FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

High-Quality Real-Time Global Illumination in Augmented Reality

PhD THESIS

submitted in partial fulfillment of the requirements of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Peter Kán
Registration Number 1027645

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Assoc. Prof. Dr. Hannes Kaufmann

External reviewers:
Professor Alan Chalmers. WMG, University of Warwick, United Kingdom.
Professor Mark Billinghurst. HIT Lab NZ, University of Canterbury, New Zealand.

Wien, 01.06.2014

(Signature of Author)

(Signature of Advisor)

Declaration of Authorship

Peter Kán

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

_____ _____

(Place, Date) (Signature of Author)

Acknowledgements

First of all, I would like to express the gratitude to my supervisor Hannes Kaufmann for his helpful guidance, support, motivation, valuable advices, and immediate feedback. This thesis would not have been possible without his encouraging leadership. A special thank is given to Christian Breiteneder, who was always willing to discuss the issues in my thesis and suggest the improvements when needed.

My thanks goes to all members of VR Group at Vienna University of Technology, especially to Christian Schönauer, Thomas Pintaric, Iana Podkosova, and others, for a wonderful working atmosphere, valuable discussions, proofreading and help with technical issues. I am grateful to Alan Chalmers who kindly agreed to be the external reviewer of this thesis.

Part of my PhD research was conducted at HIT Lab New Zealand. It was an incredible experience to be part of this lab. Therefore, I am thankful to Mark Billinghurst who provided me with this opportunity and who agreed to be the external reviewer of this thesis. Moreover, I acknowledge all members of HIT Lab NZ, particularly, Andreas Dünser, Adrian Clark, Gun Lee, and others, for their help with a user study, valuable suggestions and comments.

The evaluation of the presented algorithms was done by the comparison to the state of the art methods. I thank the authors of the used methods, particularly to Martin Knecht with coauthors and Tobias Franke, for collaboration on evaluation and for providing the results of their methods. Additionally, my thanks are sent to Holger Regenbrecht for providing his new AR presence questionnaire.

I would like to especially thank the most important person in my life, my wife Katka, who provides me with endless support and inspiration in my work.

This PhD research was funded by the Vienna PhD School of Informatics, the IMS research group and the FFG Bridge MobileBRDF project.

The dataset of various 3D models was used to create figures and to evaluate the results of the presented methods. The teapot model is the courtesy of Martin Newell. The monkey model is part of the Blender software. The bunny model, the models of the Chinese dragon and Happy Buddha belong to the Stanford 3D Scanning Repository provided by the Stanford Computer Graphics Laboratory. The Arc de Triomphe model belongs to the City of Sights project [53]. The Phlegmatic Dragon model, used in comparison to delta voxel cone tracing, is the courtesy of UTIA, Academy of Sciences of the Czech Republic, and CGG, Czech Technical University in Prague. Some of the used models were obtained from the McGuire graphics archive [111].

Abstract

High-quality image synthesis, indistinguishable from reality, has been one of the most important problems in computer graphics from its beginning. Image synthesis in augmented reality (AR) poses an even more challenging problem, because coherence of virtual and real objects is required. Especially, visual coherence plays an important role in AR. Visual coherence can be achieved by calculating global illumination which introduces the light interaction between virtual and real objects. Correct light interaction provides precise information about spatial location, radiometric properties, and geometric details of inserted virtual objects. In order to calculate light interaction accurately, high-quality global illumination is required. However, high-quality global illumination algorithms have not been suitable for real-time AR due to their high computational cost. Global illumination in AR can be beneficial in many areas including automotive or architectural design, medical therapy, rehabilitation, surgery, education, movie production, and others.

