independent of time and space. It decreases in the amplitude of an oscillation if $\vec{k}$ is complex.

The polarization can be described by a spatial and temporal constant vector $\vec{E_0}$ positioned on a plane that is perpendicular to the direction of propagation $\vec{k}$. The 2D basis representation of $\vec{E_0}$ with two unit vectors $\vec{e_1}$ and $\vec{e_2}$ where both are perpendicular to $\vec{k}$

$$\vec{E_0} = E_1\vec{e_1} + E_2\vec{e_2} \tag{2.6}$$

Where E_1, E_2 are arbitrary complex scalars.

A plane-wave solution that has a restricted or reduced amplitude of vibrations with given $w, \vec{k}$ has four degrees of freedom i.e. two complex scalars. Therefore, the polarization can be represented as:

$$\vec{E}(t) = \vec{E}_0 e^{i(\vec{k}.\vec{x}-wt)} \tag{2.7}$$

Any additional property is known as polarization. There are several approaches to represent these four quantities, for instance, if the phases of E_1 and E_2 are identical, then $\vec{E}$ oscillates in a fixed plane.

$$\vec{E}_0 = E_1 e^{i\delta_1} \vec{e_x} + E_2 e^{i\delta_2} \vec{e_y} \tag{2.8}$$

Where the wave vector in z-direction, $\vec{e_x}, \vec{e_y}$ are unit vectors in x, y directions. While E_1, E_2 are real amplitudes, and $\delta_{1,2}$ are real phases. Although two complex scalars would not be considered very informative description, it could provide a time evolution of $\vec{E}$ at given $\vec{x}$ which can be described by the polarization ellipse that is described by axes a, b orientation ϕ as shown in Figure 2.13.

Several applications depend on crossed polarizing filters to examine birefringence, for instance, check for double refracting specimens using polarized optical microscopy. The transmission axes for two crossed polarizing filters are oriented perpendicular to each other where the light transiting through the first polarizing filter is totally absorbed by the second polarizing filter which is called analyzer. The filter quality of absorbing the light is the main factor that determines the amount of random light being absorbed in case polarizing filters are used as crossed pairs, referred to as the extinction factor of a polarizing filter. The orientation of the polarizing filters' axes determines the extinction factor i.e., the light ratio passing through the filter whether

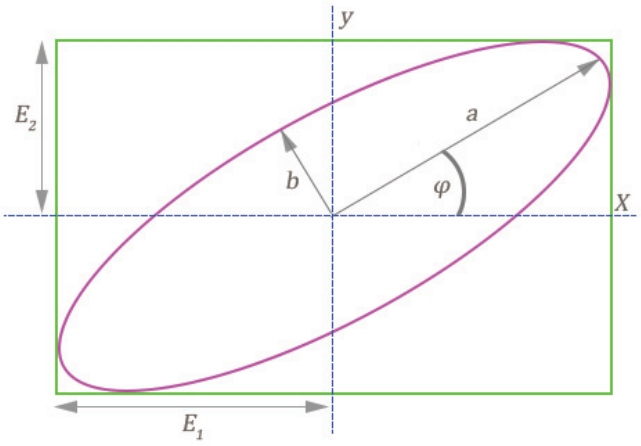

Figure 2.13. Polarization ellipse to clarify the equation variables and properties..

they are parallel or perpendicular to each other. The maximum level of extinction occurs when the pair of polarizing filters are perpendicular to each other, while the different levels of extinction are exhibited at other angles [49, 50]. Further discussion about the number of polarizing filters and the amount of light passing through them in our research is discussed in chapter 4.

2.6 Related Work of Developed Methods

2.6.1 Detection and Polarization of Incident Lights

When light hits the object surface directly from the source such as sunlight or artificial lights it is called direct illumination also known as the incident light, see Figure 1.1. The shadows in the final scene are influenced by direct illumination and are a key factor for increasing realism in Augmented, Mixed, and Virtual Reality (AR/MR/VR) applications. Therefore, several methods in computer vision and graphics had been developed over the years in order to extract information about the light source from the physical world then integrate the illumination data into the

virtual scene. Debevec [51] developed the Image-Based Lighting (IBL) method that uses multiple radiance maps to acquire the correct calculation of the incident light through the variance cut algorithm. Although the material properties of synthetic objects were calculated arbitrarily, the high cost of the sampling process was not conducive to real-time [52]. Sampling techniques and algorithms have been developed ever since where the incident lighting was spatially sampled to handle the variations over the center point [53].

A low number of clusters denoted as Virtual Area Lights (VALs) were obtained through texture atlas sampling of recorded direct radiance using K-means, then grouped in several clusters integrating direct illumination onto the virtual objects [54]. When the scene has multiple lights, it can be divided into cells to sample direct illumination by finding clusters of the light sources in each cell [55]. The surface reflectance property for several types of the light source was recovered using the inverse rendering method using an RGB-D sensor in indoor scene [56]. Light source position in a single image was used to estimate the surface reflectance property which required various coefficients that can raise the performance cost in real-time [57]. The radiance transfer method was utilized to factor the texture color into direct illumination and diffuse albedo color as material properties in the inverse rendering [58]. A Lambertian surface material integrating over the upper hemisphere was assumed to estimate the incident light source [59].

The Spherical Harmonic (SH) projection coefficients and functions were applied for light factorization techniques to find the dominant light direction and color [60]. Gruber et al. [61] projected the visibility results in per-pixel SH functions for deferred light estimation and rendering without instantly computing the lighting from a known incident light source. An analytical expression for irradiance environment maps under the Spherical Harmonic (SH) coefficients of lighting was developed for arbitrary lighting distribution on diffuse surfaces of the Bidirectional Reflection Distribution Function (BRDF), rather than relying on the graphics hardware for resources regardless of high-frequency shadows and specular components [62].

An image representation of the physical scene that includes geometry and radiance was used for direct illumination detection to render the final image of the AR scene [63,64]. Multiple light sources were sampled and extracted from a hemispherical view in each frame using variance minimized median cut algorithm. The spherical coordinates (θ, ϕ) of the light sources were computed then placed into the corresponding location of the ground truth on the sphere around the virtual object [65]. The combination of analytic illumination and stochastic ray-traced shadows were utilized for a ratio estimator of the incident lights [66]. The physical environment was captured to calculate the direct illumination using ray-tracing rendering [67].

2.6.2 Simulation of Reflected lights and Global illumination

A realistic rendering addresses the indirect illumination which is a collective effect of inter-reflections of lights between real and virtual surfaces in the final scene. Since Keller [68] introduced Instant Radiosity in 1997, the concept was developed to replace Virtual Point Lights (VPLs) for reflected lights approximation which is suitable with the modern hardware that doesn't require excessive pre-computations. Each pixel in a reflective shadow map was assumed to be a light source, then VPLs were created with the adaptive sampling for indirect illumination calculation. Nevertheless, generating shadow maps for each VPL demand a high-cost of computation due to a lack of visibility [69]. Instant radiosity combined with differential rendering can provide plausible and realistic mixed reality scenes while maintaining a frame rate of 30fps interactively [70]. The radiosity algorithm for diffuse lighting on the virtual object was used to reconstruct direct and indirect illumination for reflected light simulation in real-time [32]. The hierarchical radiosity for indirect illumination with a rough subdivision of the scene was employed to estimate the reflected light [71].

The photon map also was used for dense estimation of indirect illumination every time a diffuse surface was hit at ray-tracing rendering. The dense estimation had three approaches: histogram, nearest-neighbor, and kernel, the latter was preferred due to

reduced performance cost associated with high-quality outcome [72]. The calculation of multiple bounces of indirect illumination was possible through Monte Carlo integration to evaluate the irradiances in cache records by shooting recursive rays into arbitrary directions over the surface point [31]. A system with two Monte Carlo as path-tracer to sample the cosine hemisphere on real/virtual object for indirect illumination, while sampling a Phong lobe on virtual glossy objects only [73].

Scene geometry was reconstructed using an RGB-Depth sensor that supported fast-updating which enabled users to interact with reflected lights in a static environment [61]. A combination of RGB-Depth sensor and Monte Carlo sampling was utilized for computing the reflected lighting to a known diffuse surface [56].

Global Cub Map. An environment map that is reconstructed as a cube map is supported by many graphics cards for proper global reflections with high-resolution on virtual objects [71]. While each pixel on the unit sphere represented an individual direction, a cube map was used to evaluate the SH coefficient [74]. A low-resolution cube map from the virtual object position was rendered to simulate the indirect illumination as atlas textures of that object surrounding environment [75]. A diffuse/glossy cube map that addressed each material property was created as global environment maps in a parallel manner for each virtual object [76].

Local Sampling. A 2D texture mapping was enabled to store a frame buffer of a certain image of the real world while rendering the final scene [77]. A few HDR images or environment maps were used to capture the light representing a 2D texture or 4D surface light fields then project them onto a geometric model [78].

2.6.3 Shading Virtual Scene Surfaces

The estimation of light sources with the simulation of indirect illumination would not be useful without integrating this information into the virtual objects by supporting many surface shading (BRDF) and shadowing models. A combination of most

common models of virtual lighting in a mixed reality that includes traditional point and directional lights with light probes were used in a novel manner to capture and apply the lighting from the surrounding environment to the virtual scene surfaces [60].

The change in the radiance field is represented as *Delta Radiance Field* $L\Delta$ used to extract the difference in illumination that bounces in the augmented scene which adapts with global illumination and reduces computation overhead where the reflection operator represents the surface shading based on the incident light [65, 79, 80]. The associated information with the object geometric parameters such as vertices was stored and preserved in case the surface shading normals with each vertex of polygon due to the *varying* field continuous modification based on lighting and other dynamic change [81].

2.6.4 Physical Geometric Properties of Real-Scenes

Current devices such as smartphones contain a motion tracking module estimating device pose based on a cloud of points corresponding to the visual appearance of the spatial features of several objects in the physical environment [82]. A mapping module builds a 3D visual representation of the physical scene based on the stored depth map, feature points, and estimated device poses [83]. The 3D visual representation is received through a localization module after mapping to identify similarities between stored and observed features points [84]. The localization module performs a loop closure correction minimizing error for matching feature points while computing the localized pose [85].

Simultaneous localization and mapping (SLAM) methods are incorporated in modern devices that provide motion tracking based on the device platform [86]. SLAM features come in different forms such as concurrent odometry and mapping (COM), visual-inertial odometry (VIO), and six DOF camera pose tracking combined with/without inertial measurement unit (IMU) sensors to estimate the pose of the device relative to the world over time [86, 87]. Device camera calibration with the

virtual camera in the virtual scene [88] allows the developers to render the virtual objects into the real space from the correct perspective including occlusion, light estimation, shading properties [89]. Every feature point that is captured provides spatial information that could be used for both spatial and visual coherence. A full 3D reconstruction of the real world can be achieved using a depth map for feature points and planes which contain location, color, and other data that could build some of the real environment in the virtual space [90–93].

3 RESEARCH DESIGN AND METHODOLOGY

3.1 Introduction

A comprehensive overview of the system components is described in this chapter which provides an illumination model for visual coherence of augmented reality applications with dynamic environment where virtual objects appear as a part of the real world utilizing computer vision and graphics methods. For a more organized flow, the system's main components are discussed separately in the following chapters which work simultaneously in real-time. A visualized version of the overview (a flowchart) is shown in Figure 3.1 and broken down based on each method in Figures 3.2, 3.3, 3.4, , 3.5, 3.6, 3.7, 3.8, and 3.9. Some illumination challenges with augmented reality development and adoption are enclosed within this chapter while suggested optimization and mitigated solutions to overcome these challenges are presented in the next chapter. This chapter is a reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

3.2 Data Inputs (Camera Feed)

The system uses two cameras; (1) AR device main camera which is what the user see while holding or wearing the AR device, (2) 360° camera with live-feed feature that is instrumented on any AR device and can see the whole physical environment in order to capture the radiance map, See Figure 3.2.

3.3 Detection and Polarization of Incident Lights (Direct Illumination)

The physical lights reaching the 360° camera view is investigated through computer vision sampling methods to calculate the angle and direction of several light

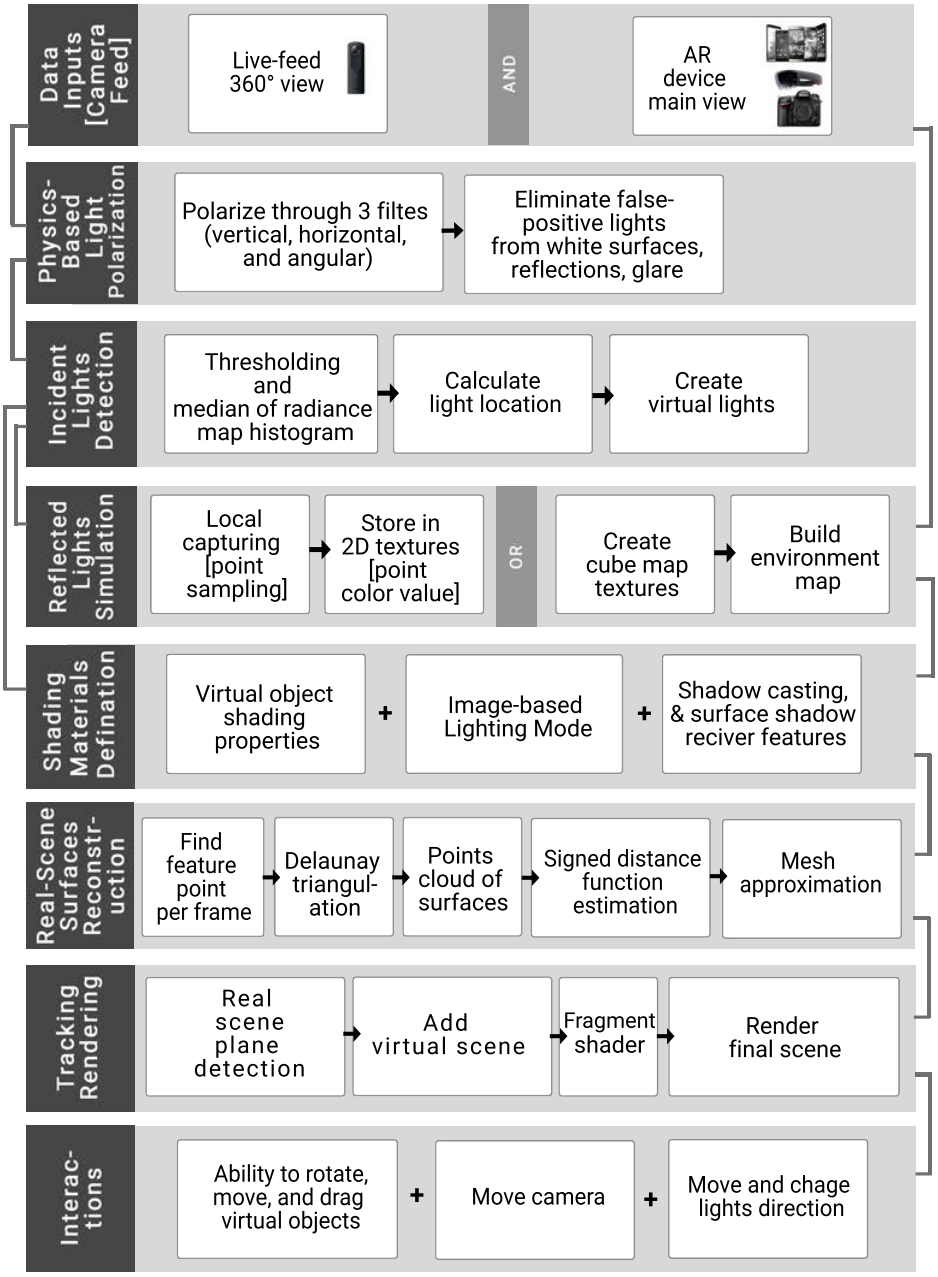

Figure 3.1. An overview of the entire system components: physics-based light polarization, incident light detection, reflected light simulation, shading materials definition, and real-scene geometric reconstruction followed by tracking, rendering, and interaction.

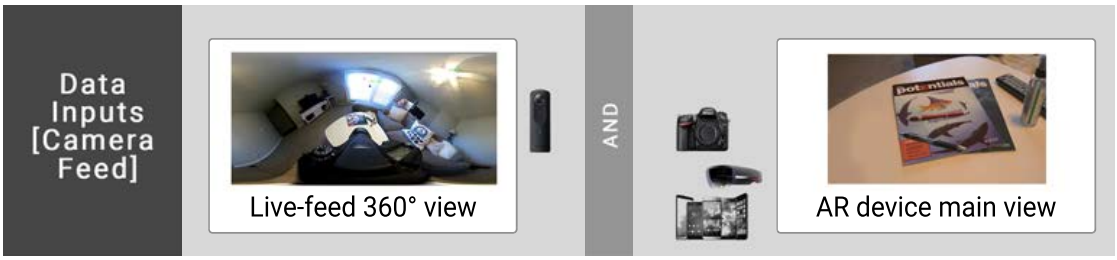

Figure 3.2. An example of both cameras feeds, while 360° camera read the whole physical environment map, the AR device main view track image-based marker and read the user current view for the final scene.

sources. The detection process is using a radiance map of the real-world environment that indicates the light sources which are updated automatically in real-time to reflect any changes in the lighting conditions, See Figure 3.3. The false-positive light conditions such as white surfaces, reflections, or glare that were a noise source to our light detection algorithm are reduced or absorbed completely after the use of polarization properties of the incident light. For the probabilities of absorbing the light passing through each filter, see Figure 3.4 and chapter 4 for more details.

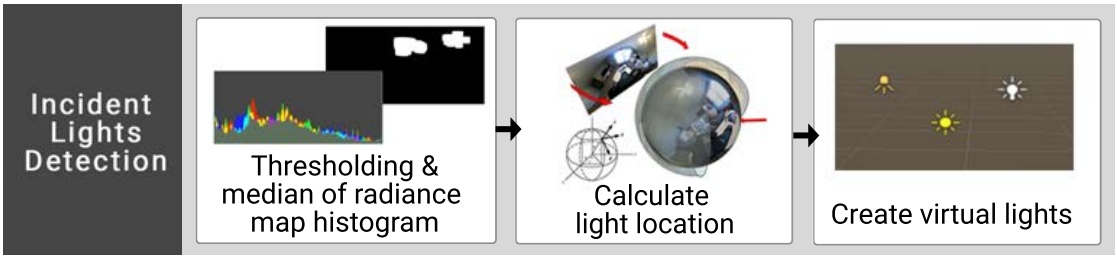

Figure 3.3. Steps of the incident lights detection method consist of thresholding based on the histogram median of the radiance map in order to calculate the light location and create a virtual light based on that direction in the virtual scene.

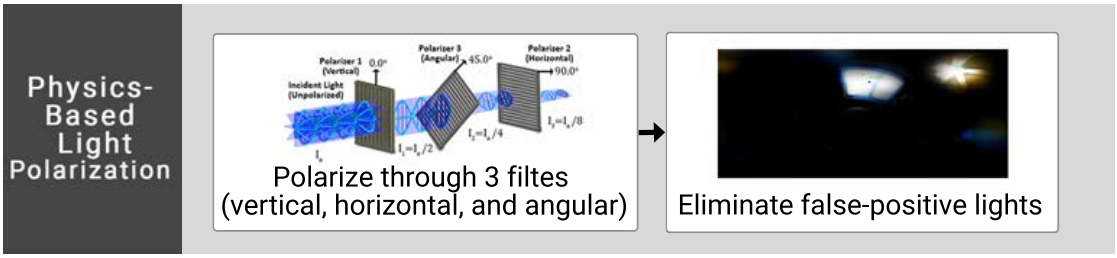

Figure 3.4. The 360° view is completely polarized for accurate detection of the incident lights direction while reduce/eliminate false-positive lights from reflections, white surfaces, and glares.

3.4 Simulation of Reflected Lights (Indirect Illumination)

The bouncing light between the objects and the surround is captured using two sub-methods, see Figure 3.5. The resulting texture is rendered into the virtual object based on the material properties. First, *Global Cube Map,* the panoramic HDR of the 360° camera is used to create the 6 faces as a 2D texture to construct a cube map for the image-based lighting mode which can be modified while defining the shading properties for each virtual object. Second, *Local Sampling,* the main camera of the AR view device is employed to sample the region below and surrounding each virtual object. The resulted texture is then used in IBL mode where the material property of each object can be reflected on the objects, chapter 5.

3.5 Definition of Shading Properties

The virtual objects materials and features are defined in specific shading programs that are created in order to meet the requirements of reading from live-feed cameras. These characteristics are defined based on the virtual object properties which include image-based lighting (IBL) mode to reflect the lights in the previous components.