This thesis approaches the problem of visual coherence in augmented reality by adopting the physically based rendering algorithms and presenting a novel GPU implementation of these algorithms. The developed rendering algorithms calculate the two solutions of global illumination, required for rendering in AR, in one pass by using a novel one-pass differential rendering algorithm. The rendering algorithms, presented in this thesis, are based on GPU ray tracing which provides high quality results. The developed rendering system computes various visual features in high quality. These features include depth of field, shadows, specular and diffuse global illumination, reflections, and refractions. Moreover, numerous improvements of the physically based rendering algorithms are presented which allow fast and accurate light transport calculation in AR. Additionally, this thesis presents the differential progressive path tracing algorithm which can calculate the unbiased AR solution in a progressive fashion.

Finally, the presented methods are compared to the state of the art in real-time global illumination for AR. The results show that our high-quality global illumination outperforms other methods in terms of accuracy of the rendered images. Additionally, the human perception of developed global illumination methods for AR is evaluated. The impact of the presented rendering algorithms to visual realism and to the sense of presence is studied in this thesis. The results suggest that high-quality global illumination has a positive impact on the realism and presence perceived by users in AR. Thus, future AR applications can benefit from the algorithms developed in this thesis.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Contributions . . . 4
- 1.4 Thesis Outline . . . 6

2 Theoretical Background and Related Work **7**
- 2.1 Theory of Light Transport . . . 7
  - 2.1.1 Rendering Equation . . . 7
  - 2.1.2 Bidirectional Reflectance Distribution Function . . . 10
- 2.2 Physically Based Global Illumination . . . 12
  - 2.2.1 Ray Tracing . . . 12
  - 2.2.2 Path Tracing . . . 14
  - 2.2.3 Photon Mapping . . . 16
  - 2.2.4 Irradiance Caching . . . 18
- 2.3 Interactive Global Illumination . . . 22
  - 2.3.1 Ray Tracing Methods . . . 22
  - 2.3.2 Instant Radiosity . . . 26
  - 2.3.3 Screen Space Methods . . . 27
  - 2.3.4 Volumetric Methods . . . 28
  - 2.3.5 Precomputed Radiance Transfer . . . 29
  - 2.3.6 Final Gathering . . . 30
  - 2.3.7 Comparison . . . 30
- 2.4 Compositing . . . 31
- 2.5 Global Illumination in Augmented Reality . . . 33
- 2.6 Depth of Field in Augmented Reality . . . 36
- 2.7 Registration and Tracking in Augmented Reality . . . 37
- 2.8 Reconstruction of the Real Scene . . . 38
  - 2.8.1 Geometry Reconstruction . . . 39
  - 2.8.2 Material Reconstruction . . . 40
  - 2.8.3 Light Source Estimation . . . 40
- 2.9 Presence in Augmented Reality . . . 42

3 Ray Tracing in Augmented Reality **43**
3.1 System Overview . . . 43
3.2 Light Source Estimation . . . 45
3.2.1 Light Source Estimation from an Environment Camera . . . 45
3.2.2 Image-Based Lighting . . . 46
3.3 One-Pass Differential Rendering . . . 47
3.4 Physically Based Depth of Field in Augmented Reality . . . 49
3.5 Antialiasing . . . 51
3.6 HDR Rendering . . . 52
3.7 Implementation . . . 52
3.7.1 Hardware . . . 53
3.7.2 System Implementation . . . 53
3.7.3 Ray Tracing . . . 54
3.7.4 Scene Representation . . . 55

4 Physically Based Global Illumination in Augmented Reality **57**
4.1 Specular Light Transport in Augmented Reality . . . 57
4.1.1 Specular Refraction and Reflection . . . 58
4.1.2 Caustics . . . 61
4.2 Diffuse Light Transport in Augmented Reality . . . 64
4.2.1 Differential Irradiance Caching . . . 65
4.2.2 Differential Irradiance Calculation . . . 67
4.2.3 Differential Irradiance Splatting . . . 70
4.2.4 Cache Miss Detection . . . 70
4.2.5 Reprojection to the Splat Buffer . . . 71
4.3 Global Illumination System for Augmented Reality . . . 72
4.4 Differential Progressive Path Tracing . . . 74
4.4.1 Previsualization Framework . . . 77
4.4.2 Interaction . . . 78
4.5 Implementation . . . 78
4.5.1 Photon Mapping . . . 78
4.5.2 Differential Irradiance Caching . . . 78

5 Evaluation and Results **81**
5.1 Rendering Speed . . . 81
5.1.1 Depth of Field . . . 82
5.1.2 Specular Global Illumination . . . 83
5.1.3 Diffuse Global Illumination . . . 83
5.1.4 Differential Progressive Path Tracing . . . 86
5.2 Quality Comparison . . . 86
5.3 Study of Visual Realism . . . 90
5.3.1 Depth of Field . . . 91
5.3.2 Specular Global Illumination . . . 91
5.4 Presence Study . . . 95

5.4.1 Illumination Models 95
5.4.2 Methods 95
5.4.3 Results 98
5.5 Discussion 101
5.6 Application Areas 102

6 Conclusion **105**
6.1 Summary 105
6.2 Future Research 107

Bibliography **109**

CHAPTER 1

Introduction

This thesis proposes novel algorithms for high-quality real-time rendering in augmented reality (AR). The physically based nature of light transport is simulated in these rendering algorithms. Furthermore, user studies, suggesting the importance of high-quality rendering for AR and the positive impact on presence and realism, are shown.

1.1 Motivation

Augmented reality is a technology which composites virtual objects with the real world. According to Azuma *"Ideally, it would appear to the user that the virtual and real objects coexisted in the same space"* [7]. The three main characteristics of AR are defined by Azuma:

- AR combines real and virtual
- AR is interactive in real time
- AR is registered in 3D

Thus, coherent behavior of virtual and real objects in AR is essential. In order to make virtual objects appear as a part of the real world, two types of coherence have to be reached: Spatial coherence and visual coherence. Spatial Coherence is approached by accurate tracking and registration of virtual objects in the real space. Visual coherence is approached by accurate rendering and light transport calculation. Visual coherence is an important property of AR, because it can enhance the perception of spatial relationships and radiometric and geometric properties of inserted virtual objects. Furthermore, the realistic appearance of virtual objects and proper light interaction with the real world is of high interest in many application areas e.g. entertainment, design, medicine (therapy, rehabilitation, preoperative visualization), education and others. The main problem of visual coherence between the virtual and real scenes is that accurate light-transport has to be calculated to produce realistic results. This thesis approaches

the problem of visual coherence in AR by introducing physically based rendering algorithms into the context of an interactive AR scenario.

Previous research in AR focused mainly on tracking, registration, and spatial coherence. The accurate registration of virtual objects within the real scene is essential for the correct positioning of virtual objects. However, the visual coherence received less attention in the past. Simple rendering and local lighting models have been used. According to Azuma [7] it should appear to the user that virtual and real objects coexist in the same space. Therefore, the visual coherence, appearance of virtual objects, and accurate light transport between real and virtual objects are essential. The poor visual quality of virtual objects has a negative impact on the presence perceived by the user and his general AR experience. High-quality rendering, including physically based global light transport, improves the perceived presence of virtual objects, their visual appearance, and the realism of AR. In contrast to local illumination, global illumination evaluates the light interreflections between objects and calculates indirect lighting, which highly increases the visual coherence between real and virtual scenes. The comparison of local and global illumination rendering in AR can be seen in Figure 1.1