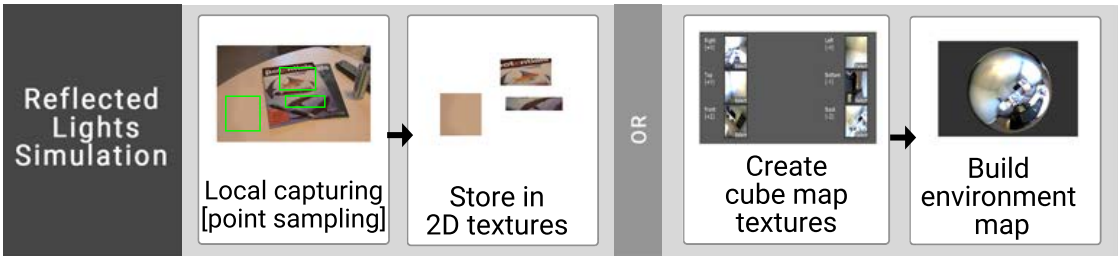

Figure 3.5. The reflected lights are simulated either by local capturing/sampling the lights around the virtual object or by creating a cube map for the entire environment map.

The shading features include normal mapping, transform the normal map from object-space to the world normal, diffuse lighting/reflection, specular lighting/reflection, and more, see Figure 3.6 and chapter 6 for detailed information.

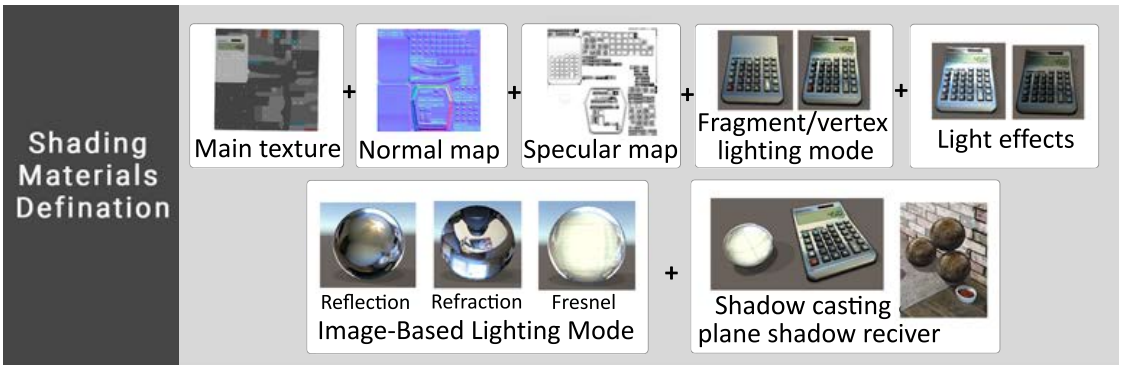

Figure 3.6. Define shading properties provide the virtual object with more realistic outcomes, each property is then affected with direct and indirect illumination methods above.

3.6 Physical Geometric Properties of Real-Scenes

The physical geometrical information of the real-world environment is a supporting component for tracking which allows locating the virtual object without markers and increase the realism of virtual-to-real objects interaction. The depth data from the real world provides transforming independent acquisitions, or point clouds, into a single-surface triangulated mesh and can be fulfilled with different algorithmic approaches. Several tools provide solutions to reconstruct the shape of an object, ranging from volumetric (Marching Cube) to implicit surfaces (Screened Poisson). Our system mainly investigates the geometric properties of real-scene including plane detection, 3D surfaces reconstruction and simple meshing which are incorporated with the virtual scene for more realistic depth interactions between the real and virtual objects, see chapter 7.

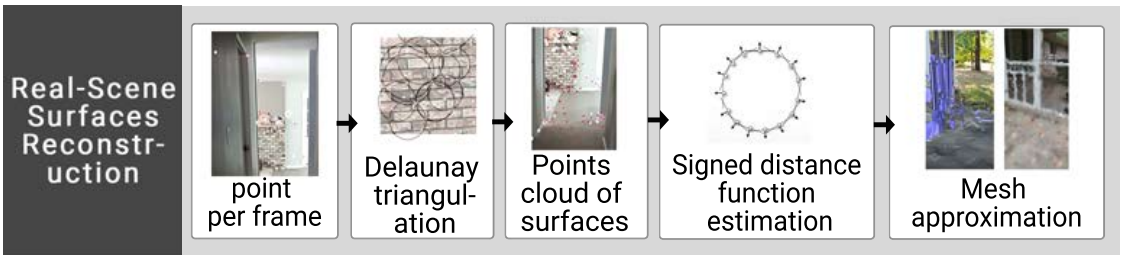

Figure 3.7. The physical world include ample amount of information that can increase realism in augmented reality, the depth data in form of points cloud is used for plane detection, surface reconstruction and mesh creation.

3.7 System Rendering

Differential rendering allows virtual objects to be augmented into the physical scene where light source is part of the input. The final image in the 3D engine uses the deferred shading rendering path where the scene is rendered into a G-Buffer stor-

ing position, normal and material coefficients for each pixel. In this case, illumination is decoupled from the object geometry into direct (incident) and indirect (reflected) lights by performing lighting calculations in image space. The point geometries covering the light-influenced regions are rendered, while the radiance is computed in fragment/vertex programs and accumulated by additive blending to get the final illumination including shadows [54]. The support of modern GPUs' computing capabilities provides two passes; one for the main light and another for any additional lights. The augmented reality package enables the 3D engine's multi-platform API which implements several subsystems and functionality including depth, ray-cast, and point cloud. Ray-cast provides methods and properties querying portions of the physical scene by casting a ray from a screen point against selected track-able planes and feature points, see Figure 3.8.

3.8 Tracking Apparatus

For mobile implementation, motion tracking supported by ARCore is utilized to understand where the device is relative to the physical world surrounding it through concurrent odometry and mapping (COM) process. For PC implementation, The positional device tracker supported by the Vuforia AR engine is used for a robust 6 degree-of-freedom (DOF) target tracking. furthermore, several modifications are added through separate scripts to support the lighting conditions in a dynamic environment, see Figure 3.9.

3.9 Illumination Challenges with AR Systems Development and Adoption

Designing and implementing a reliable AR system means facing technical challenges and constraints that impact the overall usability of the system and, in turn, how well the installation actually aids the realism of virtual-real object interactions. We review three classes of challenges that we encountered when designing and imple-

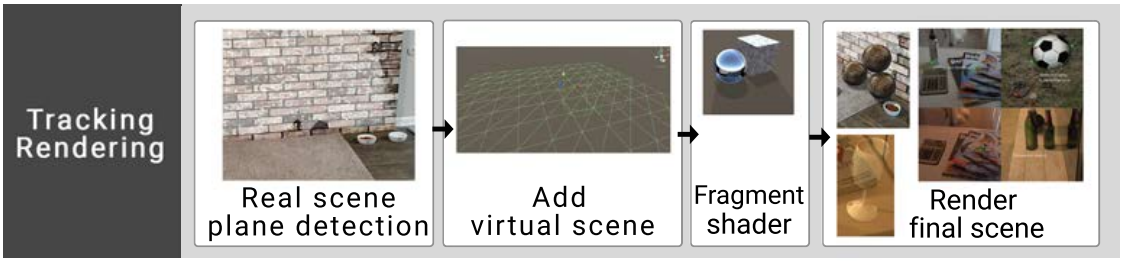

Figure 3.8. The plane detection method and the image-based marker support tracking of where the virtual will be rendered in the final scene with the correct shading properties and lighting conditions.

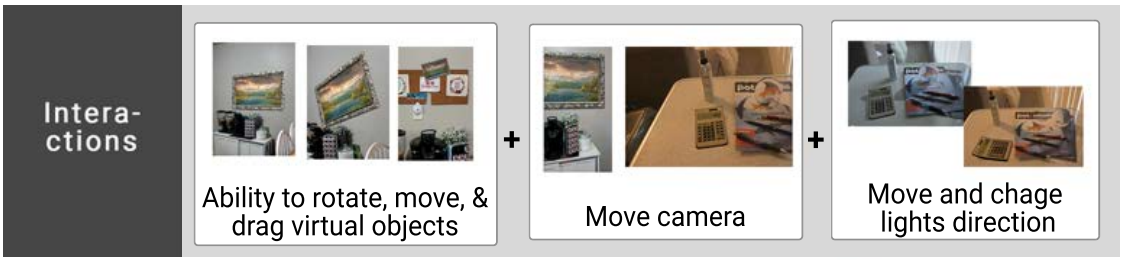

Figure 3.9. A dynamic environment can only achieved with the ability to change and move objects, lights, and camera location while every fragment and property of the virtual scene will update accordingly to fit right with the physical world.

menting our system, as they can help to optimize current and future systems and to provide insights on how to reduce similar problems and constraints.

3.9.1 False-Positive Lights & Polarization in Computer Vision

Although the incident light detection algorithm presented in section 4.2.1 is able to detect the physical light sources in the real scene, false-positive lights were also

detected from reflection, white surfaces and glares. In order to reduce/eliminate the noise and the false information, we utilized light polarization techniques as tools of optimization for this challenge. The false-positive lights detected from reflections, white surfaces, and glares are reduced or eliminated completely using physics-based light polarization. Photography inspired the use of light polarization-based filtering [3], see Figure 3.10.

Review. Physics-based computer vision became popular after Horn introduced the optical models of reflection and imaging in 1975 [94]. Polarized light alters the human perception of intensity and color; polarization-based vision information is modified based on intensity and shading. The polarization phase-based method according to intrinsic electrical conductivity was used for classifying materials theoretical development and application. An accurate and robust metal detection scheme was shown in the results under different weather conditions [95].

A shape construction method also utilized the intensity of polarized light reflected from the objects which had different refraction index based on the Fresnel theory [96]. Also, a shape reconstruction technique that used shading and polarization with one constraint for each; a pair of light directions for shading, and a pair of polarizing filter angles for polarization. The outcome recovered the shape, refractive indexes, and light directions [97]. Dehazing and denoising scheme for unclear images take the advantage of polarizing images that collected at different days to verify noise reduction algorithm and details optimization [98]. Optimized data analysis is advanced to capture the image under polarized conditions for single to noise ratio maximization. The work captured images that illustrated a range of reflectance properties such as inter-reflections, composited specular/diffuse reflection, and conductance of the surface [99].

Figure 3.10. An illustration of how the circular polarizing filter/Linear (CPL) camera filter used to reduce reflections and glare [3]

3.9.2 The Use of Live-feed Video and Equirectangular Panoramas

In order to accurately detect the direction of the physical light source, we need to consider the distortion and complete view of 360° camera and how that effect the system according to the main AR device. We assumed that the 360° camera is instrumented on the AR device (smart phone, head-mounted display, DSLR camera, etc) for an efficient camera calibration. However, an extended calculation is implemented to overcome this challenge and optimize the detection of the direct illumination location that is explained in details in section 4.2.3. The radiance map is basically an equirectangular image from a 360° camera view. A normalization of the image coordinates is performed to consider the live-feed camera view (front, sides, back) faces.

Review. A high dynamic range (HDR) panoramic images captured by 360° cameras for light detection and estimation algorithms is not a novel approach in computer vision and graphics research. The original utilization introduced a mirror sphere in the scene to capture almost 360° images [100,101], but with the current availability of

360° camera with a live-feed feature, it becomes practical to instrument it on any AR devices [1,3,4]. While others drive illumination estimation and prediction algorithms from a large data-set of spherical panorama through a light classifier [102, 103]. The reconstruction of an environment map through a 360° manual rotational motion on a mobile device was used to capture consistent indirect illumination usually [104]. The virtual objects were rendered using IBL while streaming Low Dynamic Range (LDR) 360° panoramic video as an input to build a radiance map [105]. For high-quality lighting, a verse tone mapping was demonstrated that converts LDR to HDR input stream [106].

Also, virtual objects were scanned inside a 360° video to relight environment geometry through estimating lights and materials for each frame where the data inputs are albedo color and lighting shading [107]. A conventional LDR 360° panoramic video was used in an immersive system with a Head-Mounted Display (HMDs) as a lighting source that illuminates the 3D virtual objects seamlessly into the live-feed video [108]. The scene radiance was measured relative to the first frame then compensated for the error accumulation over time when closing the loop of 360° video by providing a stable color adjustment in consecutive frames [64]. The real-world lighting was captured directly from a 360° camera which shows the difference of light estimating methods and the use of environment map through a panoramic video [109].

3.9.3 Sampling of Spherical Light Fields

The process of sampling a spherical view can be challenging in identifying the light field of each light source separately. When a single light source is warped at both end on the view in can be read as two or more light sources which can result in fault information about the incident light affecting the entire direct illumination model. Our system encounters light sampling challenge that is explained in detail in section 4.2.4 where a single incident light was identified as multiple light sources.

Review. A sample representation of light fields allowed both inward and outward-looking views efcient creation and display. The light fields were created using a large array of rendered and digitized images. A video camera mounted on a computer-controlled gantry was required for the digitized images. New views can be constructed in real-time once a light field has been created through extracting slices in the appropriate directions. A high sampling rate was achieved using a compression system that was able to compress the light elds with very little loss of delity [110]. An "object-space" algorithm was embedded into the polygonal rendering system supporting the primitive functionality such as the viewing and smooth shading, and allowing simple-to-implement hybrid rendering. While less complex regions are rendered from simple polygonal models, more complex regions are rendered from sampled spherical light fields [111].

A sparse set of discrete light field samples on the surface of the camera sphere around a virtual object that includes per-pixel depth information is used to develop light field parameterization and rendering algorithms [112]. A light field rendering technique that performed per-pixel depth correction of rays also was presented in Todt's work [113] that reconstructs a high-quality light field. A combination of RGB and depth values were stored in a parabolic 2D texture for every light field sample that was acquired at discrete positions in a uniform spherical setup. A lightweight system with real-time feedback that captures hundreds of light fields through sampling was provided and advanced in a rendering algorithm that was tailored to unstructured the captured dense data [114]. These methods and algorithms were used for a full light field reconstruction which is similar to the requirement of our system but that only captures the location of the light source as a spatial data point in image-space then transfers it into object-space.

4 DETECTION AND POLARIZATION OF INCIDENT LIGHTS

4.1 Introduction

This chapter investigates the visual coherence problem in augmented reality by developing a system that provides real-time dynamic illumination for interactive augmented reality based on the virtual objects appearance in association with the real objects and other criteria. The proposed algorithm **detects the incident light** (direct illumination) in the physical scene through a live-feed 360° camera that is instrumented on any AR device to capture the entire environment map using computer vision techniques. The **physics-based polarization** properties of light are utilized to reduce any false-positive lights from white surfaces, reflections, and glare in the detection of incident light resulting in fewer errors based on the lighting conditions. This chapter is reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

4.2 Method and Implementation

4.2.1 Detection of Incident Lights

The live-feed from the panoramic 360° camera produces a continuous radiance map that can be sampled efficiently. For a dynamic environment map that runs in real-time, the luminance pixels are captured using a thresholding approach based on topological structural analysis of 2D images. The camera feed is converted from 2D texture to object matrix using parallel computations. An automatic threshold value is given, the pixel is considered a point of the light source if its intensity is above the threshold. A mask is produced that only depicts the area of light sources. The

radiance map image is converted to a gray-scale to ensure connectivity. In order to determine the threshold automatically, two approaches are considered:

- Calculate the median of the radiance map histogram because the temporal and spatial variations of the real-time environment change drastically which will affect the overall threshold value.
- The polarized light allows only a small amount of incident light to pass through. Thus, the range of threshold is expanded significantly to a static binary value which performs faster in real-time.

In order to reduce the noise among the sampled points, Gaussian blur is applied before thresholding. An erosion morphological transformations were applied with a rectangular kernel in small size to discard the pixels near boundaries of the mask, followed by a dilation transformation with the triple-sized kernel to that used in the erosion. The pixel's luminance indicates the incident light in the radiance map which was extracted as a series of regions based on their area size to define the main light first, then the second, third light and so on. Identifying the color of light provides more realistic illumination information which was computed based on mean color for collective pixels in each regional area. The resulting color was more accurate when the polarizing filters are used where the incident lights are the main concerning point in the radiance map. The error metric for the pixel of basic color Pb and the polarized one Pp is:

$$\sum_{Pixels} \left| \Big(R_b, G_b, B_b \Big) - \Big(R_p, G_p, B_p \Big) \right| \tag{4.1}$$

The intensity of the pixels also is calculated relative to the entire radiance map which is used to determine the soft and hard shadow strength based on the light type.

The centroid of the sampled regions is converted from the screen coordinates (x, y) to the spherical coordinates (θ, ϕ) using the inverse spherical projection after

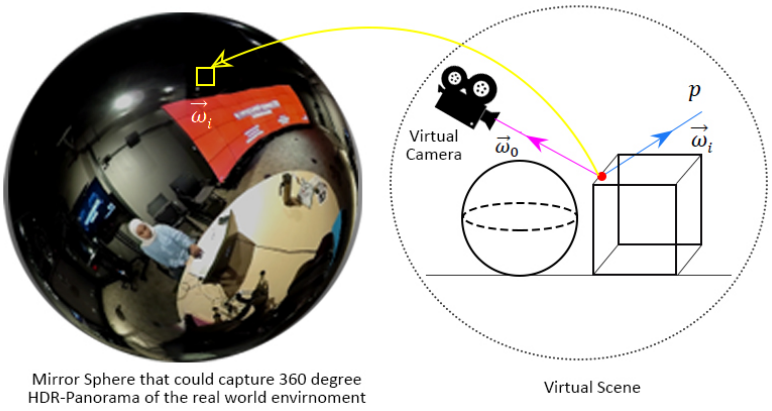

Figure 4.1. Allocation in HDR environment map and spherical projection concept [6]

normalization as shown in Figure 4.1 in the second step of detecting the incident light. The spherical coordinates are represented as:

$$(\theta, \phi) = \left(\tan^{-1}\frac{y}{x}, \cos^{-1}\frac{z}{r}\right) \tag{4.2}$$

Where $r = \sqrt{x^2 + y^2}$ and $lights$ is the number of light detected in panoramic 360° view which influences the inverse spherical projection as follows:

$$(dw, dh) = \left(\frac{(360 \times 2 - 1)(x - 0)}{(w - 0) + 1}, \frac{(360 - 0)(x - 0)}{(w - 0) + 1}\right) \tag{4.3}$$

Eventually, the pose of each light source (θ, ϕ) in the spherical coordinates is representing with a virtual light L for several numbers of $Lights$ as follows:

$$L = f(x, y, z) = \left(\frac{\sin\phi\cos\theta}{\pi \times dh}, \frac{\cos\phi}{\pi \times dw}, \frac{\sin\phi\cos\theta}{\pi \times 90}\right), \forall L \in Lights \tag{4.4}$$

On the other hand, these coordinates have a negative direction in the reverse direction of the view as shown:

$$L = f(x, y, z) = \left(\frac{\sin\phi\cos\theta}{\pi \times (dh - 90)}, -\frac{\cos\phi}{\pi \times (dw - 180)}, \frac{\sin\phi\cos\theta}{\pi \times 90} \right), \forall L \in Lights \quad (4.5)$$

The created lights are normalized for the 3D engine where quaternions are used to represent the orientation or rotation of the lights. Also, When the light turns off it can easily be deactivated. The incident lights are essential for the AR system as it can be seen in Figure 4.2.

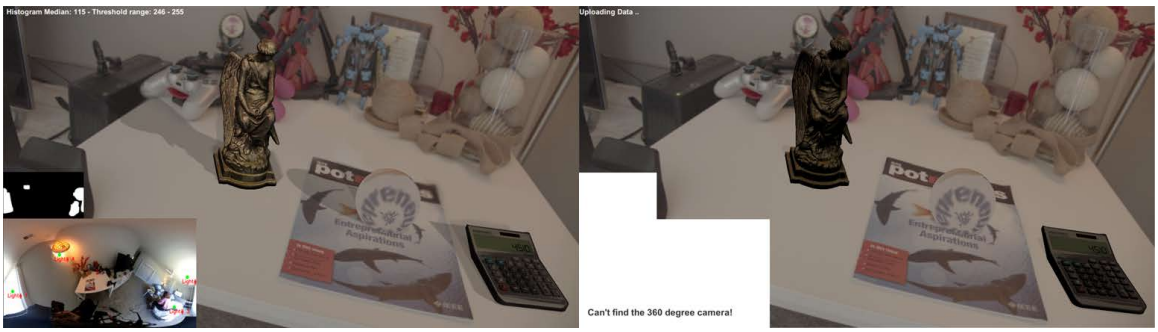

Figure 4.2. The final scene with and without the direct illumination which show the importance of incident light detection for more realistic perception.

4.2.2 Physics-based Light Polarization

Natural and artificial illumination yields light waves with electric field vectors that lie in planes oriented in every direction that is perpendicular to the propagation direction. The polarization of light is based on the quantum state of a photon as described in [115, 116]. Here we investigate the use of polarizing filters in order to focus only on the genuine light sources while reducing false-positive lights from reflections

and glares which could be mistaken as light sources. The 3D engine supports semi-transparent objects and anti-aliasing. The illuminated pixels affect the performance cost regardless of the number of lights.