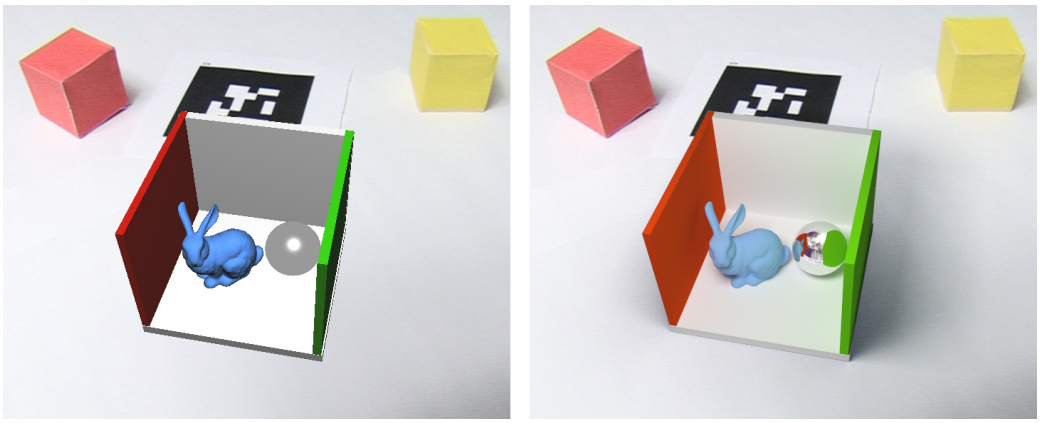

Figure 1.1: Comparison of local (left) and global (right) illumination rendering in augmented reality. The Phong BRDF model is used in both scenes for diffuse surfaces.

Real-time performance is the essential requirement of AR. In order to allow a user to interact with virtual content, rendering and registration algorithms have to work in real time. The disadvantage of physically based rendering is its high computational cost. Therefore, research in fast high-quality rendering is needed. Furthermore, real-time performance of physically based algorithms is of high interest in computer graphics. This thesis focuses on the improvement of physically based algorithms, their GPU implementation, and their effective application in AR.

1.2 Problem Statement

According to the definition of AR (Section 1.1) virtual objects have to be spatially and visually registered in 3D and the interactivity has to be preserved. These requirements pose the following open problems for AR:

Visual Coherence. High quality image synthesis has been a challenging problem in the field of computer graphics from its beginning. Wide mathematical background [67, 73, 133] was developed to precisely model the problem of light transport with the goal to create artificial images indistinguishable from reality. The problem of photorealistic rendering in AR is even more complicated, because the light transport between real and virtual objects has to be simulated. Despite of its computational complexity, the visual coherence is an important problem because many applications could benefit from photorealistic AR.

Various visual effects contributing to the visual coherence are of high interest in AR. One example is specular light transport which causes reflection, refraction, and caustics. Another important feature is a correct camera simulation causing a depth of field effect. Due to the optics of real cameras, objects which are out of focus appear blurred. The same effect has to be simulated in rendering to achieve coherence between the video images and computer generated content. Important features of light transport are indirect light reflections which cause light contribution to the geometric surface. Most AR applications omit this indirect light contribution and therefore suffer from missing indirect illumination. The problem of visual coherence between virtual and real scenes requires the accurate global light transport to be simulated. Finally, the problem of visual coherence is partially caused by aliasing artifacts created by an insufficient sampling rate. Therefore, antialiasing is required in AR. This thesis addresses the mentioned visual effects by presenting interactive high-quality rendering algorithms for AR.

Interactivity. Interactivity is an essential property of AR. However, light interreflections between virtual and real objects have to be calculated to achieve visual coherence. This calculation involves a global light transport simulation. The main problem of high-quality light transport is its high computational cost. Rendering synthetic objects into real videos doubles the computational expenses since two solutions of light transport are needed for compositing [27, 36]. Therefore, high-quality rendering techniques have not been suitable for real-time applications. The speed of high-quality rendering algorithms has to be improved due to the interactivity requirement of AR. This thesis solves the problem of slow rendering speed by improving the algorithms and presenting novel parallel GPU implementations. In order to effectively composite virtual objects with a real image and to simulate light interreflections between real and virtual worlds a novel one-pass differential rendering algorithm is introduced.