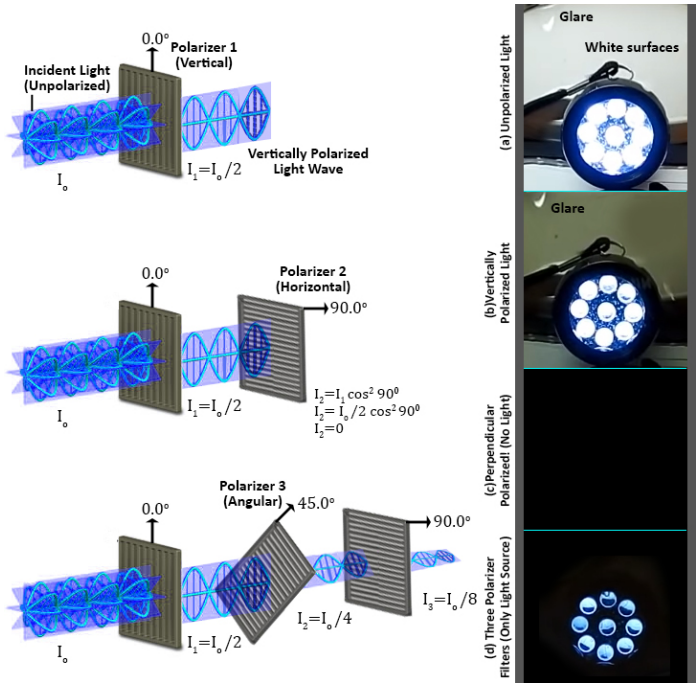

Figure 4.3. Polarization of light waves through three filters: vertical, horizontal, and angular in order to capture the light source only using Malus's Law [3]

Three polarizing film sheets are used as an initial component of the system. The light transmitted through the first filter is vertically polarized, while if a second polarizing filter perpendicular to the first one at a 90° angle is added, it absorbs the polarized light passing the first film. However, the interesting part is that when a third filter is placed in the middle of the previous two perpendicular polarizing filters at a 45° angle, it allows some of the light to get through the last filter, see the illustration in Figure 4.3.

Using matrix representations we can model the different type of polarizing filters as vectors. The vertical $|\uparrow\rangle\langle\uparrow|$, horizontal $|\rightarrow\rangle\langle\rightarrow|$, and angular $|\nearrow\rangle\langle\nearrow|$ directions can be represented as the following vectors:

$$|v\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |h\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, |\Theta\rangle = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{4.6}$$

Let the intensity of unpolarized light be noted as I_o. The polarized light passing through the three filters can be illustrated using Dirac notation for the photons. The vertically polarized light transmitted through the first film:

$$I_1 = \frac{I_o}{2}, |\uparrow\rangle = \frac{1}{\sqrt{2}}[|\nwarrow\rangle + \langle\nearrow|] \tag{4.7}$$

Followed by the photons passing through the second polarizing filter:

$$I_2 = \frac{I_o}{4}, |\nearrow\rangle = \frac{1}{\sqrt{2}}[|\uparrow\rangle + \langle\rightarrow|] \tag{4.8}$$

Producing the final notation as:

$$I_3 = \frac{I_o}{8}, |\rightarrow\rangle \tag{4.9}$$

The relation between each polarization direction and the momentum of the photon is shown in Figure 4.4.

From the description above, it can be seen that the orthogonal and normalized basis set represents $|v\rangle$ and $|h\rangle$, while $|\Theta\rangle$ represent a vector defined by $|v\rangle$ and $|h\rangle$. The normalized vector is shown as:

$$(\cos(\theta)\ \ \sin(\theta)) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} = \cos^2(\theta) + \sin^2(\theta) = 1 \tag{4.10}$$

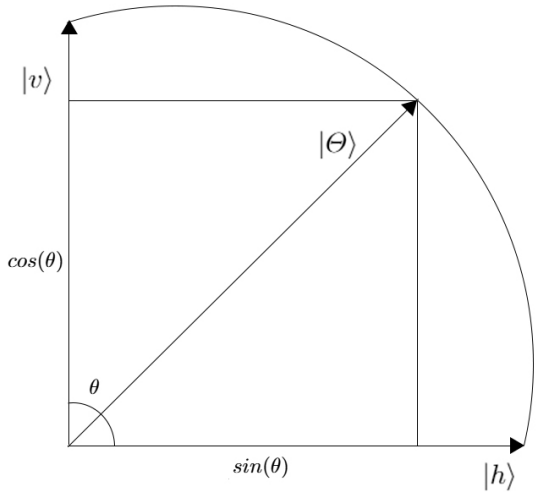

Figure 4.4. The relation between the vertical $|\uparrow\rangle\langle\uparrow|$, horizontal $|\rightarrow\rangle\langle\rightarrow|$, and angular $|\nearrow\rangle\langle\nearrow|$ polarizing films.

The angular vector can be written as a linear superposition of base states for the polarization:

$$|\Theta\rangle = |v\rangle\langle v|\Theta\rangle + |h\rangle\langle h|\Theta\rangle = |v\rangle\cos(\theta) + |h\rangle\sin(\theta) \tag{4.11}$$

The following matrix represents the vertical polarizing filter as:

$$\hat{V} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{4.12}$$

So, to define the three polarization states using the equation (4.12), the resulting measurements are:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{4.13}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{4.14}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ 0 \end{pmatrix} \tag{4.15}$$

The last polarized light is simply vertically polarized with reduced intensity. Thus, $\cos^2\theta$ is the probability for the light to pass the vertically polarizing film. The horizontal and angular filters rotated at angle θ relative to the vertical filter are represented as:

$$\hat{H} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \hat{\Theta} = \begin{pmatrix} \cos^2(\theta) & \cos(\theta)\sin(\theta) \\ \sin(\theta)\cos(\theta) & \sin^2(\theta) \end{pmatrix} \tag{4.16}$$

Therefore, the general matrix for any filter that is rotated at an angle θ relative to the vertical is generated as:

$$\hat{\theta} = |\theta\rangle\langle\theta| = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} (\cos(\theta)\sin(\theta)) = \begin{pmatrix} \cos^2(\theta) & \cos(\theta)\sin(\theta) \\ \sin(\theta)\cos(\theta) & \sin^2(\theta) \end{pmatrix} \tag{4.17}$$

The basic principle of matrix mechanics is used to examine the three polarizing filters' mathematical apparatus. The unpolarized light vibrating from the light source is a combination of every polarization angle from 0 to π radians. The θ-polarized light passing through a vertically rotated filter has probability as:

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{4.18}$$

The absolute magnitude square to the probability amplitude represents the final probability. From equation (4.18) the amount of light passing the vertical polarizing filter is 0.5 which results from integrating every angle, and with $\frac{1}{\pi}$ as the normalization constant, the value can be calculated as:

$$\frac{1}{\pi}\int_0^{\pi}\left|\begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}\right|^2 d\theta = 0.5 \tag{4.19}$$

As mentioned above, the vertically polarized light, when it passes through the horizontal filter, both perpendicular to each other, the probability is shown as:

$$\frac{1}{\pi}\int_0^{\pi}\left|\begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}\right|^2 d\theta = 0 \tag{4.20}$$

Yet, when a third 45° angled filter is sandwiched between the vertical and horizontal filters, the 45° polarizing filter operator will be:

$$\theta = \frac{\pi}{4},$$

$$\hat{\Theta} = \begin{pmatrix} \cos^2(\theta) & \cos(\theta)\sin(\theta) \\ \sin(\theta)\cos(\theta) & \sin^2(\theta) \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} \tag{4.21}$$

The final probability shows the final unpolarized light

$$\frac{1}{\pi}\int_0^{\pi}\left|\begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}\right|^2 d\theta = 0.125 \tag{4.22}$$

This probability represents the whole polarized physical scene where most of the information is lost but the incident lights in the real world can be detected in a practical approach as shown in the next section. This concept can be utilized in hardware manufacturing for direct illumination detection.

Another basic approach to proof the intensity of the light passing through the three polarizing filters is to use Malus's Law of optics which state that the intensity of polarized light after passing a rotatable polarizing filter can be calculated using the square of the cosine of the rotation angle from the position that provides maximum intensity.

An important note that the angle of the light passing through must consider both polarizing filters between the passing light. Figure 4.3 show how only the vertically polarized light is passing through the first filter as:

$$I_1 = \frac{1}{2} I_0 \tag{4.23}$$

where I_0 is unpolarized light and I_1 is the vertically polarized wave. However, when the light passes through the second filter with the 45° the intensity of light waves are calculated as follow:

$$\begin{aligned} I_2 &= I_1 \cos^2(45) \\ &= \left(\frac{\sqrt{2}}{2}\right)^2 I_1 \\ &= \frac{1}{2} I_1 \\ &= \frac{1}{2} \times \frac{1}{2} I_0 \\ &= \frac{1}{4} I_0 \end{aligned} \tag{4.24}$$

However, when calculating the intensity of the light passing through the third horizontal filter the angle assumed to be 90°, but that is not correct because we

must consider the previous amount of light passing through the second angular filter. Therefore, the angle in the third equation is actually the difference between the second and the third angles $90° - 45° = 45°$.

$$\begin{aligned} I_3 &= I_2 \cos^2(45) \\ &= \left(\frac{\sqrt{2}}{2}\right)^2 I_2 \\ &= \frac{1}{2} I_2 \\ &= \frac{1}{2} \times \frac{1}{4} I_0 \\ &= \frac{1}{8} I_0 \end{aligned} \tag{4.25}$$

As can be seen, both mathematical proofs provide us with the same amount of intensity or probability of light passing through three filters vertical, angular, and horizontal consecutively. The advantages and disadvantages of using polarization in AR applications are listed in table 4.1.

Table 4.1.
Advantages and disadvantages of using polarization in AR.

Pros	Cons
Produce more true color of the real light source.	Requires additional physical resources (Polarizing filters).
A fast and practical determination of the automatic statistical thresholds.	Environment maps cannot be used for indirect illumination simulation. Cube map method is no longer a viable option.

4.2.3 Optimized Detection of the Direct Illumination Location

The radiance map is basically an equirectangular image from a 360° camera view. A normalization of the image coordinates (x, y) is performed where the width and height of the image is w, h respectively $(u, v) = (\frac{x}{w}, \frac{y}{h})$ so the normalized coordinates range from 0 to 1. Then, the spherical coordinates (θ, ϕ) are computed where θ is assigned to be the angle from the positive x-axis with the range $0 \leq \theta \leq 2\pi$ and ϕ is assigned to be the polar angle from $Z+$ in the range $0 \leq \phi \leq \pi$ on the xy plane as: $(\theta, \phi) = (u \times 2\pi, v \times \pi)$. A unit vector is formed using the 3D polar coordinates which points toward the required face of the cube or direction in the virtual world as $(x, y, z) = (r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)$. where r is equal to one in this unit vector. In order to find the ray direction toward the center of the virtual world $(0, 0, 0)$, the maximum absolute value is computed $\max(abs(X), abs(Y), abs(Z))$ and each coordinate is divided by this maximum. The largest value determines whether the sign is positive or negative, which means that if the x coordinate is equal to $+1$ then the ray is pointing to the front or at the face $+X$ in cube mapping. The light direction from the center is known, which provides information to calculate the distance from the center to the light source. based on which direction of the spherical coordinates is the light source facing $X\pm, Y\pm, or Z\pm$ axes, the corresponding position in the virtual scene is assigned as follow:

$$f(x,y,z) = \begin{cases} (0.5x, \rho\sin\theta\sin\phi, \rho\cos\phi), & \text{for } \rho = \frac{x}{\cos\theta\sin\phi} \\ (\rho\cos\theta\sin\phi, 0.5y, \rho\cos\phi), & \text{for } \rho = \frac{y}{\sin\theta\sin\phi} \\ (\rho\cos\theta\sin\phi, \rho\sin\theta\sin\phi, 0.5z), & \text{for } \rho = \frac{z}{\cos\phi} \end{cases} \tag{4.26}$$

The current coordinates are in 3D and located in the virtual world and have the dimension $1\times1\times1$ where a virtual light L among several $Lights$ is activated to relight the scene. However, the direction of the light needs to be relative to the 360° input image. The equirectangular panorama is divided into five sections where the resulting coordinates are modified based on point of view, see Figure 4.5. The two sections of

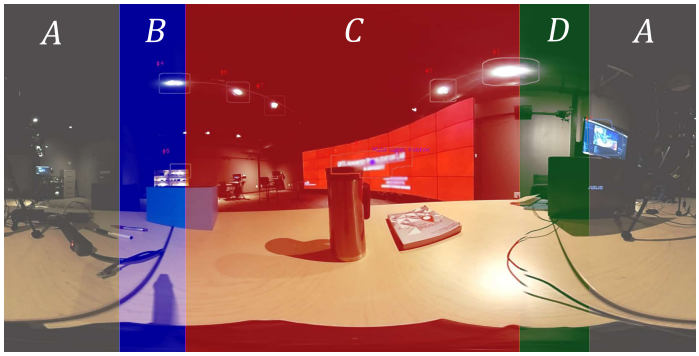

Figure 4.5. The equirectangular image of 360° camera is divided into five section to find the relative light source location based on the input data from the live-feed.

A are wrapped into one section to represent any light that falls in the back face from the point of view. While the light that hits the front face of viewpoint are confined in section C. To simplify we can represent the light locations in the following form:

$$L = f(x, y, z) = \begin{cases} (-y, -x + 0.5, z), & \text{for Front face} \\ (-0.5y, 0.5x, z), & \text{for Back face} \\ (-y, -x + 0.5, z), & \text{for Left face} \\ (-x - 1.75, z + 0.5, z), & \text{for Right face} \end{cases} \quad \forall L \in \mathit{Lights} \tag{4.27}$$

4.2.4 Sampling physical light sources

In previous work, developers usually assumed there is only one main light in AR scenes and ignore any additional lights. However, our direct illumination algorithm detects every light source in the physical scene based on topological structural analysis of 2D images. This creates the current challenge of having to incorporate multiple or extended light sources. The light source that is wrapped around the 360° view will have an accurate location depending on the center point of the light Kernel, see Figure

4.6. Only one point is considered as a light source for the spherical projection in each kernel to reduce the performance cost but as a compromise for real-time execution, more points need to be considered in order to fix this issue of extended light sources. Thus, two options are introduced to overcome this challenge.

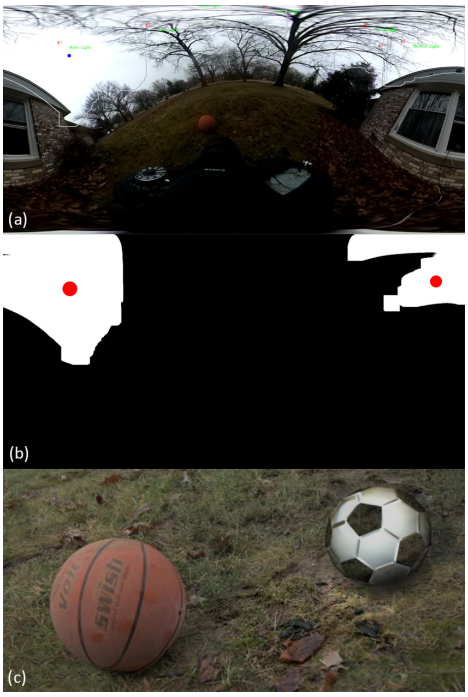

Figure 4.6. Illustration of a split one light source that is detected as two points of light sources based on the topological structural analysis of 2D images (a) original 360° view, (b) threshold of detected light sources indicating the center point of each kernel, (c) final scene where the virtual object (soccer ball) is augmented into the real world while depicting the direct and indirect illumination.

Manually Prioritize the Most Correct Light Kernel

A default approach to managing the light angle is to manually prioritize the light source that provides the most correct direction of the physical light in the final scene where shadow casting from both real and virtual objects are indistinguishable. As we can see in Figure 4.6.b one of these points can be manually chosen. A list of

all detected lights can be available for the user to prioritize. Also, the developer could automatically prioritize the one with the largest kernel, but that option is not always accurate so it was ignored. Thus, the manual activation of the virtual light corresponding to the physical/real light is more accurate and preferable, see the result in Figure 4.6.c where the shadow casting from both real and virtual objects is the same.

Statically Re-evaluate the light position

An alternative approach is to re-evaluate the light position using the two or more center points of the split single light source. Figure 4.6.a illustrates how the sunlight is split into two kernels of lights in the 360° using the topological structural analysis. The average point of the two centers is re-evaluated and considered as the shared direct light source. On the other hand, an extended area light or several point lights on both contours is another method to re-evaluate the correct light direction. In the experimental results, this approach was a success as shown in Figure 4.6.c but only works in a static environment due to the performance cost with the dynamic scenes.

5 SIMULATION OF REFLECTED LIGHTS AND GLOBAL ILLUMINATION

5.1 Introduction

In this chapter, the simulation of the reflected light (i.e. indirect illumination) will be described where the real world surroundings are captured and rendered into the virtual objects in the final scene. The reflected light is defined as the inter-object surface bouncing of light, see Figure 5.1. The indirect illumination has two separate sub-methods that are used and tested to reduce the performance cost. This chapter is a reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

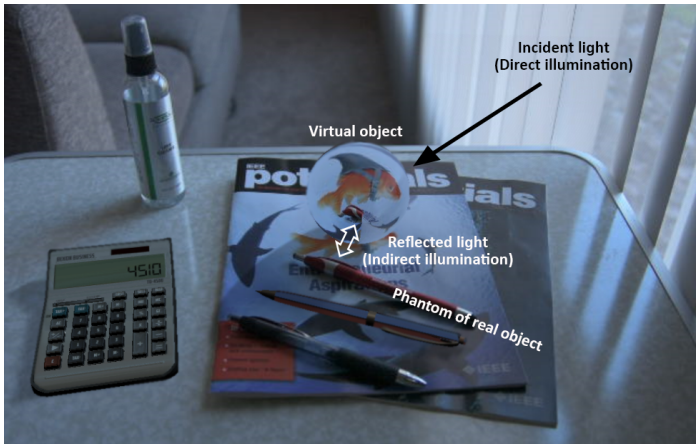

Figure 5.1. Illustration of the incident light and the reflected light differences and their interactions with the real and virtual objects [4]

The inter-reflections of lights between real and virtual surfaces are captured using two methods.

- *Global Cube Map.* The panoramic HDR of the 360° camera is used to create the 6 faces as a 2D texture to construct a cube map for the image-based lighting mode which can be modified while defining the shading properties for each virtual object.

- *Local Sampling.* The main camera of the AR view device is employed to sample the region below and surrounding each virtual object. The resulting texture is then used in IBL mode where the material property of each object can be reflected on the objects.

The resulting texture is rendered into the virtual object based on the material properties. In the shading process, these textures are added and updated as part of the image-based lighting (IBL) mode.

5.2 Method and Implementation

In this section, two sub-methods are used to simulate the reflected light which embodies the surrounding environment for each virtual object in the scene. An Empirical Evaluation for both approaches was conducted in our work [4] where the system overview was presented as in Fig. 5.2.

5.2.1 Global Cube Map.

The reflected light embodies the surrounding atmosphere where any object in the scene must be affected by it to achieve believable realism. In this section, the 360° panoramic video is also used to create a cube map that will be added onto the object shading characteristic. The number of details and HDR exposure can be changed based on the environmental intensity and the object material.