Light Source Estimation. Light sources used in the virtual world have to be similar to ones used in the real world to achieve similar visual appearance of virtual and real objects. Accurate light sources create coherent shadows which have high impact on presence and perception of the spatial relationships in AR. Therefore, the real lighting has to be reconstructed. This problem can be approached by image-based lighting [132] or light source estimation [37]. This thesis investigates how to acquire the real lighting in real time.

Geometry and Material Reconstruction. The light interaction between virtual and real objects is needed to increase visual coherence in AR. Thus, the geometry and materials of the real scene have to be known to properly simulate the light reflection on the real surfaces. For example, if the scene contains a real mirror surface, the virtual objects should be reflected in this

mirror. In order to properly simulate this mirror reflection, the rendering system needs to know the geometry and material of the mirror surface. This problem can be solved in two possible ways. The first solution is to use a predefined model of the real scene as an input to the rendering. In this case, the geometry and materials are modeled before the system runs. Both real and virtual objects can be dynamic during the run time, but their geometry and materials have to be known in advance. The second solution is to reconstruct the 3D model of the real scene in real time. In this case, real geometry and materials are reconstructed when the system is running. This thesis employs a predefined model of the real scene to simulate correct light interaction between virtual and real objects. However, the developed system can be extended by including any of the real-time scene reconstruction techniques (Section 2.8).

Tracking. Tracking is an essential part of AR applications, because virtual objects have to be spatially registered in 3D. Thus, a tracking system needs to estimate a camera pose with relation to the real world. If a camera pose is accurately estimated, the virtual objects can be placed on a particular location in the real world. Tracking should be stable to achieve a temporally stable positioning of virtual objects. A vast amount of tracking approaches was presented in previous AR research (Section 2.7). This thesis focuses mainly on high-quality rendering and visual coherence in AR. Thus, available tracking solutions are reused.

User Perception. A visual coherence perceived by the user highly depends on a subjective observation of AR. Therefore, it is important to study the effect of AR on the user. Different factors can be studied that help to achieve a high quality AR experience: Presence, realism, usability, task performance, physiological properties, etc. Several experiments are described in this thesis. They study the effect of high-quality rendering on the users' perception of AR.

1.3 Contributions

This thesis focuses mainly on the problem of visual coherence and interactivity in AR. Additionally, the problems of light source estimation and human perception are approached. The visual coherence is addressed by proposing novel algorithms for high-quality rendering in AR based on previous research in physically based rendering [133]. To overcome the problem of computationally expensive light transport, the improvements of physically based algorithms together with the novel GPU implementations are introduced. The proposed algorithms are based on GPU ray tracing. Improvements of the following physically based rendering methods are proposed: Path tracing [73], photon mapping [71], irradiance caching [193], ray tracing, etc. The problem of two rendering solutions needed for compositing is solved by proposing a one-pass differential rendering algorithm which works directly in the ray tracing pipeline. The proposed algorithms utilize the massive parallel power of GPUs to achieve high-quality results while preserving interactivity.

A high-quality rendering system for AR based on the presented algorithms was developed during the PhD research. The developed rendering system takes a frame of a real-time video stream and superimposes virtual geometry onto it, introducing light interreflections between real and virtual worlds, on the fly. Direct illumination is calculated by ray tracing from the camera.