The panoramic live stream is converted to cube maps, where a co-routine is deployed to accelerate the construction at run-time. Thus, the textures are updated at periodic intervals every 10 frames while the whole code runs at 60 frames/sec. The

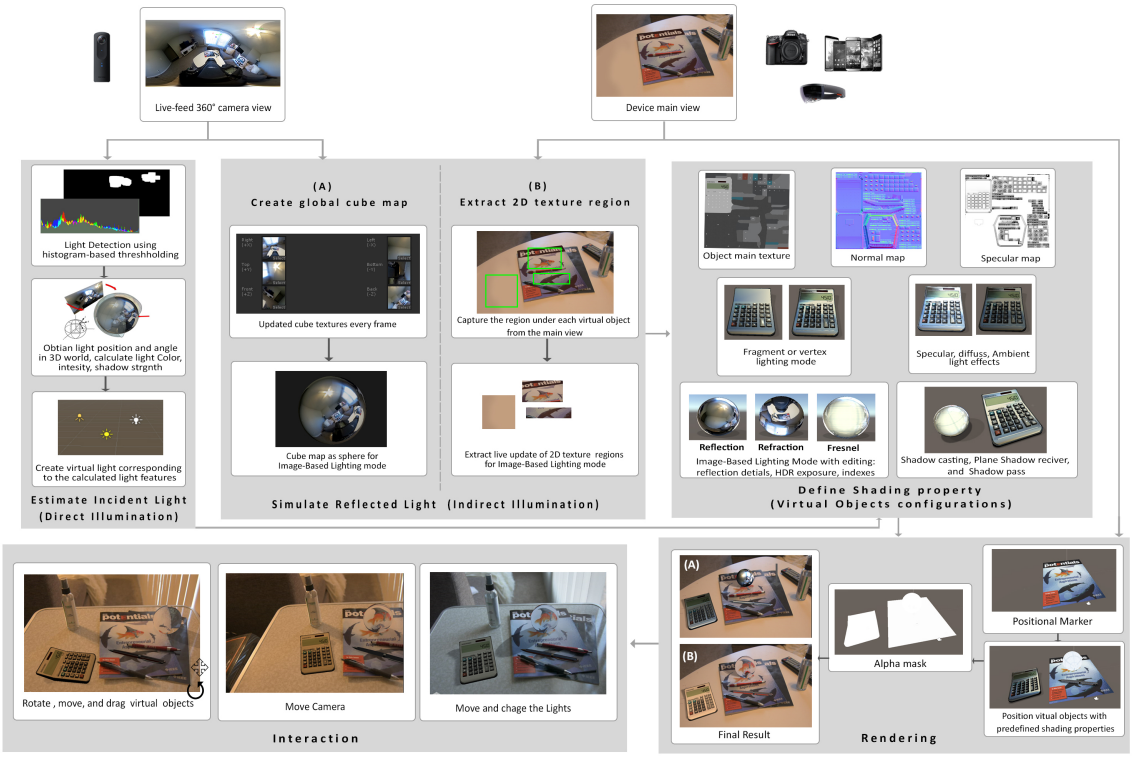

Figure 5.2. A full overview of the entire system which consists of estimate incident light, simulate reflected light, and define shading property followed by rendering and interaction [4]

spherical panoramas combine a vertical 180° angle viewing and a horizontal 360° angle viewing. Data about the light are contained from all directions in these panoramas which can be visualized on a sphere as comprising points. The cubic format, also known as cube maps, has six faces to fill the entire sphere around the viewer. The maps are created by live video feed from the 360° camera giving a left, back, right, front, top, and bottom textures. The six faces have appropriate texture maps whose area is typically arranged in a horizontal cross configuration forming an unfolded cube. When these texture maps are folded, the view is remapped to the cube faces that fit perfectly (See Fig. 5.3).

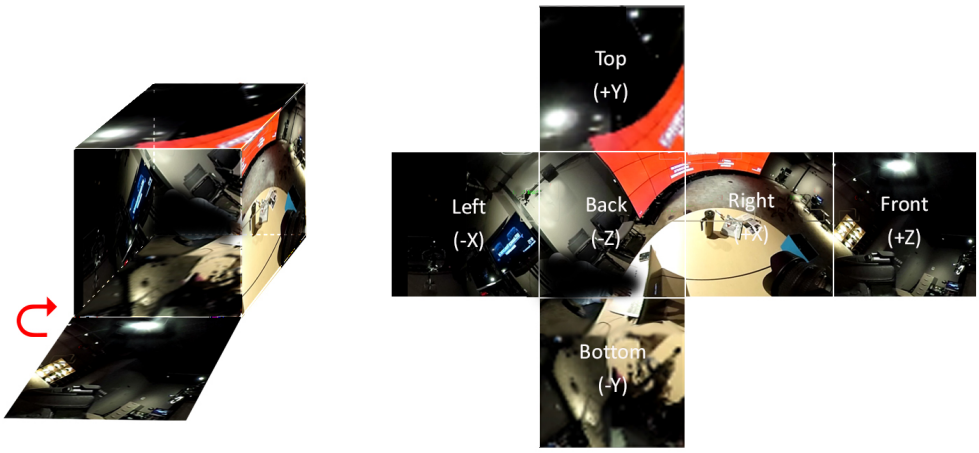

Figure 5.3. Cube map faces, an illustration of how the light is going to reflect on the virtual objects [2].

For this component, the cube map algorithms are implemented specifically for the current system. The existent online tools that provide this feature require loading a pre-recorded video or a single panoramic image which is not what the system is meant to achieve. The creation of the cube map textures from the panoramic video is about making a big box and putting it into a vertex buffer without adding texture coordinates.

Each face of the cube textures is saved as a separate image file then collected again in the arrangement. Each pixel in every face has a color that is calculated through the spherical projection then normalized for any further rotation and movements as follows:

$$(\theta, \phi) = \int_0^{points} (\tan^{-1}\frac{z}{x}, (\cos^{-1} z) + \frac{dir \times \pi}{180}) \quad (dx, dy) = \int_0^{points} (\frac{\theta}{\pi}, \frac{\phi}{\pi}) \qquad (5.1)$$

The values should be maintained inside the height and the width of the created texture, to represent the final color as:

$$(px, py) = \int_0^{points} (dx \times w, py \times h) \qquad C = \begin{pmatrix} px \\ h - py - 1 \end{pmatrix} \tag{5.2}$$

After that, the edges in the wrap mode are clamped for all the components to avoid visible seam on our textures. Minor anti-aliasing used inside filtering is applied while the camera rotates in the scene. The final step is to generate the cube maps by assigning these textures to our OpenGL cube map variables.

The resulting cube maps rendered in each virtual object share material that has the same property through the shading process as a skybox to reflect the image-based lighting mode whether it is a regular reflection, refraction, or Fresnel. Each mode and HDR exposer can be manually manipulated as needed which is discussed in the next chapter.

The cube map construction was developed purely for 3D engine C# which allows reading from a live streaming video in real-time. There are many cube map tools available but having a previous video recording or a panoramic static picture are required for these tools [2].

5.2.2 Local Sampling.

The creation of cube maps in every frame raises the performance cost, so we sample the region surrounding each virtual object from the view of the main camera which depicts illumination information from a close area around the virtual object. A similar but short version was covered in our previous work [1,4] but it was further developed and improved to fit the local sampling approach.

Therefore, a plane attached to the virtual object is added to sample the live-video texture of the AR main camera. A mesh filter is initialized to keep the vertices of the mesh updated at 100 frames per second. A culling mask is used to hide the undesirable part of the view inside the object layer.

The mesh renderer of the background plane gets the main texture and assigns it to the video texture. The background plane is dispatched every 10 frames per second. Each vertex in the final mesh is transformed from the 3D world to the viewport point. The region capturing in progress the Vuforia device orientation is modified based on the screen orientation. Also, the local texture resolution is transformed based on the local scale of the entire camera output texture.

A virtual camera renderer captures the target texture to set the image-based lighting mode with that texture to define the material property of the virtual object.

The advantage of this sub-method is to sample only the local region on a specific object instead of the whole environment. It provides a suitable outcome for diffuse and glass materials, however, the cube map provides better results with the specular materials.

6 SHADING VIRTUAL SCENE SURFACES

6.1 Introduction

In this chapter, the standard graphics shading language that was adopted for shader programming for providing several effects: characterized surfaces, volumes, and objects. The shading process defines special data types, such as vector, matrix, color, and normal. Several shading languages have been developed due to varieties of 3D computer graphics applications. The system used OpenGL shading language also known as GLSL or Glslang which unifies vertex and fragment processing in a single set of instructions which allows more conditional loops and branches. This chapter is reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

6.2 What is a Shader?

A virtual object contains vertices, vertex colors, UV data, and normal which is rendered with material assigned based on the lighting condition and the scene perspective. The material uses a shader with textures, colors, and other property values. The shading process obtains these data from the virtual object materials to draw pixels to the screen depending on its CG or GLSL code.

As known, the shader is a collective computation of the shading properties during rendering which runs on GPU. For a realistic object, the proper level of light, darkness, color must be considered in the final image. The virtual object consists of vertices, UV information, and normals as part of the material features. Some of these properties are predefined such as the objects main texture and normal map, while others can

be updated/manipulated at the run-time like image-based lighting textures or cube maps.

In this system, two types of shaders are used to fully define the entire shading properties based on how the virtual objects receive the lights.

Lit Surface.

The objects that receive and reflect the lights in the virtual scene have this type of shader which provides three passes: first forward pass for the main light, second forward pass adds any additional lights, and the third pass for casting shadows on other surfaces.

Also, it contains various properties such as vertex/fragment lighting mode, normal mapping, Ambient light, Lambert diffuse, Blinn Phong specular, Fresnel, image-based Lighting mode, Ashikhmin, Shirley and Premoze BRDF anisotropy.

Unlit Surface

The object that only casts shadows without receiving any light is assigned to this shader. The background of the virtual scene should be hidden but must reflect the shadow of other virtual objects. Therefore, an alpha mask is used to cut-out the main color of the background plane to produce a transparent material looking like it receives the shadow but nothing else.

6.3 Normal Mapping

The development of computer graphics, 3D modeling software, or the content creation software support high poly meshes. However, in 3D engines and some gaming devices supporting very high poly meshes can be challenging, therefore, the concept of normal mapping was introduce to overcome this problem. Normal mapping is a technique for lighting a 3D model with a low poly mesh as if it were a more detailed

model. The normal mapping starts by reading the normals (x, y, z) of high poly mesh, then baked onto a texture using three different color channels (R, G, B) known as a normal map. A low poly mesh is used to fake the details of a high poly mesh in the game, AR platform. When the lighting mode is changing, the fake details of the mesh (shadow, highlights) follow that change based on the light that reacts toward the normal maps applied on the mesh.

The normal map has two major types: world-space, object-space. If the normal map was baked on the world-space positions of these normals, the normal map is called a world-space normal map. However, When transforming a virtual object from the 3D software to another platform as the game engine, the world space position must be calculated based on the world-space normal map that was created in the 3D software because it is challenging to know exactly where the object was placed while baking the normal map. If the object position is changed in the game engine, there is no way of calculating a new world-space position for those normals from the previous world-space position where the object was created initially. Thus, in order to use the world-space normal map, the object position in the game engine must be the same position in the 3D software or the mesh must be static.

Nevertheless, the new world-space position can be determined from the object-space where the model matrix can convert from object-space to world-space. The object normal information can be read from the normal map inside the fragment shader to read the lighting information per pixel. To conclude, the interaction between object normals and lights can be calculated if they are in the same coordinate space and that the main reason for simulating the illumination condition of the real world into the virtual world in our system.

Transform the normal map from object-space to the world normal. The normal map of an object is transfer into the world space by taking a 2D texture for each object where the normal at a pixel is converted from the color value. Then,

composed and normalized the tangent, bi-normal, and normal (TBN) matrix. After that, the returned color is defined as property in the shader.

6.4 Diffuse Lighting/Reflection

According to the basic lighting model described in 2.2, the surface color is a summation of emissive, ambient, diffuse, and specular light. In this section, the diffuse light is explored by the basic understanding of the human vision. The object seems visible when the light is emitted from the source. Then, it hits the object's surface, get reflected, and reaches the observer's eye. The objects that do not reflect the light or absorb it completely are invisible to the eye. Thus, in order for the surface to be visible, it must have light-reflecting property.

Diffuse reflection is the property of rough surfaces. The ideal rough surface reflects the light in all directions equally, but in reality, that surface does not exist. However, Lambert-cosine law defined by Lambert [117] is used in our shader for virtual object reflection which is represented as follow.

Lambert Diffuse Model.

$$d = \sum_{i=0}^{n} c \times f \times a \times max(0, \overrightarrow{N} \cdot \overrightarrow{L}) \tag{6.1}$$

Where $\overrightarrow{N}$ normal direction, $\overrightarrow{L}$ light direction, (c) color, (f) diffuse factor and (a) attenuation.

6.5 Specular Lighting/Reflection

Specular reflection is a property depicted by smooth surfaces or reflective surfaces in the form of specular highlights or shining material. The phenomena are explained by the law of reflection where the "angle of incidence" i.e., the angle between the normal and incident ray, equals the "angle of reflection" i.e., the angle between the normal and the reflected ray, see Figure 2.2. In the diffuse surfaces, the light reflected

in all the directions with a cosine fall-off, but on the specular surfaces, the light is reflected only in one direction which means it may not travel into the eye. Thus, the specular reflection is view-port/eye dependent.

Bui Thong Phong [118] observed these phenomena where the specular highlight is small for shiny surfaces with rapid fall-off intensity, see the object (c) in Figure 6.1. On the other hand, the highlight is large for less shiny surfaces and falls off slowly, see the object (a, b) in Figure 6.1. Phong adds a "cone of reflectance" on the law of reflection model to describe the specular reflection where the angle of the cone defines the area of visibility of the specular highlight and the diameter of the cone base defines the specular property of the surface material. Therefore, the intensity of the specular reflection is defined by the angle between the eye/viewpoint vector and reflected ray. The projection of the negative incident vector onto the normal vector is required to calculate the specularly reflected vector with respect to the viewpoint vector.

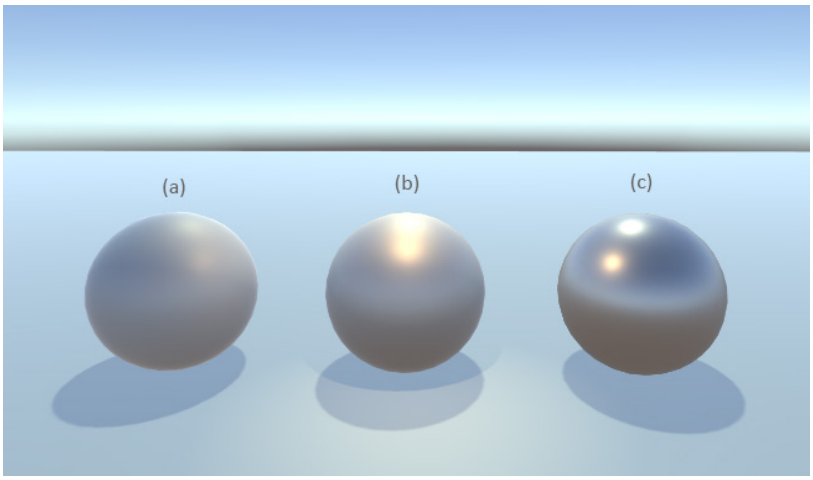

Figure 6.1. different specular highlights based on the surface materials, (a) least sharper highlight or diffuse reflection, (b) sharper highlights, (c) the most sharper highlights, or specular.

Blinn-Phong Specular Model. Blinn [119] equation is a developed version of Bui Thong Phong which focuses on the halfway vector $\overrightarrow{h}$ (between incident light and viewpoint vector $\overrightarrow{L} \cdot \overrightarrow{V}$) with respect to normal vector instead of reflected and viewpoint vector $\overrightarrow{R} \cdot \overrightarrow{V}$. This formula requires previous knowledge of mesh normal ($\overrightarrow{N}$), light direction ($\overrightarrow{L}$) with world space viewpoint vector (V), specular color (S_c), specular factor (S_f), attenuation (a) and specular power (S_p). In order to calculate the halfway vector direction $\overrightarrow{h}$, a normalization of the light direction $\overrightarrow{L}$ and world space viewpoint vector $\overrightarrow{V}$ are performed which is presented in the specular model:

$$S = \sum_{i=0}^{n} S_c \times S_f \times a \times max(0, \overrightarrow{N} \cdot \overrightarrow{h})_p^S \tag{6.2}$$

6.6 Image-Based Lighting Model

Image-Based Lighting Reflection. The simulated reflected light in the form of a cube map or local 2D textures is assigned in this part where the required computation for each type is redefined based on the texture form. The texture will stay updated with live-feed video in real-time.

Image-Based Lighting Refraction. same as the previous description except the received texture is bent to mimic the refraction effect. Snells law is used to represent the refraction model:

$$Refraction(R) = e \times i + [(e \cos_i \theta) - \sqrt{1 - e^2(1 - \cos_r^2 \theta)}]\overrightarrow{N} \tag{6.3}$$

Where (i) is the velocity of light in vacuum and (r) is the velocity of light in the medium.

Image-Based Lighting Fresnel. It simulates the glass and water reflection or refraction where the viewers point of view influences the normal vector.

$$Fresnel(F) = f + (1 - f)(1 - \overrightarrow{V} \cdot \overrightarrow{h})^5 \tag{6.4}$$

Figure 6.2. The effect of image-based Fresnel on the glass [2]

Ashikhmin, Shirley and Premoze BRDF Anisotropy. A function that defines a certain pattern of how the light is going to hit a surface material such as silver or copper, The BRDF secular reflectively model used is:

$$S = \frac{\sqrt{(n_u + 1)(n_v + 1)}(\overrightarrow{N} \cdot \overrightarrow{h})^{\frac{n_u(\overrightarrow{h}.T)^2 + n_v(\overrightarrow{h}.\overrightarrow{B})^2}{1-(\overrightarrow{N}.\overrightarrow{h})^2}}}{8\pi \times (\overrightarrow{v}.\overrightarrow{h}) \times max((\overrightarrow{N}.\overrightarrow{L}), (\overrightarrow{N}.\overrightarrow{V}))} \times F \tag{6.5}$$

where (n_u, n_v) refer to width/height of tangent map, while $(\overrightarrow{N})$ is normal vector, $(\overrightarrow{h})$ halfway vector, $(\overrightarrow{T})$ tangent vector, $(\overrightarrow{B})$ BiTangent vector.

6.7 Shadow Mapping

Shadow mapping is a technique for casting and generating shadows in the final scene. Shadow mapping is based on a concept that the light has eyes, if the light can

see the surface then it is lit, but if the light can't see the surface it will be dark. The light mapping is performed in two steps specified based on the 3D engine.

In the first step, the depth map of the scene is baked/created from the light's eye which mean render the scene form the light's point of view instead of the camera's point of view. The light will be considered as a camera with a frustum that is orthographics (square/rectangular) for the directional light, and perspective for the point light. The z-depth after the z-test for every fragment is stored in the depth buffer in the form of a texture refer to as light depth map texture which is different than the depth map that is rendered by the camera. For instance, if we have a sphere above a plane and the light is above the sphere, the first object that is being rendered is the plane with a z-test for every fragment based on the distance from the light which will be stored in the depth buffer. Then, the sphere is rendered where the fragments at the smaller distance from the light. Thus, the z-depth of this fragment will be stored in the texture first. The world space position will transformed into the view space of the light then into the projection space of the light using the view metric/projection matrix of the light. The projection space is transformed into normalized device coordinates that is then transformed into texture space in order to stored the depth map in the form of texture.

In the second step, the scene is rendered from camera's point of view as usual following the same rendering pipeline of the first step. For every fragment, the depth map is drawn based on the distance from the light is checked against the sampled depth stored in the light depth map texture. Test the z-depth of every fragment from the light against the sampled z-depth of the light depth map texture. if the z-depth of a fragment is greater than the sampled z-depth that means the fragment is in the dark, while the else fragment is lit. This way. we will figure out the result of every fragment to be drawn and mark them as (1 for lit fragment) or (0 for shadow fragment). The 3D engine stored this complete map of 1's and 0's as global variable, that will be provided in our shader to indicate which fragment is lit or in shadow.

7 PHYSICAL GEOMETRIC PROPERTIES OF REAL-SCENES

7.1 Introduction

This chapter covers the process of transforming independent acquisitions of physical geometry, or point clouds, into a surface of triangulated mesh using localization and mapping methods and can be fulfilled with different algorithmic approaches. The geometric properties of real-scene including plane detection, 3D surfaces reconstruction and simple meshing are investigated to provide more realistic features for the augmented scene into the real world. This chapter is a reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

7.2 Method and Implementation

The interaction from real-to-virtual has different requirements than the interaction from virtual-to-real; the latter must address the real-scene geometric understanding and depth information. In the previous sections, we can see the effects of image-based lighting and marker tracking improving realism on the virtual objects. However, in order to improve that outcome with a sense of occlusion and inclusion of the physical environment into the virtual scene, geometric reconstruction of the real-scene is a helpful tool to adopt in our system [5]. This component provides an understanding of the real environment without building a perfect mesh reconstruction that could reduce the real-time performance cost but is good enough to indicate the real object locations and some features, see Figure 7.1. The virtual object only needs to provide a feel for the existence of the real objects with a sense of lighting color for reflected light

estimation, see Figure 7.2. The model of a physical surface goes through the scanning device and registration process to acquire cloud points for geometric reconstructions.

Figure 7.1. The real-word depth data and marker features estimation in red for both indoor and outdoor environments with different lighting conditions.

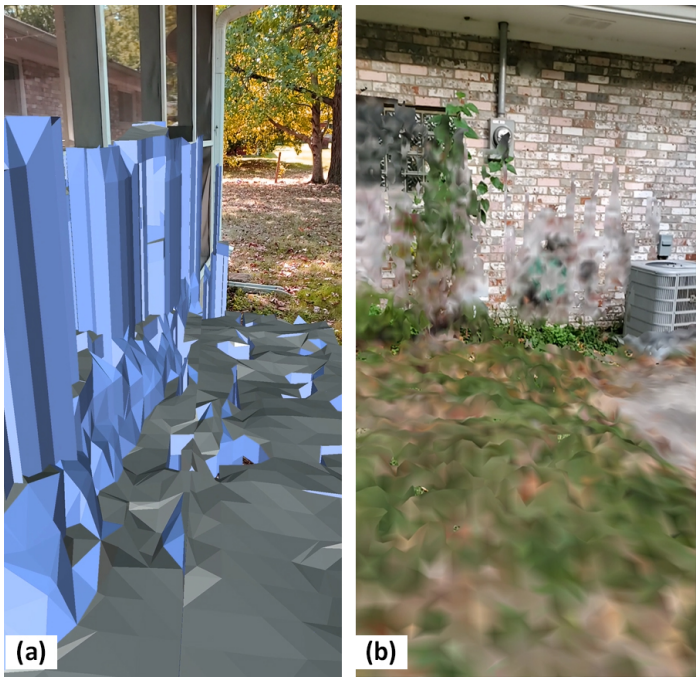

Figure 7.2. Physical environment reconstruction: (a) plane detection, (b) mesh building that merges the color of the original plane for more realistic reflected light simulation [5].