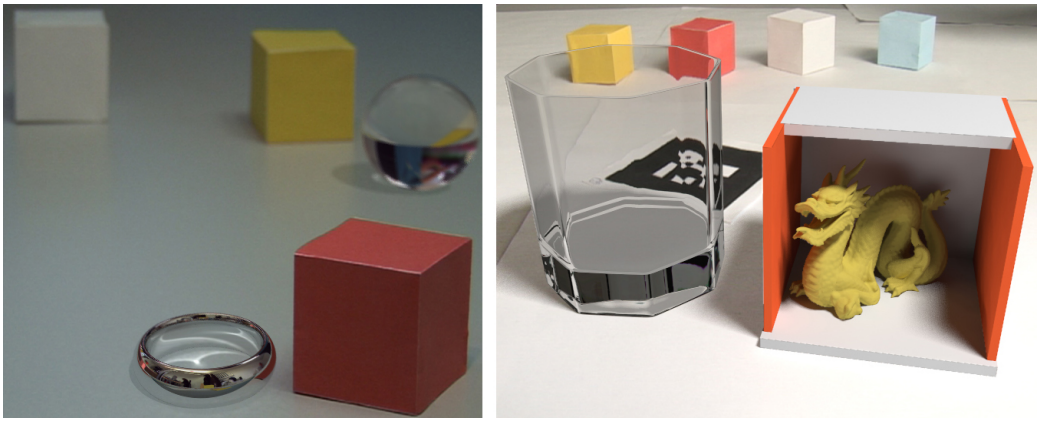

Figure 1.2: Virtual objects are placed into real scenes and advanced light transport effects are calculated. Left: Scene contains three real cubes, virtual ring and a virtual glass sphere. Correct refraction of real world in the glass sphere, caustics created by the virtual ring on the real table, and depth of field effect can be seen. Right: A virtual glass and a diffuse box with a dragon are inserted into the real scene. Correct diffuse global illumination is calculated by differential irradiance caching (Section 4.2.1). The virtual glass correctly refracts the real cubes behind it.

The multi-bounce global illumination is supported in the presented system. Additionally, the real lighting is estimated and employed for rendering in AR.

The importance of high dynamic range (HDR) rendering was highlighted by Debevec [27]. Therefore, the developed system calculates the result in HDR and finally tonemaps it for displaying on standard display devices. The results of high-quality rendering in AR, produced by the developed system, are shown in Figure 1.2. Finally, the results of our evaluation demonstrate the contribution of our algorithms to augmented reality.

The main contributions of this PhD thesis are:

- A one-pass differential rendering in ray tracing. This algorithm effectively solves the problem of compositing in a single ray tracing pass.
- Physically based depth of field simulation in AR. A finite sized aperture model is used to introduce blur, in virtual camera, coherent to the real one.
- High-quality specular light transport in AR by GPU ray tracing and photon mapping. Physically accurate reflections and refractions are produced together with high-quality caustics in interactive rates.
- A novel algorithm for high-quality diffuse light transport in AR named Differential Irradiance Caching. A combination of GPU ray tracing and rasterization is used in this algorithm. The irradiances in cache records are evaluated using Monte Carlo integration by shooting recursive rays into random directions above the surface point. This method naturally enables the calculation of multiple bounces of indirect illumination. A novel miss-detection technique is proposed which can use a linear irradiance cache.

- Differential progressive path tracing, an algorithm for physically accurate light transport simulation in AR. A temporal coherence is utilized to obtain a high-quality final image in a progressive manner. The implementation offers two interaction modes. First, an interactive preview mode allows for real-time observation of AR scene in a coarse quality. Second, the progressive refinement mode progressively calculates the full quality of the AR image.

- User studies evaluating the visual coherence, presence and realness of virtual objects. The results suggest that high-quality rendering has a positive impact to the perception of AR.

Scientific Publications. The results of this PhD research were published in the following publications:

- Peter Kán and Hannes Kaufmann. Physically-Based Depth of Field in Augmented Reality In *Eurographics 2012 Short papers*, pages 89–92, Cagliari, Italy, 2012. Eurographics Association.

- Peter Kán and Hannes Kaufmann. High-Quality Reflections, Refractions, and Caustics in Augmented Reality and their Contribution to Visual Coherence In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99-108, 2012. IEEE Computer Society.

- Peter Kán and Hannes Kaufmann. Differential Progressive Path Tracing for High-Quality Previsualization and Relighting in Augmented Reality In *ISVC 2013, Part II, LNCS 8034*, pages 328-338, 2013. Springer-Verlag Berlin Heidelberg.