A physical model in the process of surface reconstruction goes through scanning device and registration to acquire cloud points in order to achieve geometric reconstructions. A set of sample points as input with/without normal vectors estimation is the most general approach [120]. A set of feature points obtained utilizing augmented reality tool $P = \{p_1, ..., p_n\}$ where $p_i \in \mathbb{R}^3$ is transformed into screen coordinates. The implicit geometry representations has zero set of 2D distance function $F(x, y) = \sqrt{(x^2 + y^2)} - r$ to define a 1D curve with which it is easy to handle different topologies. The goal is to find a manifold surface $S \subset \mathbb{R}^3$ that approximates P where $S = \{x | d(x) = 0\}$ with $d(x)$ a signed distance function (SDF).

As known that the SDF of a set S in a metric space can be determined by the distance of a given point x from the boundary of S, and the sign determines by whether x is in S. If the value is inside S that mean the function has positive values at points x. The value decreases if x approaches the boundary of S where the SDF is zero, and it takes negative values if the value outside of S. However, a reverse convention is also considered instead (i.e., negative inside S and positive outside) [121]

It is important to note that the motion tracking provided by the AR tool is not always accurate which makes it impossible to create homogeneous point clouds, then reconstruct surfaces based on that, however, these points can be accurate per frame. Thus, a Delaunay triangulation is performed on the convex hull of the points from the actual frame in which every circumcircle of a triangle is an empty circle [122] to provide a regular distribution of points for estimating SDF.

The neighborhood of a sample point in every relevant directions are explored using Delaunay triangulation which even accommodates non-uniform samplings. The tangent plane methods of Delaunay triangulation considers a dense sampling on a smooth surface, thus the neighbors of a point in the point cloud would not deviate too much from the tangent plane of the surface at that point. The tangent plane can be suitably approximated by exploiting the condition of sampling the Voronoi cell of the sample point is elongated in the direction of the surface normal at the sample

point. The local triangulation around each point can be derived from this normal or tangent plane information, respectively [122, 123].

The direct use of SDF without Delaunay triangulation enables scanning with a uniform grid 15 cm which is too smooth for our purposes, i.e., the vertices in the range ±15 cm are smoothed together. The reconstructed model would cover the area and it works great for a grid of 4 cm $d(i), i = [i, j, k] \in \mathbb{R}^3$. For reduced performance cost, working with a grid of 2 cm was not considered.

Finally, we extract zero isosurfaces by Marching Cubes to create the mesh by approximation of input points which results in a closed two-manifold surface. This part of the system runs on a smart phone with support of Google ARCore features in order to perform plane detection, mesh creation, and rough 3D surface reconstruction, see Figure 7.3, the results of this experiment is presented and evaluated in the next chapter.

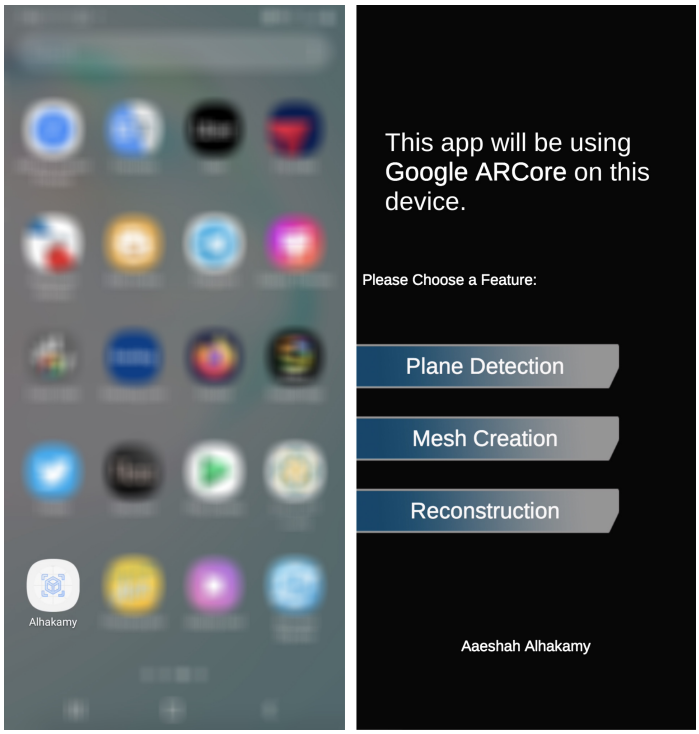

Figure 7.3. Smart phone application to perform plane detection, mesh creation, and 3D surface reconstruction of the real-scene geometry

8 EVALUATION AND DISCUSSION

8.1 Introduction

This chapter evaluates the system for interactive illumination of augmented reality in a dynamic environment, presented in previous chapters. The rendering time of the presented system conditions and algorithms can be measured to asses the best performance cost. The rendering process of these algorithms is compared to result in images to asses the quality. In addition to the rendering performance and quality exact measurements, the user perception impacts most of the evaluation of AR systems. The result of user evaluation presents the effect of several visual features rendered by the described system for realism perception. This chapter is reprint [with minor edits] of some passages, figures, tables and algorithms have been quoted verbatim from our published work in [1–6].

8.2 Initial Experiment

The primary results of our work depicted the direct and indirect illumination instantly based on the system overview in Figure 8.1. Then, we confirmed the outcome of the method using other lighting conditions such as white light, dim yellow light, sunlight, the light coming from a window, and others. We also tested our system with different scene-setting, i.e., indoor, outdoor, other objects types, or other locations. By observing the shadow falling from the real objects compared to the shadow of the virtual object, we could promptly evaluate the estimated incident light. The reflected light could observe the influence on the virtual objects when a real object moved, or when real light hit the scene immediately, see Figure 8.2.

Figure 8.1. Initial System overview which includes the main three part of the whole system: direct illumination estimation, indirect illumination extraction, and shading features, also providing the rendering and interaction procedure [1], for clear diagram and methods see Chapter 3.

8.3 Cube Map Experiment

Multiple scenes are tested in different lighting conditions with diverse environmental settings (e.g., dim yellow lamplight, white lamplight, direct sunlight, sunlight from the window) to evaluate the visual quality and light estimation system presented in Figure 8.3. All trials were with the dynamic video feed. Most of our evaluations

Figure 8.2. Initial results, each case has: live 360 view, threshold mask of direct illumination detection, final result [1]

are based on shadow observation by comparing the shadows of the real and virtual objects in the final scene. Furthermore, the light falling and reflected on the virtual objects is a valid indication of the similarity to the real light which exists in the real environment. The results continued to be valid even when the lights changed, and when the virtual objects moved, rotated, and the marker or camera moved.

Furthermore, the data feedback from 33 college students are evaluated where a questionnaire about the final scene is presented to the subjects, see Figure 8.4. Most of the collected data are focused on rating the objects' realism where participants answered on a scale from 1 ('looks synthetic') to 5 ('looks real'). As many subjects could not differentiate between the real and virtual objects, the average of the real objects that were mistaken as virtual objects are subtracted from the original rating. The virtual objects scored values above the average, see Figure 8.5. Participants were also asked to provide a general opinion on the correctness of the light and shadow; the lighting on the virtual objects has a gap greater than 39.01% while the shadow was more believable by a difference of 18% compared with the real objects.

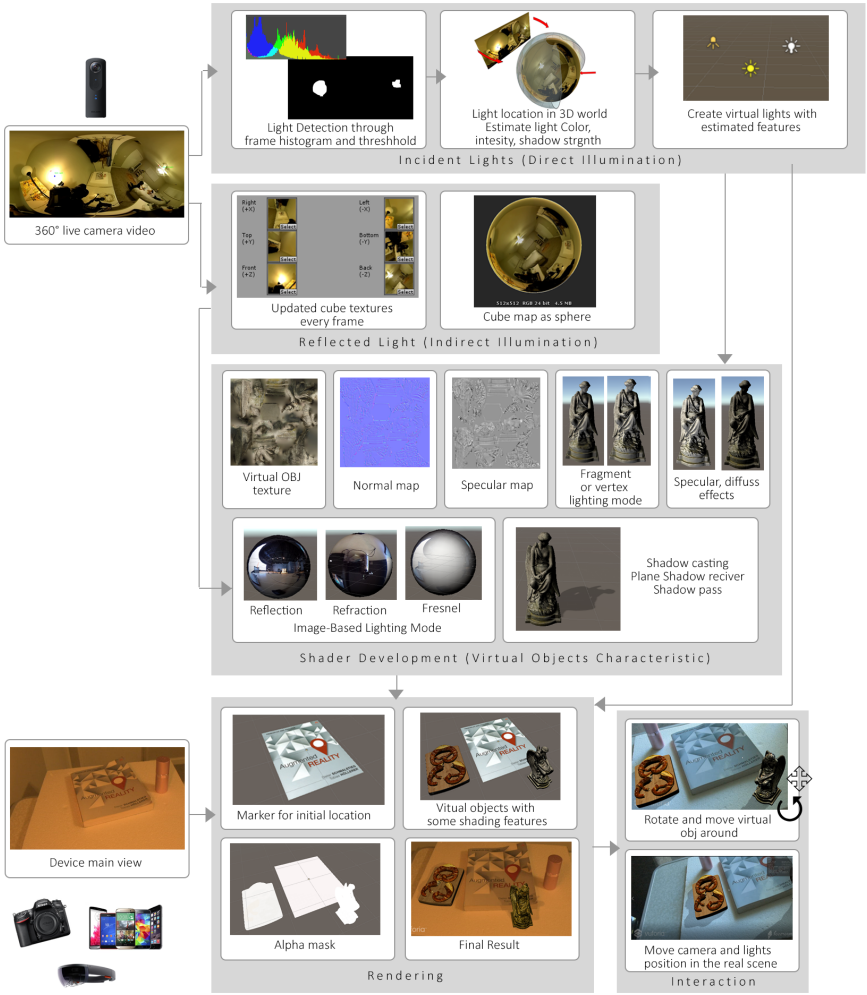

Figure 8.3. An overview of the cube map experiment system including the three components: incident light detection, reflected light simulation, and shading materials, in addition to the rendering and interaction process [2], for clear diagram and methods see Chapter 3.

The feedback from the shadow allocation indicates that the light detection should provide an insight to calculate the light angle of the real objects' shadows manually, then compare it with the angles that the system estimated (the real object length was 7.8 cm). The difference margin in degrees was found to be insignificant as it is shown in Table 8.1. The light was concentrated based on the area center in the threshold where the angle computed manually is illustrated in Figure 8.6. The statistics of these

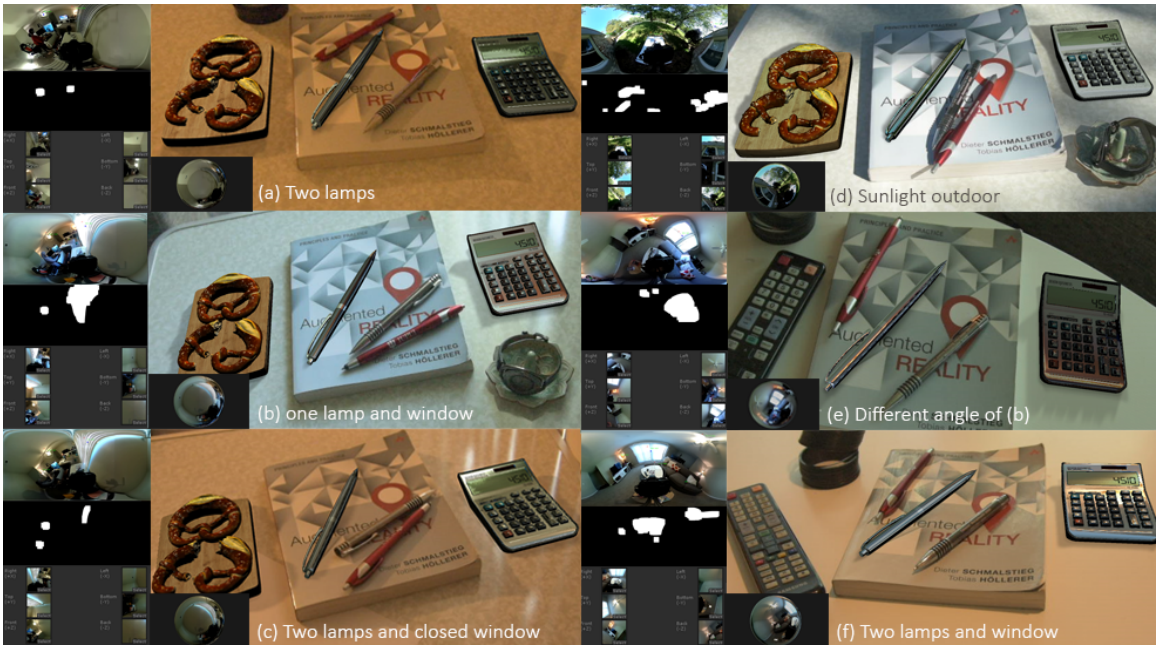

Figure 8.4. Cube map experiment results have different lighting conditions: (a) two lamps (b) one lamp and window, (c) two lamps and closed window, (d) sunlight outdoor, (e) different angle of (b), and (f) two lamps and window, and in each condition we can see the panoramic 360° view, the threshold mask for the incident light detection, cube map faces for the surrounding environment for the reflected illumination, the environment map as a sphere, and final scene [2]

errors can be computed using the root-mean-square error $(RMSE) = \sqrt{\frac{\sum_{i=1}^{3} Error_i^2}{3}} = 1.38$. The average error is slightly more than a single degree which is quite accurate considering the difficulty of selecting the center of the light. The absence of significant trends in the errors suggests that the light estimation model is sufficiently precise.

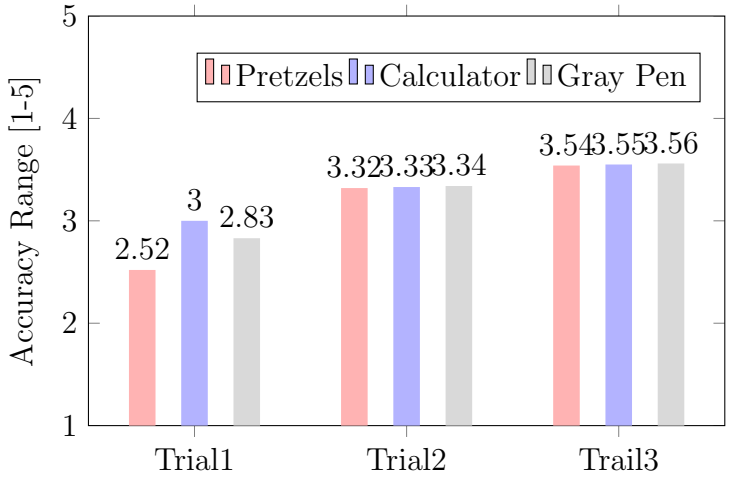

Figure 8.5. Rating of the virtual objects realism in comparison to each other [2]

Table 8.1.
The difference (error) between the measured and estimated angles of the light in degrees [2]

Scene No.	Shadow length (s)	Measured ($\theta°$)	Detected ($\phi°$)	Error
Trial1	6	-127.56	-126.98	0.58
Trial2	15	27.47	26.24	1.23
Trial3	18	203.42	203.01	0.41

8.4 Local Sampling vs Global Cube Map Experiment

The developed system described in Figure 8.7 is examined through multiple scenes with different lighting conditions and various environment settings. For further analysis, the results are evaluated based on three categories:

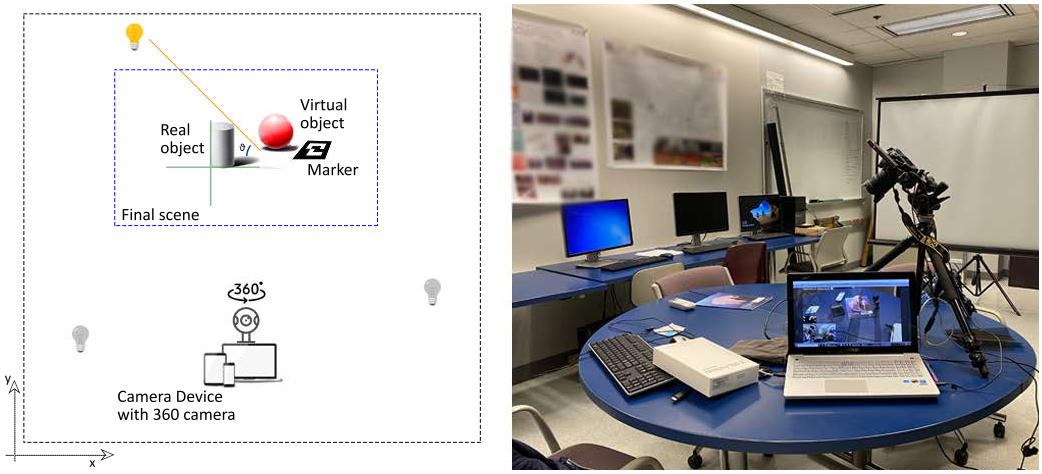

Figure 8.6. System settings and an illustration of how the measured angle of the incident light is calculated then compared to the detected/estimated angle using the system algorithm [5].

8.4.1 Incident Light Evaluation

Shadow observation is the most visible clue that can be used to determine the accuracy of the incident light estimation in our experiment, in addition to the light color and intensity of the whole scene. The shadow cast from the real objects in comparison with the shadow cast from the virtual objects is a tangible evidence for the estimated light angle validation. For the real objects, the angle of the physical light is calculated manually by measuring the real object length (t) in the physical scene, see Figure 8.6 with the aspect ratio to the shadow length (s) in six scenarios to obtain the measured angle as $\theta = \tan^{-1}\frac{t}{s}$. The measured angle θ then is compared to the estimated angle ϕ presented in the virtual light y-axis where both angles provide a small margin of error. Table 8.2 shows the difference between the measured angle θ and the estimated angle ϕ is a reasonable value which proves that our estimation is nearly accurate. The computed statistics of the errors combined are averaged also using the root-mean-square error $RMSE = \sqrt{\frac{\sum_{i=1}^{6} Error_i^2}{6}} = 1.567$.

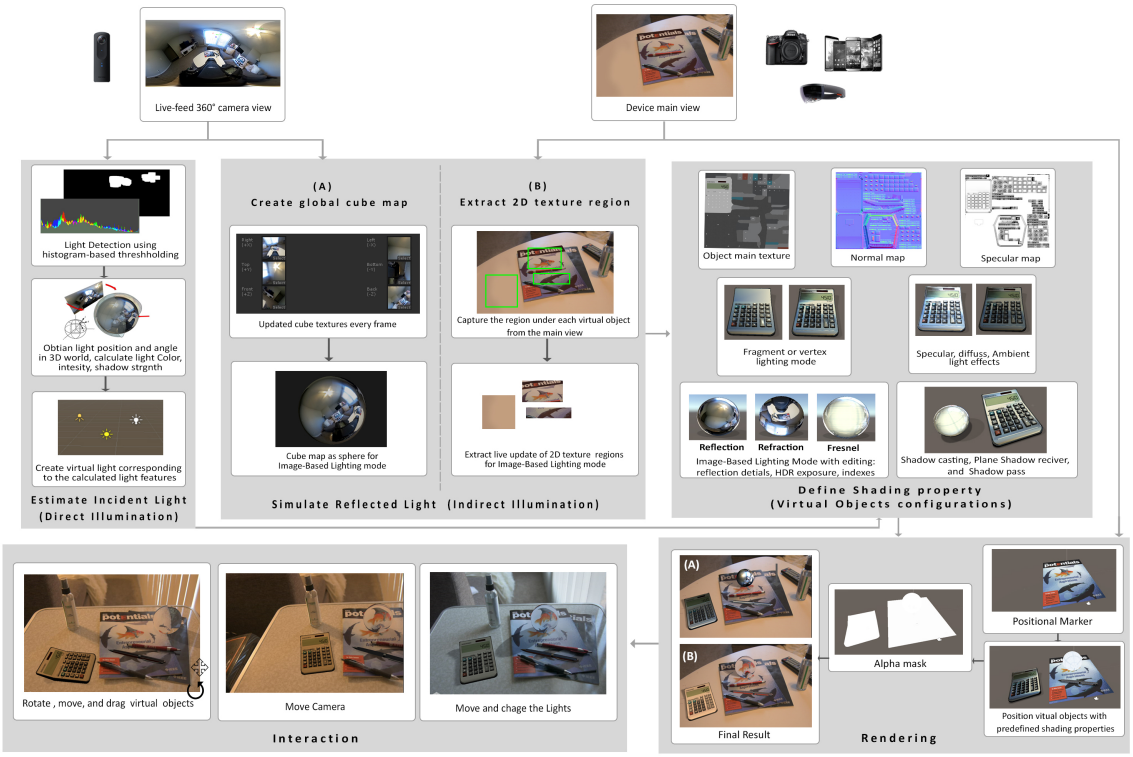

Figure 8.7. A full overview of the local sampling vs global cube map experiment system which consists of estimate incident light, simulate reflected light including both algorithms: local sampling vs global cube map, and define shading property followed by rendering and interaction [4], for clear diagram and methods see Chapter 3.

8.4.2 Performance Evaluation

The reason for presenting two sub-methods for simulating the reflected light is to reduce the performance cost as mentioned above. In this section, we evaluate the performance through different scenes for both global cube maps and local sampling. Table 8.3 shows that the number of virtual objects in each scene has a limited influence on the performance cost. Although the local sampling reduces the performance cost significantly, the camera render is delayed by approximately 1 ms. In order to sample the required local area from the main view in certain regions related to the location

Table 8.2.
Local sampling vs global cube map experiment: The error margin between the measured (θ) and estimated (ϕ) angles of the incident light in degrees [4]

Scene No.	Shadow length (s)	Measured (θ°)	Detected (ϕ°)	Error
1st	28	-167.04	-166.12	0.2
2nd	5.4	-50.19	-49.01	-1.18
3rd	15	27.47	26.24	1.23
4th	6	-127.56	-126.98	0.58
5th	18	203.42	203.01	0.41
6th	4	58.39	61.10	-2.71

of the virtual object, the performance cost has to be increased at a small fraction compared to the enhanced performance in the update function and other aspects of the system.

Table 8.3.
Performance evaluation for the global cube map against the local sampling with different number of objects [4]

Operation	Global Cube Map			Local Sampling		
	1	2	3	1	2	3
FPS [1/s]	6	5	4	45	40	36
Update [ms]	173.4	177.3	183.2	35.9	37.8	40.2
Input [ms]	0.04	0.05	0.07	0.05	0.06	0.06
Surfaces [ms]	4.21	4.98	5.48	4.42	4.69	4.75
Camera render[ms]	0.66	0.68	0.84	1.08	1.40	1.69
Rendering [ms]	0.03	0.04	0.05	0.04	0.05	0.06

8.4.3 User-Feedback Evaluation

A small user study is conducted as an online survey that contains several pictures and videos of the results in different lighting conditions. The feedback data from 33 subjects who identified as college students shows that the incident light estimation is 51.2 % more accurate than the average of faulty results. However, the angle of virtual objects shadow scored 20.4 % higher than the estimation of the incident light. This observation is what led to the calculation of incident light angle above, see Figure 8.8. The results from the reflected light simulation method are not the same as shown in Figure 8.9 where the subjects are asked to evaluate which column presented more realistic results. The sampling of local regions had 66.67 % more realistic results than the global cube map.

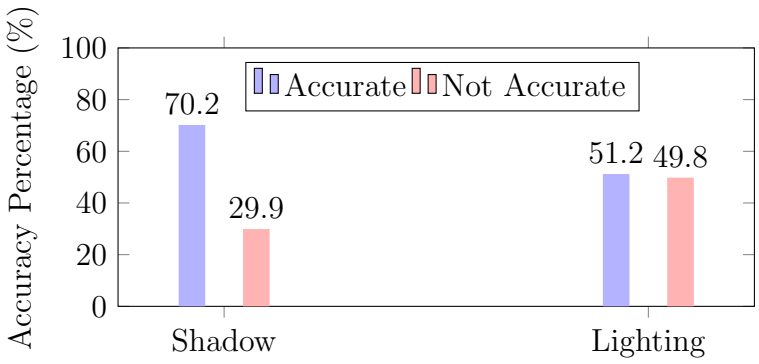

Figure 8.8. User-feedback for the incident light estimation compared to the shadow accuracy in the local sampling vs global cube map experiment [4].

Furthermore, each object in the scene was rated based on how realistic they look compared to the other object whether it was real or virtual. The user ranked the object from 0 to 10 range as [”very virtual, 0”, ”virtual”, ”natural”, ”realistic”, ”very realistic, 10”]. Some users believed that some real objects were virtual by 1.5 points of the ranking which is proof that the system works. While there is a slight misconception recognizing the virtual object from the real ones, the rating is also

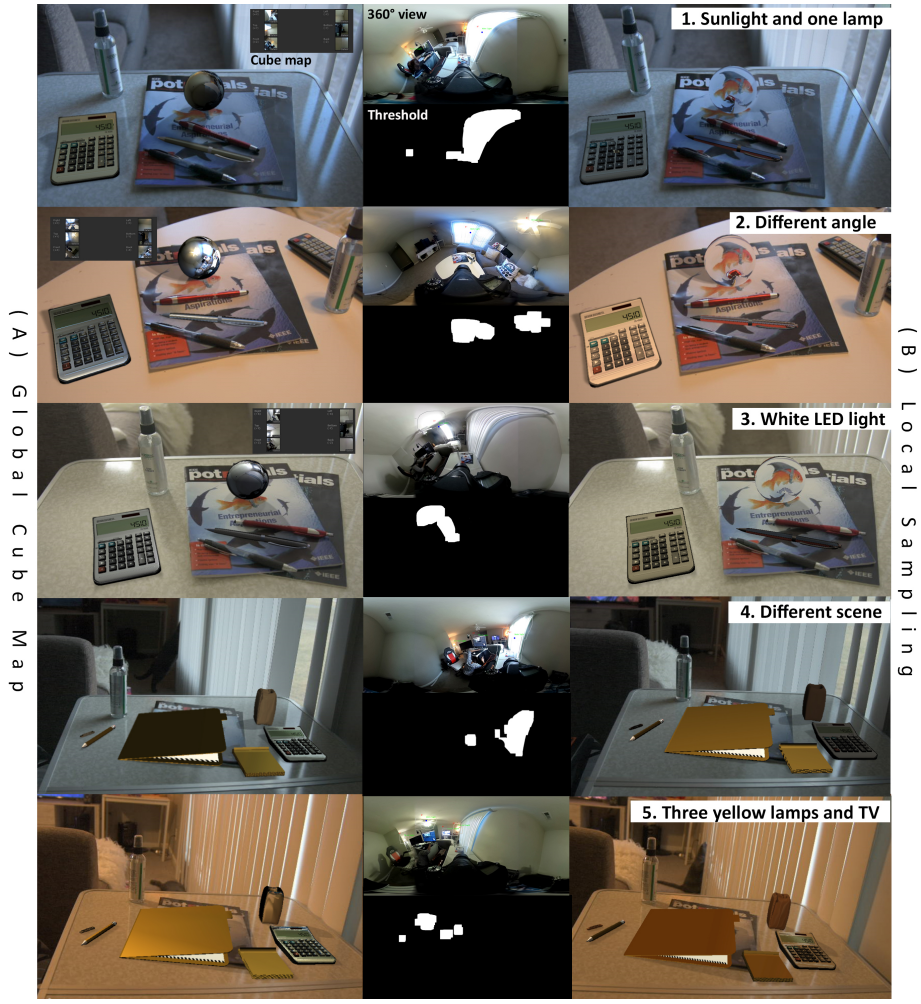

Figure 8.9. Local sampling vs global cube map experiment results where each environment condition has: (1) 360° view, (2) histogram-based thresholding, (3) cube map textures, (4) final scene of the global cube map creation result, and (5) final scene of the local sampling [4].

influenced by that misconception. Thus, the results are re-evaluated for the final feedback from the users based on the scene that provided the best realistic outcome according to the subjects recommendations where all the virtual objects scores above the average, see Figure 8.10. The reflected light sub-methods influence the shading properties, and it seems that the global cube map creation provides more realistic results for the metal material objects where the local sampling has a more realistic

outcome when the materials are glass or transparent. Finally, the lighting conditions are updated and changing in real-time accordingly, even when the object, camera, or light are moving based on their locations.

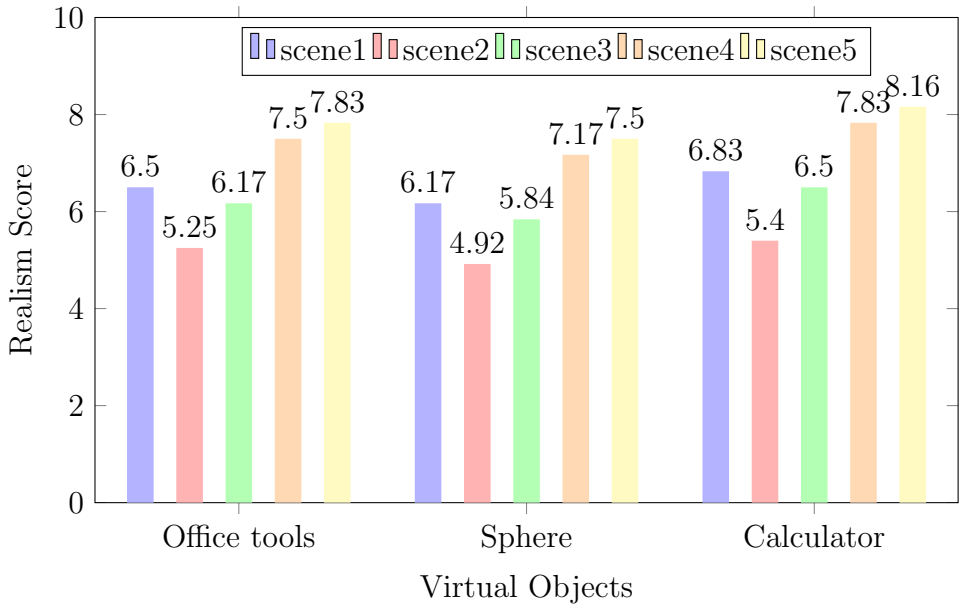

Figure 8.10. Evaluating the realism of the virtual objects based on the scene environment and lighting condition in the local sampling vs global cube map experiment [4].

8.5 Polarization Experiment

Several environments, lighting conditions, and locations are tested through the evaluation process in order to ensure that our system presented in Figure 8.11, as promised, has a visually coherent augmented reality scene. Some examples of the results are presented in Figure 8.12.

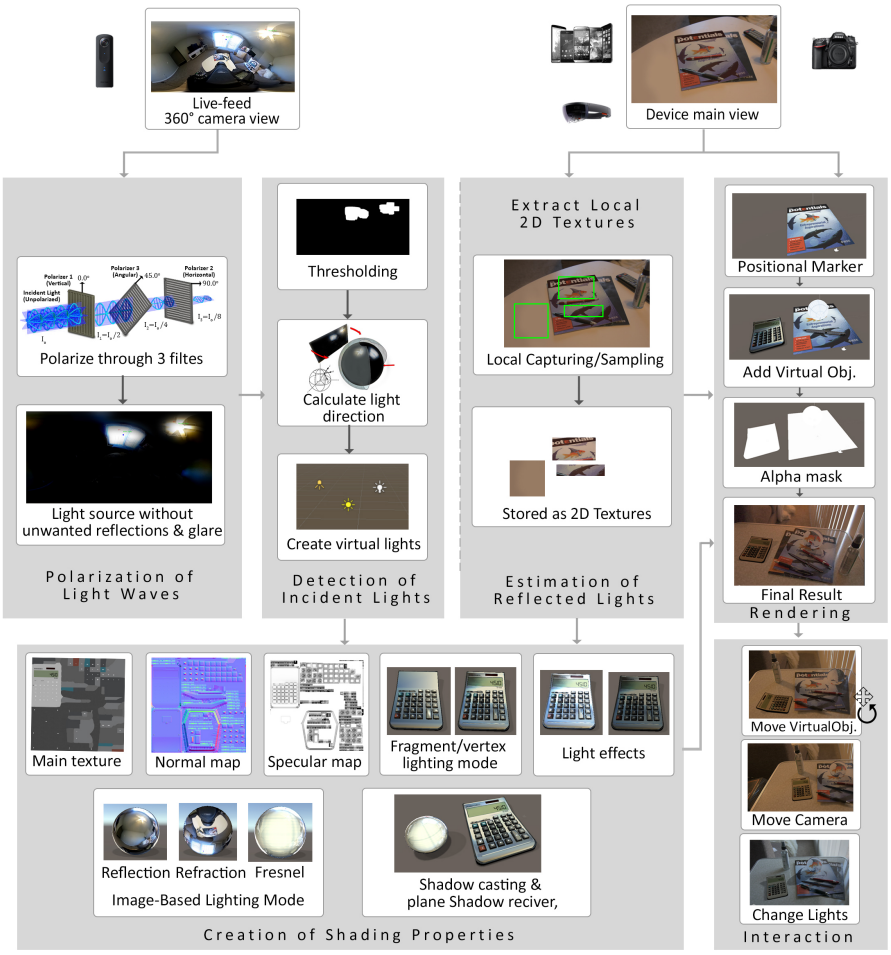

Figure 8.11. An overview of the polarization experiment system components: Polarization of 360° Live-feed, detection of incident light, simulation of reflected light, and creation shading property followed by rendering and interaction [3], for clear diagram and methods see Chapter 3

8.5.1 Polarized Direct Illumination Evaluation

The shadow cast from the virtual objects is an indication of the incident light location when it is compared with the shadow cast from the real object, as stated in Section 8.4.1. Therefore, the basic approach of calculating the direction of the sunlight is used in this step of the evaluation process. The real object's shadow s and height h provides the required inputs to calculate the angle θ that indicates

Figure 8.12. Polarization experiment results with different lighting conditions and locations where each case has: (1) original 360° view "the red circles indicate the locations of unwanted reflections and glares mistaken with light sources", (2) the view after polarization, (3) threshold with minimal requirements, (4) the final image of the scene [3].

the location of the light source as $\theta = \tan^{-1}\frac{h}{s}$. The angle of direct polarized light is calculated manually in the physical scene. The measured angle listed then aligns with the corresponding detected angle ϕ that is obtained through the methods explained in previous sections. Table 8.4 shows the small amount of error as calculated in degrees between both angles. The error statistics is averaged using the root-mean-square-error $RMSE = \sqrt{\frac{\sum_{i=1}^{6} Error_i^2}{6}} = 1.567$. This resulting average is slightly improved compared to the older version of our system which did not include polarization [1,2].

Table 8.4.
Polarization experiment: the measured angle θ compared with detected angle ϕ in degrees and the corresponding errors [3].

Scene No.	Shadow length (s)	Measured (θ°)	Detected (ϕ°)	Error
1	26	-169.13	-168.05	-1.08
2	4.3	229.15	230.06	-0.91
3	14	19.61	18.55	1.06
4	3	-120.97	-120.04	-0.93
5	16	17.32	15.2	2.12
6	7	35.52	36.45	-0.93

8.5.2 Performance Evaluation After Polarization

Global illumination has a major influence on the performance cost of an AR/MR system. Our previous work investigated several methods that can calculate the reflected lights with minimal cost. These methods include cube map (CM) [2], 2D Textures Sub-Sampling (2D) [1], and polarization (PZ) that is also compared with the methods presented in [61,74,124] (GR12, GR14, and GR15, respectively; see Table 8.5). The categories of interest include: (FPS) number of frame per second where our system has achieved the required range of (30fps/60fps) for real-time execution, (Update) describes the process of the entire pipeline in [ms], (Input) is about the data captured through the main camera and the 360° view. (Tracking) is involved with geometry reconstruction and 6-DOF camera tracking, the (Surfaces) involves the surfaces extracted by the renderer including occlusion computation. (Rendering) the time used for the virtual objects rasterization, differential rendering, and the composition of AR scenes.

Table 8.5.
Performance evaluation compared to our previous methods: cube map (CM), 2D textures sub-sampling(2D), with the current method polarization (PZ), also compared against methods from other research (GR12, GR14, GR15) [3].

Operation	Related Work			Ours		
	GR12	GR14	GR15	CM	2D	PZ
FPS[1/s]	5.8	12.29	22.46	6	45	50
Update[ms]	172.4	81.36	44.53	173.4	35.9	32.4
Input[ms]	7.3	7.44	7.02	0.04	0.05	0.06
Tracking[ms]	11.1	10.24	10.64	2.31	1.9	1.10
Surfaces[ms]	6.69	6.69	12.6	4.21	4.42	4.56
Rendering[ms]	0.92	0.98	0.87	0.51	0.43	0.32

8.6 Real-scene Geometry Experiment

Indoor and outdoor environments with different lighting conditions are examined to evaluate the system illustrated in Figure 8.13 from several perspectives. The general outputs seem promising for an online AR applications—See Figures 8.14, 8.15.

8.6.1 Geometric Evaluation

A geometric comparison to the ground truth model is the approach used to evaluate the scene reconstruction. Although the final model is not built to create a well-reconstructed mesh, an accuracy evaluation of the feature points is conducted on a small-scale. The ground truth model is denoted as T and the reconstructed result to be evaluated as R. The goal is to assess the accuracy of R which means how close R is to T. For the purposes of this evaluation, we assume that R is itself a triangle mesh. The reconstruction accuracy is measured by computing the distance between

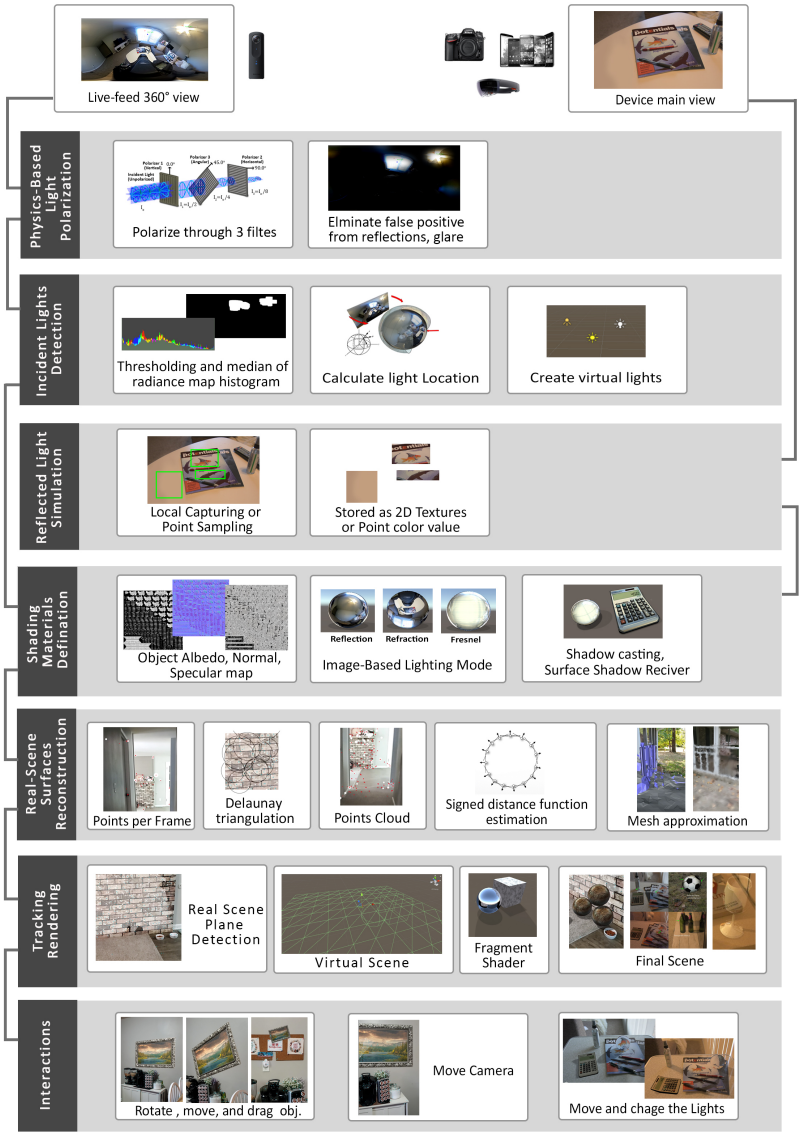

Figure 8.13. An overview of the geometric reconstruction experiment system components: physics-based light polarization, incident light detection, reflected light simulation, shading materials definition, and real-scene geometric reconstruction followed by tracking, rendering, and interaction [5], for clear diagram and methods see Chapter 3.

Figure 8.14. Geometric reconstruction experiment: output with different scenes (a) plane detection without shading materials, (b) Plane detection, and shading properties, (c) plane detection with a sense of lights where objects in the shadow, (d) light and shadow changing on the objects, (e) vertical plane detection also with the ability to place objects on them. virtual objects [cubes, picture frame] [5].