- Peter Kán and Hannes Kaufmann. Differential Irradiance Caching for Fast High-Quality Light Transport Between Virtual and Real Worlds In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 133-141, 2013. IEEE Computer Society.

- Peter Kán, Andreas Dünser, Mark Billinghurst, Christian Schönauer and Hannes Kaufmann. The Effects of Direct and Global Illumination on Presence in Augmented Reality In *Challenging Presence - Proceedings of 15th International Conference on Presence (ISPR 2014)*, pages 223-230, 2014. Facultas Verlags- und Buchhandels AG.

1.4 Thesis Outline

The thesis continues as follows. Theoretical background and recent research are discussed in Chapter 2. The core of the developed rendering system based on ray tracing together with one-pass differential rendering are described in Chapter 3. Chapter 4 shows the algorithms for global light transport in AR together with the implementation of the mentioned algorithms. The evaluation and results are shown in Chapter 5. Chapter 6 concludes the thesis.

CHAPTER 2

Theoretical Background and Related Work

2.1 Theory of Light Transport

Photorealistic image synthesis requires the simulation of high-quality global illumination (GI). This simulation has to take the light contribution from light sources and the light reflected from other objects into account. Both of these light contributions to the final image are modeled by the rendering equation described in Section 2.1.1. High-quality rendering algorithms, evaluating the rendering equation in a physically based manner, were developed in previous research (Section 2.2). In contrast to that, real-time approximative algorithms for global illumination calculation were developed to allow fast calculation at the cost of an approximation error (Section 2.3). The real-time approximative methods, calculating GI in an AR scenario, were presented in the past (Section 2.5). This section describes the mathematical theory for modeling the light transport.

2.1.1 Rendering Equation

In order to simulate the process of global light transport in the scene, a mathematical model describing this process is required. The rendering equation, modeling the light transport, was simultaneously introduced by Kajiya and Immel [67, 73]. This equation mathematically describes the process of light propagation and reflection. Some simplifications are made in the rendering equation in comparison to the light transport in the real world. For example the media between objects is considered to be of homogenous refractive index and does not itself participate in the light scattering.

The rendering equation describes the light transport between surface points by calculating the light reflected from a scene point towards a desired direction. The most important quantity in the rendering equation is radiance, which represents the differential radiant power per unit projected area per unit solid angle [33]. Radiance expresses the amount of light that arrives at

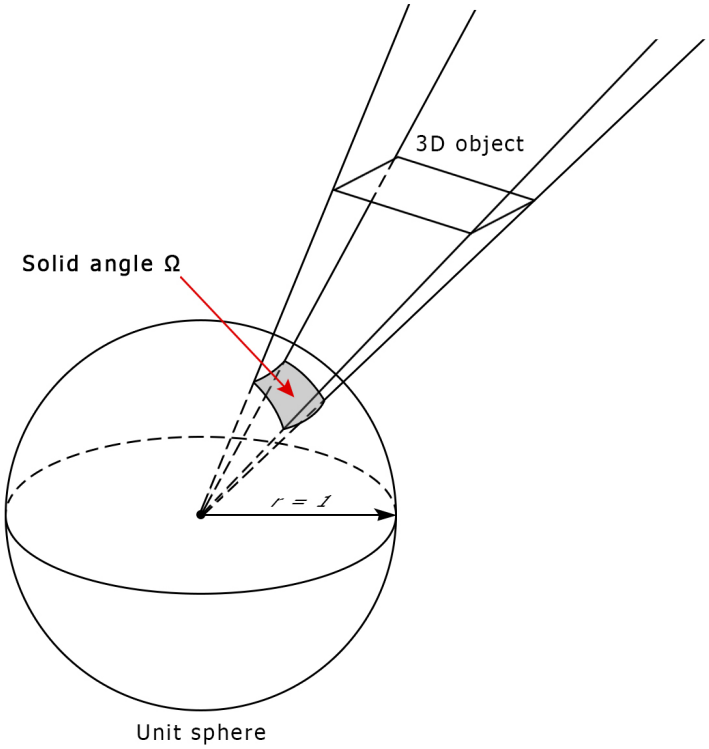

Figure 2.1: The solid angle Ω subtended by a 3D object, is equal to the surface area of its projection onto the unit sphere.