Figure 8.15. Geometric reconstruction experiment: output with different scenes (A) a virtual rock that depict the sunlight condition, (B) the virtual rock from a different angle where the shading is changing based on the perspective, (C) virtual tree branch that captures similar colorization to the real objects around it, (D) a whole virtual tree that shows a sense of occlusion behind the real trees [5].

the points in R and the nearest points on T which can be on its boundary or on a distant part of the mesh. In theory, R is considered a surface that should have measures applied entailing integration over R, but in practice, the vertices of R are sampled for the evaluation.

When the nearest valid points on T are determined from R, the signed distances are computed whether the reconstruction is over- or under-estimated the true geometry. The sign of each distance is set to be equal to the sign of the dot product between the normal facing outward at the nearest point on T and the vector from that point to the inquired point on R. The distribution of signed distances from the vertices of R to T can be visualized to compute statistics comparing the accuracy of the reconstruction process. The root-mean-square-error (RMSE), mean, median, and standard deviation (STDEV) are computed against the ground truth. The results are shown in Table 8.6.

Figure 8.16. Camera frames from the datasets [5].

8.6.2 Mobile Performance Evaluation

The global illumination in a dynamic scene in real-time impacts the performance cost of the entire system. The previous versions of the system are investigated to

Table 8.6.
Evaluating the geometric reconstruction experiment system using the datasets in Figure 8.16 [5].

Statistic	Exploring points	
	dataset1	dataset2
RMSE	0.80901	2.7302
Mean	0.80250	2.3754
STDEV	0.09420	0.3425
Median	0.80510	2.5920

evaluate the total cost of each version. Some features of the system were removed or replaced with other methods that provide the same feature to reduce the performance cost. The system versions include cube map (CM) [2], 2D Textures Sub-Sampling (2D) [1], physics-based polarization (PZ) [3], and geometric reconstruction of the physical scene(GS) which are also compared with the models presented in other related work: GR12 [74], GR14 [124], GR15 [61]. Table 8.7 presents the criteria used to evaluate the system performance such as **(FPS)** which refer to the number of frame per second where the real-time range of (30fps/60fps) is achieved in the latest versions of the system, **(Update)** reflects the time spent for the entire pipeline process in [ms], **(Input)** is the data captured from the main AR view and the 360° view. While **(Tracking)** is the time taken for the geometry reconstruction or 6-DOF camera tracking, the **(Surfaces)** is the renderer time to extract including occlusion computation. Lastly, **(Rendering)** is for objects rasterization, rendering procedure, and composition of the final scenes.

8.7 Optimization Experiment

The system is evaluated through several indoor and outdoor scenes with different lighting conditions to ensure that the optimization in our system are valid and provide

Table 8.7.
Performance evaluation comparing the previous versions of the system: cube map (CM) [2], 2D textures sub-sampling (2D) [1], physics-based polarization (PZ) [3], and geometric reconstruction of the physical scene (GS), which also compared against methods from other research (GR12 [74], GR14 [124], GR15 [61]).

Operation	Related Work			Ours			
	GR12	GR14	GR15	CM	2D	PZ	GS
FPS[1/s]	5.8	12.29	22.46	6	45	50	55
Update[ms]	172.4	81.36	44.53	173.4	35.9	32.4	30.0
Input[ms]	7.3	7.44	7.02	0.04	0.05	0.06	0.07
Tracking[ms]	11.1	10.24	10.64	2.31	1.9	1.10	1.00
Surfaces[ms]	6.69	6.69	12.6	4.21	4.42	4.56	5.55
Rendering[ms]	0.92	0.98	0.87	0.51	0.43	0.32	0.31

more accurate outcomes from the older versions. Some scenes are presented in Figure 8.17 as an example. Although the total number of virtual objects in the scene is not a crucial factor for the system evaluation, the experiment includes a calculator, a pen, a statue, a glass, and a transparent sphere.

8.7.1 Optimized Illumination Evaluation

The real objects in the scene act as the ground truth of this experiment, the shadow casting from the real objects is the system cue that the detection method of direct illumination is working properly, as mentioned in Section 8.4.1. The shadow of the virtual object is compared with the ground truth shadow in several scenes. Thus, the approach used to calculate the angle of sunlight is utilized here to measure the direction of incident light on the real objects. The measured angle α° with comparison to the detected angle β° by the system is used to calculate the error margin of the developed method. The height h and shadow length s of the real

object are the required inputs to calculate the angle α° as $\alpha^\circ = \tan^{-1}\frac{h}{s}$. Figure 8.6 demonstrates how the angle of incident light α° is calculated manually in the physical scene. The corresponding angle β° which is computed by the method in section 8.3 is listed aligned to the measured angle α°. As it can be seen in Table 8.8 the error statistics for the six scenes in Figure 8.17 are averaged using the root-mean-square-error $RMSE = \sqrt{\frac{\sum_{i=1}^{6} Error_i^2}{6}}$. The resulting mean from the optimized computations is improved slightly compared to the older version of the system which indicates a better estimation of the incident light direction.

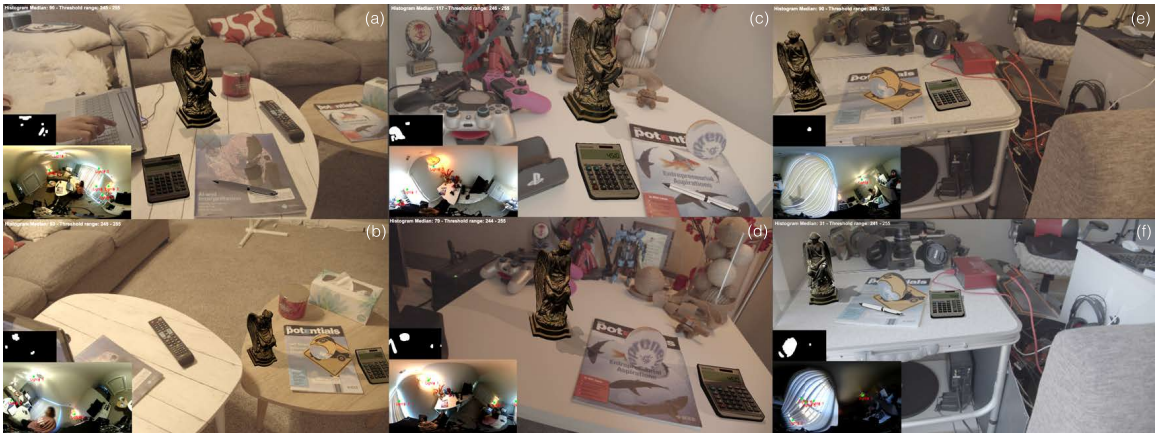

Figure 8.17. Optimization experiment results with different lighting conditions and locations where each case has: (1) original 360° view before polarization (2) threshold after polarization, (3) the final image of a rendered scene; Lighting conditions: (a)(b) the first location with different camera poses (field of view), lights position, and virtual objects placement, (c)(d) second location with two main lights in (c) and three lights in (d) where the sunlight intensity coming from the window is differ based on the blinds movement, (e)(f) the third location which clearly shows the effect of artificial lights compare to the sunlight; *virtual objects are angel statue, transparent sphere, calculator, and pen.*

Table 8.8.
Optimization experiment: the measured angle α and the corresponding detected angle β are listed for several scenes with different lighting conditions to find the error margin of the incident light direction.

Scene	Shadow s	Older Version			Current Optimization		
		Measured α°	Detected β°	Error	Measured α°	Detected β°	Error
A	15.02	-149.4	-150.2	0.8	-149.4	-149.9	0.5
B	16.48	9.7	10.6	-0.9	9.7	9.1	0.6
C	5.45	-71.5	-69.2	-2.3	-71.5	-73.0	1.5
D	6.82	-140.3	-139.5	-0.8	-140.3	-141.04	0.47
E	8.22	105.4	104.2	1.2	105.4	106.2	-0.8
F	11.32	-147.8	-146.5	-1.3	-147.8	-146.1	-1.7
root-mean-square-error (RMSE)				1.751			1.101

8.7.2 Performance Evaluation After Optimization

The performance cost is influenced by the global illumination that consists of direct and indirect illumination. In order to evaluate the performance progress, the previous methods developed in the system must be addressed which include: **(CM)** create cube map textures for indirect illumination, **(2D)** capture 2D textures for reflected light Sub-Sampling, and **(PZ)** polarizes the direct illumination for a preferable threshold. The current optimization **(OP)** merges (2D) and (PZ) features but also improved the computations of the direct illumination based on the point of view. The performance of our methods are listed to show a slight improvement in the system and also are compared with the methods presented in [61, 74, 124], see Table. 8.9.

The comparison criteria of interest include the following: (FPS) is the number of frame per second where our system has achieved the required range of (30fps/60fps) for real-time execution, (Update) refers to the entire pipeline process in [ms], (Input)

Table 8.9.
Performance evaluation compared to our previous methods: cube mapping (CM), 2D textures sub-sampling(2D), polarization (PZ), and optimization (OP) also compared against methods from other research (GR12, GR14, GR15) [61, 74, 124].

Operation	Related Work			Ours			
	GR12	GR14	GR15	CM	2D	PZ	OP
FPS[1/s]	5.8	12.29	22.46	6	45	50	52
Update[ms]	172.4	81.36	44.53	173.4	35.9	32.4	30.9
Input[ms]	7.3	7.44	7.02	0.04	0.05	0.06	0.07
Tracking[ms]	11.1	10.24	10.64	2.31	1.9	1.10	1.05
Surfaces[ms]	6.69	6.69	12.6	4.21	4.42	4.56	4.61
Rendering[ms]	0.92	0.98	0.87	0.51	0.43	0.32	0.25

describes the time taken to capture the data from the main camera and the 360° view. While (Tracking) is for the geometry reconstruction and 6-DOF camera tracking time, and (Surfaces) refer to the extraction time of all surfaces by the renderer including occlusion computation. (Rendering) the time used for the virtual objects rasterization, differential rendering, and the composition of AR scenes per frame.

Although it is not ideal to compare a Mixed Reality (MR) system with the AR system due to the slight differences, the system is compared with the work presented in [108] where both systems utilized 360° camera, see Table 8.10. As known, the MR augments or "records" the real world inside the user environment which by default makes the direct illumination detection more convenient where all the information about the final scene is available in the user's virtual world. A similar number of virtual objects are added in each scene for a fair comparison.

Table 8.10.
Performance evaluation between MR360 [108] system and our system where both employed the 360° camera as data input to detect the incident light.

Trial No.	MR360			Ours		
	1	2	3	1	2	3
LightDetection[ms]	22.1	23.2	24.4	25.3	27.2	29.7
Rendering[ms]	11.1	12.9	11.1	0.32	0.43	0.51

8.7.3 Collective User-Feedback Evaluation

A brief user-feedback study was conducted through an online survey that contains several final scenes of the system results. The 39 users were identified as college students. The collected feedback which was asking directly about the overall realism in scenes with several lighting conditions and locations (S1, S2, S3, S4) was compared with the feedback received from our earlier work. Figure 8.18 presents the variation of user feedback based on the lighting condition, object materials, and locations. The weighted mean of the results falls in the range $(70\% - 80\%)$ which is above the average. We noticed that the difference between the original work of local sampling and polarization has a slight impact on the final scene but provides a significant impact on the incident light detection process where the unwanted reflections and glares are reduced dramatically. The slight impact of polarization was mainly observed on the final color of the direct illumination which provided better perception for the users in the final image of the current work.

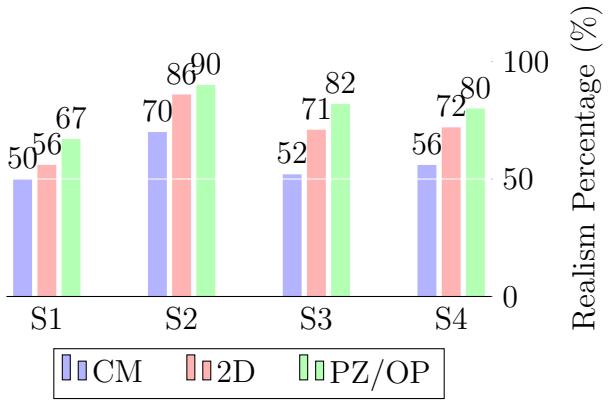

Figure 8.18. User-feedback after evaluating virtual objects realism and the overall perception of the optimization experiment.

8.8 Implementation Requirements

8.8.1 Software Development

The software is developed in Unity 3D engine using C# language and takes into consideration the possible exporting of the software to several platforms: PC, Android, IOS, or Universal Windows Platform such as AR headsets (Microsoft HoloLens). Following libraries are used as necessary:

- **OpenCV**, The Open source Computer Vision library, is used to run a quick test for the standard computer vision functions and methods.
- **OpenGL**, Open Graphics Library is used to render 2D and 3D vector graphics. The application programming interface (API) is also used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.
- **Shading Language**, Also known as GLSL or Glslang, which is a standard shading language used with OpenGL. It unifies the process of vertex and fragment in a single set of instructions that allows more conditional loops and branches. It was used to defined the shading properties of the virtual scene surfaces and objects.

- **Adcock.Parallel**, A lightweight implementation of a small subset of Microsoft's Parallel Extensions for .Net 3.5/4.0 that can be used with C# also.
- **Vuforia**, is an augmented reality software development kit (SDK) to create an augmented reality application on mobile devices. It used with computer vision technology to track and recognize planar images or simple 3D objects as AR markers.
- **ARCore**, also an augmented reality tool that has a point cloud and plane detection feature that can support the 3D reconstruction of the real-scene geometry.

8.8.2 Hardware Description

The following hardware devices are used through this research.

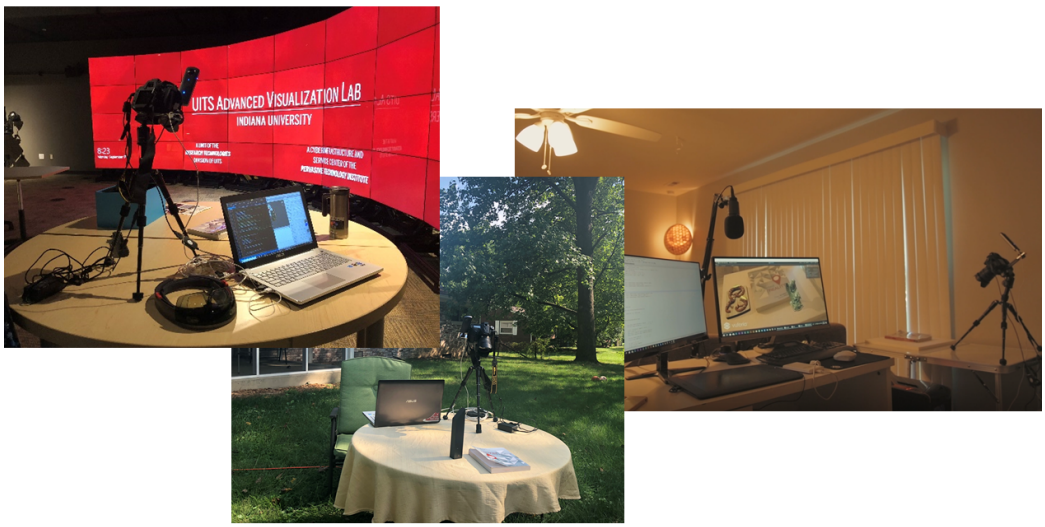

Figure 8.19. Workstation setup in different lighting environments (indoor, outdoor) for experiments validation.

- **PC, Laptop**. Intel®Core™ i7-3930k CPU @ 3.20GHz 3201 MHz, six core(s), 64.0 GB RAM, and NVIDIA GeForce GTX 970 GPU.

- **AR Main Camera**, for data input device a DSLR Nikon D7200 is used to capture the real scene and the AR marker to include the virtual scene.
- **360° Camera**, a live-feed RICOH THETA S 360° is dedicated for reading the environment maps and lighting conditions.
- **Polarization**, Three polarizing sheets (filters) are used to reduce reflections and glare.
- **RGB-D sensor**, A Microsoft Kinect is considered to read the depth data for additional virtual to real interactions.
- **Galaxy note 10**, For Mobile implementation, a device with an Android system that accepts the minimum requirement of ARCore API level is used to capture the final result.

9 CONCLUSION AND FUTURE WORK

9.1 Conclusion

Photo-realistic rendering of virtual objects into the final scene that are indistinguishable from the real world in a dynamic environment has been one of the major problems to be solved in computer graphics and computer vision. Correct illumination information extracted from the physical scene can be used to render augmented objects in a scene to provide realism and proper integration of virtual objects in the real scene. An AR scene that integrates the illumination model requires addressing several aspects in each frame [31,39]. Dynamic illumination is a problem that studies light transportation from the direct light sources and the light reflected from other objects. Usually, the real-time algorithms for illumination are developed to allow fast computation with a minimum amount of error. This thesis investigated the visual coherence problem in augmented reality by developing a system that provides real-time dynamic illumination for interactive augmented reality based on the virtual objects appearance in association with the real objects and other criteria. The proposed algorithm **detects the incident light** (direct illumination) in the physical scene through a live-feed 360° camera that is instrumented on any augmented reality (AR) device (e.g., a handheld mobile device, a head-mounted display, or webcam camera) to capture the entire environment map using thresholding approach based on topological structural analysis of 2D images.

The **physics-based polarization** properties of light are utilized to reduce any false positive lights from white surfaces, reflections, and glare in detecting the incident lights resulting in fewer errors in lighting conditions. The system also **simulates the reflected light** (indirect illumination) from surfaces while rendering the virtual objects by implementing two separate sub-methods which result in totally different

outcomes for the internal interaction among the real and virtual objects. The system computes various **shading properties** for each virtual object in every frame. These properties include material textures, shadows, specular and diffuse illumination, reflection, refraction, and Fresnel. Moreover, an improvement of the physics-based illumination through polarization is presented which allows more accurate light source estimation in AR. In order to provide a realistic sense of the physical environment, the system **geometric properties of real-scenes** including plane detection, 3D surfaces reconstruction and simple meshing which incorporate the physical scene into the final scene for realistic and real-time interaction between the virtual and real objects without the use of image-based markers.

The interaction with the virtual objects, rendering, and registration methods must work in real-time. The challenge of fully estimating the direction of incident light with the simulation of reflected light, in addition to including shading properties can be costly in terms of computational time. This work evaluates and improves multiple tracks to reach an effective illumination model for AR systems and applications. In addition, optimization strategies were developed to overcome some design and implementation challenges encountered in our system. For instance, the enhancement of incident light detection under the consideration of 360° camera view distortion while reading the environment map; and prioritization of the lights when a single extended light is detected as multiple lights due to sampling of spherical light fields. Augmented reality systems and applications are still under investigation and development and our system also has the potential for improvement.

9.2 Future Work

The following is a list of some possible future extensions of our work.

- **Light estimation based on the device's pose and 3D geometric features.**

  Detection of the incident light is an essential feature to obtain a perfect light for

any environment. The current method studies the light based on color brightness and topological structural analysis of 2D images. Thus, An inclusion of 3D geometric features with robust simultaneous localization and mapping, or SLAM will increase the photo-realistic perception for our final scene.

- **Adopt additional physical features and sensors.**
  The effect of light polarization was so satisfactory that it makes us think about adopting additional physics-based features or sensors such as light detection sensor or infrared sensor and how they can be engineered into our system in order to reduce light computation time and reduce the overall cost of performance.

- **Support physics shading properties**
  Although our system supports reflection, refraction, and shadowing, other shading properties would increase the photo-realism perception of the final scene. The ability to simulate underwater caustics and refractive caustics of glass would provide a more realistic output. Some algorithms use aggressive assumptions about good candidates for caustics using backward Monte Carlo ray tracing which provides low computational cost. The result usually is not physically correct but mimics the real caustic's look and behavior with convincing effect.

- **Supporting dynamic physical tracking for real objects**
  Although the system supports dynamic scenes where the lights, camera, and AR devices can move freely, the virtual objects can also move anywhere in the scene. However, tracking of moving real objects such as people in the scene needs some improvement which can include human-gesture tracking and other features.

- **Implement the system entirely on GPU for low-cost performance**
  A complete GPU implementation of this system gives even better visual quality and improves the performance cost in general. A high-level shading language will allow moving all these computations to the GPU. Calculating some physics

and shading features per-pixel instead of per-vertex enhances the overall visual coherence with higher quality and decouples the effect of geometric complexity.

REFERENCES

[1] A'aeshah Alhakamy and Mihran Tuceryan. Ar360: Dynamic illumination for augmented reality with real-time interaction. In *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pages 170–174, March 2019.

[2] A'aeshah Alhakamy and Mihran Tuceryan. Cubemap360: Interactive global illumination for augmented reality in dynamic environment. In *2019 SoutheastCon*, pages 1–8. IEEE, IEEE Press, 2019.

[3] A'aeshah Alhakamy and Mihran Tuceryan. Polarization-based illumination detection for coherent augmented reality scene rendering in dynamic environments. In *Proceedings of the 36th Computer Graphics International (CGI'19) in cooperation with ACM SIGGRAPH and Eurographics, chapter in (Advances in Computer Graphics)*, pages 3–14, Cham, 2019. Springer International Publishing.

[4] A'aeshah Alhakamy and Mihran Tuceryan. An empirical evaluation of the performance of real-time illumination approaches: Realistic scenes in augmented reality. In *Augmented Reality, Virtual Reality, and Computer Graphics*, pages 179–195, Cham, 2019. Springer International Publishing.

[5] A'aeshah Alhakamy and Mihran Tuceryan. Physical environment reconstruction beyond light polarization for coherent augmented reality scene on mobile devices. In *Transactions on Computational Science XXXVII: Special Issue on Computer Graphics*, pages 19–38, Berlin, Heidelberg, 2020. Springer Berlin Heidelberg.

[6] A'aeshah Alhakamy and Mihran Tuceryan. Real-time illumination and visual coherence for photorealistic augmented/mixed reality. *ACM Comput. Surv.*, 53(3), May 2020.

[7] Abeer Alsaiari, Ridhi Rustagi, A'aeshah Alhakamy, Manu Mathew Thomas, Angus G Forbes, et al. Image denoising using a generative adversarial network. In *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pages 126–132. IEEE, 2019.

[8] Francesco Cafaro, Milka Trajkova, and A'aeshah Alhakamy. Designing embodied interactions for informal learning: Two open research challenges. In *Proceedings of the 8th ACM International Symposium on Pervasive Displays*, PerDis '19, pages 28:1–28:2, New York, NY, USA, 2019. ACM.

[9] Milka Trajkova, A'aeshah Alhakamy, Francesco Cafaro, Rashmi Mallappa, and Sreekanth R. Kankara. Move your body: Engaging museum visitors with human-data interaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[10] A'aeshah Alhakamy, Milka Trajkova, Francesco Cafaro, Rashmi Mallappa, Sreekanth R. Kankara, and Vedak Sanika. Design strategies and optimizations for human-data interaction systems in museums. In *Proceedings of the 2020 International Conference on Advanced Learning Technologies (ICALT '20)*. IEEE, 2020.

[11] Dieter Schmalstieg and Tobias Hollerer. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[12] Peter Kán and Hannes Kaufmann. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 99–108. IEEE, 2012.

[13] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the Game Developers Conference*, volume 56, page 4, 2003.

[14] James T Kajiya. The rendering equation. In *ACM SIGGRAPH computer graphics*, volume 20, pages 143–150. ACM, 1986.

[15] David S Immel, Michael F Cohen, and Donald P Greenberg. A radiosity method for non-diffuse environments. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 133–142. ACM, 1986.

[16] Philip Dutre, Philippe Bekaert, and Kavita Bala. *Advanced global illumination*. AK Peters/CRC Press, 2006.

[17] Claudia Doppioslash. *Physically Based Shader Development for Unity 2017: Develop Custom Lighting Systems*. Apress, 2017.

[18] Ravi Ramamoorthi. *A signal-processing framework for forward and inverse rendering*. Citeseer, 2002.

[19] Chayan Vinayak Goswami. Shader development in unity. Lecture series, shaderdev.com, 2016.

[20] Turner Whitted. An improved illumination model for shaded display. In *ACM SIGGRAPH Computer Graphics*, volume 13, page 14. ACM, 1979.

[21] Francesco Banterle, Patrick Ledda, Kurt Debattista, and Alan Chalmers. Inverse tone mapping. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 349–356. ACM, 2006.

[22] Eric Lafortune. Mathematical models and monte carlo algorithms for physically based rendering. *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven*, 20:74–79, 1996.

[23] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[24] Eric P Lafortune and Yves D Willems. *Bi-directional path tracing*. KU Leuven, 1993.

[25] Frank Suykens and Yves D Willems. *Adaptive filtering for progressive monte carlo image rendering*. University of West Bohemia, 2000.

[26] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM Transactions on Graphics (TOG)*, volume 27, page 33. ACM, 2008.

[27] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive sampling and reconstruction using greedy error minimization. In *ACM Transactions on Graphics (TOG)*, volume 30, page 159. ACM, 2011.

[28] Anthony Pajot, Loïc Barthe, Mathias Paulin, and Pierre Poulin. Combinatorial bidirectional path-tracing for efficient hybrid cpu/gpu rendering. In *Computer Graphics Forum*, volume 30, pages 315–324. Wiley Online Library, 2011.

[29] Yusuke Tokuyoshi and Shinji Ogaki. Real-time bidirectional path tracing via rasterization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 183–190. ACM, 2012.

[30] Anton S Kaplanyan and Carsten Dachsbacher. Path space regularization for holistic and robust light transport. In *Computer Graphics Forum*, volume 32, pages 63–72. Wiley Online Library, 2013.

[31] Peter Kan. *High-Quality Real-Time Global Illumination in Augmented Reality*. PhD thesis, Kan, 2014.

[32] Alain Fournier, Atjeng S Gunawan, and Chris Romanzin. Common illumination between real and computer generated scenes. In *Graphics Interface*, pages 254–254. CANADIAN INFORMATION PROCESSING SOCIETY, 1993.

[33] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH'98 the 25th Annual Conference on Computer Graphics and Interactive Techniques*, volume 10, pages 189–198. ACM, 1998.

[34] Dieter Schmalstieg and Tobias Hollerer. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[35] Thorsten Grosch. Differential photon mapping-consistent augmentation of photographs with correction of all light paths. In *Eurographics (Short Presentations)*, pages 53–56, 2005.

[36] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[37] Thorsten Grosch, Tobias Eble, and Stefan Mueller. Consistent interactive augmentation of live camera images with correct near-field illumination. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 125–132. ACM, 2007.

[38] Martin Knecht, Christoph Traxler, Oliver Mattausch, Werner Purgathofer, and Michael Wimmer. Differential instant radiosity for mixed reality. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 99–107. IEEE, 2010.

[39] Peter Kán and Hannes Kaufmann. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99–108. IEEE, 2012.

[40] Peter Kán and Hannes Kaufmann. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 133–141. IEEE, 2013.

[41] Masayuki Kanbara, Naokazu Yokoya, and Haruo Takemura. *Real-time estimation of light source environment for photorealistic augmented reality.* IEEE, 2004.

[42] Philipp Lensing and Wolfgang Broll. Instant indirect illumination for dynamic mixed reality scenes. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 109–118. IEEE, 2012.

[43] Kai Rohmer and Thorsten Grosch. Tiled frustum culling for differential rendering on mobile devices. In *2015 IEEE International Symposium on Mixed and Augmented Reality*, pages 37–42. IEEE, 2015.

[44] Kai Rohmer, Johannes Jendersie, and Thorsten Grosch. Natural environment illumination: coherent interactive augmented reality for mobile and non-mobile devices. *IEEE transactions on visualization and computer graphics*, 23(11):2474–2484, 2017.

[45] Rafael Monroy, Matis Hudon, and Aljosa Smolic. Dynamic environment mapping for augmented reality applications on mobile devices. In *Vision, Modeling & Visualization, VMV 2018, Stuttgart, Germany, October 10-12, 2018*, pages 21–28, 2018.

[46] C Brosseau. *Fundamentals of polarized light, a statistical optics approach.* John Wiley and Sons, 1998.

[47] Thomas Kreis. *Handbook of holographic interferometry: optical and digital methods.* John Wiley & Sons, 2006.

[48] Bleeker Johan and Verbunt Frank. *Lecture Notes on Observational Astrophysics II.* SIU, 2006.

[49] Douglas B Murphy. *Fundamentals of light microscopy and electronic imaging.* John Wiley & Sons, 2002.

[50] Inoué Shinya and Spring Kenneth K. *Video Microscopy: The Fundamentals.* Springer, 1997.

[51] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH97*, page 369378. ACM, 1997.

[52] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH'98 the 25th Annual Conference on Computer Graphics and Interactive Techniques*, volume 10, pages 189–198. ACM, 1998.

[53] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 527–536. ACM, 2002.

[54] Rohmer Kai and Grosch Thorsten. Tiled frustum culling for differential rendering on mobile devices. In *2015 IEEE International Symposium on Mixed and Augmented Reality*, pages 37–42, Sep. 2015.

[55] Petr Vévoda and Jaroslav Křivánek. Adaptive direct illumination sampling. In *SIGGRAPH ASIA 2016 Posters*, SA '16, pages 43:1–43:2, New York, NY, USA, 2016. ACM.

[56] Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):174, 2016.

[57] Kenji Hara, Ko Nishino, and Katsushi Ikeuchi. Determining reflectance and light position from a single image without distant illumination assumption. In *null*, page 560. IEEE, 2003.

[58] Thomas Richter-Trummer, Denis Kalkofen, Jinwoo Park, and Dieter Schmalstieg. Instant mixed reality lighting from casual scanning. In *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, pages 27–36. IEEE, 2016.

[59] Kai Rohmer, Wolfgang Büschel, Raimund Dachselt, and Thorsten Grosch. Interactive near-field illumination for photorealistic augmented reality on mobile devices. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 29–38. IEEE, 2014.

[60] D. Nowrouzezahrai, S. Geiger, K. Mitchell, R. Sumner, W. Jarosz, and M. Gross. Light factorization for mixed-frequency shadows in augmented reality. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 173–179, Oct 2011.

[61] Lukas Gruber, Jonathan Ventura, and Dieter Schmalstieg. Image-space illumination for augmented reality in dynamic environments. In *Virtual Reality (VR), 2015 IEEE*, pages 127–134. IEEE, 2015.

[62] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500. ACM, 2001.

[63] M. Meilland, C. Barat, and A. Comport. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 143–152, Oct 2013.

[64] K. Rohmer, J. Jendersie, and T. Grosch. Natural environment illumination: Coherent interactive augmented reality for mobile and non-mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 23(11):2474–2484, Nov 2017.

[65] T. A. Franke. Delta light propagation volumes for mixed reality. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 125–132, Oct 2013.

[66] Eric Heitz, Stephen Hill, and Morgan McGuire. Combining analytic direct illumination and stochastic shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '18, pages 2:1–2:11, New York, NY, USA, 2018. ACM.

[67] Weiqi Shi, Zeyu Wang, Metin Sezgin, Julie Dorsey, and Holly Rushmeier. Material design in augmented reality with in-situ visual feedback. In *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations*, EGSR '17, pages 93–103, Goslar Germany, Germany, 2017. Eurographics Association.

[68] Alexander Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56. ACM Press/Addison-Wesley Publishing Co., 1997.

[69] Naveen Dachuri, Seung Man Kim, and Kwan H Lee. Estimation of few light sources from environment maps for fast realistic rendering. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pages 265–266. ACM, 2005.

[70] Martin Knecht, Christoph Traxler, Oliver Mattausch, Werner Purgathofer, and Michael Wimmer. Differential instant radiosity for mixed reality. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 99–107. IEEE, 2010.

[71] Katrien Jacobs and Céline Loscos. Classification of illumination methods for mixed reality. In *Computer Graphics Forum*, volume 25, pages 29–51. Wiley Online Library, 2006.

[72] Peter Kán and Hannes Kaufmann. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 99–108. IEEE, 2012.

[73] Soham Uday Mehta, Kihwan Kim, Dawid Pajak, Kari Pulli, Jan Kautz, and Ravi Ramamoorthi. Filtering environment illumination for interactive physically-based rendering in mixed reality. In *Eurographics Symposium on Rendering*, volume 7, 2015.

[74] Lukas Gruber, Thomas Richter-Trummer, and Dieter Schmalstieg. Real-time photometric registration from arbitrary geometry. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 119–128. IEEE, 2012.

[75] Kai Rohmer, Wolfgang Büschel, Raimund Dachselt, and Thorsten Grosch. Interactive near-field illumination for photorealistic augmented reality on mobile devices. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 29–38. IEEE, 2014.

[76] Tobias Schwandt and Wolfgang Broll. A single camera image based approach for glossy reflections in mixed reality applications. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 37–43. IEEE, 2016.

[77] Kusuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. Photorealistic rendering for augmented reality using environment illumination. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 208–216. IEEE, 2003.

[78] Joel Kronander, Francesco Banterle, Andrew Gardner, Ehsan Miandji, and Jonas Unger. Photorealistic rendering of mixed reality scenes. In *Computer Graphics Forum*, volume 34, pages 643–665. Wiley Online Library, 2015.

[79] Tobias Alexander Franke. Delta light propagation volumes for mixed reality. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 125–132. IEEE, 2013.

[80] Tobias Alexander Franke. Near-field illumination for mixed reality with delta radiance fields. In *ACM SIGGRAPH 2013 Posters*, pages 1–1. ACM, 2013.

[81] K Ahlers, Chris Crampton, Douglas Greer, Eric Rose, and M Tuceryan. Augmented vision: A technical introduction to the grasp 1.2 system. Technical report, Technical Report ECRC-94-14, ECRC, Munich, Germany, 1994.

[82] Esha Nerurkar, Simon Lynen, and Sheng Zhao. System and method for concurrent odometry and mapping, November 23 2017. US Patent App. 15/595,617.

[83] D Rettenmund, M Fehr, S Cavegn, and S Nebiker. Accurate visual localization in outdoor and indoor environments exploiting 3d image spaces as spatial reference. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(1), 2018.

[84] Thomas Schneider, Mingyang Li, Michael Burri, Juan Nieto, Roland Siegwart, and Igor Gilitschenski. Visual-inertial self-calibration on informative motion segments. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6487–6494. IEEE, 2017.

[85] E. Parisotto, D. S. Chaplot, J. Zhang, and R. Salakhutdinov. Global pose estimation with an attention-based recurrent network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 350–35009, June 2018.

[86] Boyan Mladenov, Lorenzo Damiani, Pietro Giribone, and Roberto Revetria. A short review of the sdks and wearable devices to be used for ar application for industrial working environment. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 23–25, 2018.

[87] Ye Dai and Wenjun Hou. Research on configuration arrangement of spatial interface in mobile phone augmented reality environment. In *International Conference on Human Centered Computing*, pages 48–59. Springer, 2018.

[88] Mihran Tuceryan, Douglas S. Greer, Ross T. Whitaker, David E. Breen, Chris Crampton, Eric Rose, and Klaus H Ahlers. Calibration requirements and procedures for a monitor-based augmented reality system. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):255–273, 1995.

[89] David E Breen, Ross T Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. In *Computer Graphics Forum*, volume 15, pages 11–22. Wiley Online Library, 1996.

[90] C. Zhang. Cufusion2: Accurate and denoised volumetric 3d object reconstruction using depth cameras. *IEEE Access*, 7:49882–49893, 2019.

[91] J. Wang, H. Liu, L. Cong, Z. Xiahou, and L. Wang. Cnn-monofusion: Online monocular dense reconstruction using learned depth from single view. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 57–62, Oct 2018.

[92] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE international conference on Robotics and automation (ICRA)*, pages 1524–1531. IEEE, 2014.

[93] Jan Weingarten and Roland Siegwart. Ekf-based 3d slam for structured environment reconstruction. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3834–3839. IEEE, 2005.

[94] Patrick Henry Winston and Berthold Horn. *The psychology of computer vision*, chapter Obtaining shape from shading information. McGraw-Hill Companies, 1975.

[95] Hua Chen and Lawrence B Wolff. Polarization phase-based method for material classification in computer vision. *International Journal of Computer Vision*, 28(1):73–83, 1998.

[96] JJ van Kessel. Shape from colored structured light and polarization. Master's thesis, Utrecht University, 2013.

[97] Trung Ngo Thanh, Hajime Nagahara, and Rin-ichiro Taniguchi. Shape and light directions from shading and polarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2310–2318, 2015.

[98] Linghao Shen, Yongqiang Zhao, Qunnie Peng, Jonathan Cheung-Wai Chan, and Seong G Kong. An iterative image dehazing method with polarization. *IEEE Transactions on Multimedia*, 2018.

[99] Gary A Atkinson, Thomas J Thornton, Demitri IC Peynado, and Jürgen D Ernst. High-precision polarization measurements and analysis for machine vision applications. In *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pages 1–6. IEEE, 2018.

[100] Unger Jonas, Kronander Joel, Larsson Per, Gustavson Stefan, and Ynnerman Anders. Temporally and spatially varying image based lighting using hdr-video. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5, USA, Sep. 2013. IEEE.

[101] Joel Kronander, Francesco Banterle, Andrew Gardner, Ehsan Miandji, and Jonas Unger. Photorealistic rendering of mixed reality scenes. *Computer Graphics Forum*, 34(2):643–665, 2015.

[102] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic scene inference for 3d object compositing. *ACM Trans. Graph.*, 33(3):32:1–32:15, June 2014.

[103] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *ACM Trans. Graph.*, 36(6):176:1–176:14, November 2017.

[104] Peter Kán, Johannes Unterguggenberger, and Hannes Kaufmann. High-quality consistent illumination in mobile augmented reality by radiance convolution on the gpu. In *International Symposium on Visual Computing*, pages 574–585, Switzerland AG, 2015. Springer, Springer.

[105] Thomas Iorns and Taehyun Rhee. Real-time image based lighting for 360-degree panoramic video. In Fay Huang and Akihiro Sugimoto, editors, *Image and Video Technology – PSIVT 2015 Workshops*, pages 139–151, Cham, 2016. Springer International Publishing.

[106] Tobias Schwandt and Wolfgang Broll. A single camera image based approach for glossy reflections in mixed reality applications. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 37–43, USA, Sep. 2016. IEEE.

[107] Thomas Richter-Trummer, Denis Kalkofen, Jinwoo Park, and Dieter Schmalstieg. Instant mixed reality lighting from casual scanning. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 27–36, USA, Sep. 2016. IEEE.

[108] Taehyun Rhee, Lohit Petikam, Benjamin Allen, and Andrew Chalmers. Mr360: Mixed reality rendering for 360 panoramic videos. *IEEE Transactions on Visualization & Computer Graphics*, 4:1379–1388, 2017.

[109] David Mandl, Kwang Moo Yi, Peter Mohr, Peter M. Roth, Pascal Fua, Vincent Lepetit, Dieter Schmalstieg, and Denis Kalkofen1. Learning lightprobes for mixed reality illumination. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 82–89, USA, Oct 2017. IEEE.

[110] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[111] Insung Ihm, Sanghoon Park, and Rae Kyoung Lee. Rendering of spherical light fields. In *Proceedings The Fifth Pacific Conference on Computer Graphics and Applications*, pages 59–68. IEEE, 1997.

[112] Severin Todt, Christof Rezk-Salama, Andreas Kolb, and K Kuhnert. Fast (spherical) light field rendering with per-pixel depth. *Technical report*, 2007.

[113] Severin Sönke Todt. *Real-time rendering and acquisition of spherical light fields*. Universit at Siegen, 2009.

[114] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012.

[115] Joseph M Brom and Frank Rioux. Polarized light and quantum mechanics: An optical analog of the stern–gerlach experiment. *The Chemical Educator*, 7(4):200–204, 2002.

[116] Michael C Zerner. Semiempirical molecular orbital methods. *Reviews in computational chemistry*, pages 313–365, 1991.

[117] Johann Heinrich Lambert. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. Klett, 1760.

[118] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[119] James F Blinn. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH computer graphics*, volume 11, pages 192–198. ACM, 1977.

[120] Tamal K Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2006.

[121] Tony Chan and Wei Zhu. Level set based shape prior segmentation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 1164–1170. IEEE, 2005.

[122] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

[123] Frédéric Cazals and Joachim Giesen. Delaunay triangulation based surface reconstruction: ideas and algorithms. *Project Geometrica, INRIA Sophia*, 2004.

[124] Lukas Gruber, Tobias Langlotz, Pradeep Sen, Tobias Hoherer, and Dieter Schmalstieg. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *2014 IEEE Virtual Reality (VR)*, pages 15–20. IEEE, 2014.

VITA

A'aeshah Alhakamy is a faculty member of Computers and Information Technology at the University of Tabuk, Tabuk, Saudi Arabia, where she awarded a full scholarship to pursue higher education abroad in Computer Graphics, Visualization, and Imaging. She is currently a Ph.D. candidate in Computer and Information Science at Purdue School of Science, Indiana University-Purdue University Indianapolis (IUPUI) under the supervision of Professor Mihran Tuceryan. She holds M.Sc. (2016) in Computer and Information Science from the Purdue School of Science, Indiana University-Purdue University Indianapolis (IUPUI), IN, USA and a B.Sc. (2005) in Computer Science from the Taibah University, Medina, Saudi Arabia. She worked as Teaching Assistant at the Department of Computer and Information Science in fall 2017.

In addition, she is working as Graduate Research Assistant at the School of Informatics and Computing in Human-Centered Computing department under the supervision of Professor Francesco Cafaro in Data Lab with NSF funded project named "Aiding Reasoning about Correlation and Causation". She is an ACM member Since 2017 and a member of the IEEE Computer Society. Her primary research interests include: Computer Graphics (CG), Visualization, Imaging, Augmented Reality (AR). Particularly, illumination models in mixed and augmented reality applications; interaction between the virtual and real objects, worlds; human-data interactions and embodied interaction systems with data visualization. You can find A'aeshah Alhakamy on `https://www.cs.iupui.edu/~aalhakam/`