or leave from a certain surface point in particular direction. The unit of radiance is $\frac{W}{steradian \cdot m^2}$. Radiance is a five-dimensional quantity that varies with position and direction and it is denoted as $L(x, \vec{\omega})$. The solid angle Ω, subtended by a surface S, is defined as the surface area Ω of a unit sphere covered by the surface's projection onto the sphere (Figure 2.1). In order to calculate the radiance exiting from a scene point x towards direction $\vec{\omega}$, the emitted radiance L_e and reflected radiance L_r are summed:

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + L_r(x, \vec{\omega}) \tag{2.1}$$

The outgoing radiance is denoted by L_o. ω stands for a direction of outgoing radiance. The integration of incoming light to the scene point is necessary to calculate the reflected radiance. The rendering equation can be written as:

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') \mid \vec{n} \cdot \vec{\omega}' \mid d\omega' \tag{2.2}$$

where L_o is the outgoing radiance from point x towards direction $\vec{\omega}$, L_e is the radiance emitted from point x to direction $\vec{\omega}$ and L_i is the radiance of incident light incoming from direction $-\vec{\omega}'$ to point x, while the incoming radiance is integrated on the hemisphere Ω. This

hemisphere is centered at point x and rotated with respect to the normal $\vec{n}$. f_r denotes the bidirectional reflectance distribution function (BRDF) of the surface at position x, with surface normal $\vec{n}$. The BRDF function is described in detail in Section 2.1.2. The dot product $\vec{n} \cdot \vec{\omega}'$ is the cosine of the angle between vector $\vec{\omega}'$ and the surface normal. $d\omega'$ is the differential solid angle of incoming light. The light reflection on the surface is depicted in Figure 2.2.

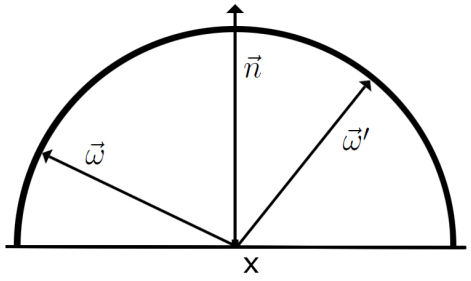

Figure 2.2: Light reflection on the surface.

The rendering equation can be solved numerically in different ways. The first one is the forward light propagation in which the light rays, starting from the light position, are traversing the scene, reflecting on surfaces, and finally reaching the camera. Forward light propagation is employed to propagate the light rays from the light position through the scene to the camera. For example, methods based on photon mapping [11, 71, 97, 112] use this approach. Another way to calculate the light arriving to the sensor of camera is the backward light propagation from the camera towards the scene and light positions. Ray tracing and path tracing [73, 195] are well known techniques, in which the rays from the camera are traced to obtain the final rendering solution. Path tracing produces highly realistic and physically accurate solutions. Finally, the bidirectional techniques combine the two above described approaches.

Types of Illumination. The calculation of the light transport in the scene can consists of the following types of illumination:

- **Direct illumination** - Contains only a single light reflection from surfaces to the camera. In this subpart of the illumination, the light is emitted from light sources and reflected from surfaces towards the camera. The result of this calculation contains also shadows. Direct illumination is the most important one for rendering, because it has the biggest contribution of the light energy to the image.

- **Indirect illumination** - This illumination consists of multiple bounces of light being reflected from surfaces in the scene. The single-bounce direct illumination is excluded from indirect illumination. Therefore, it contains two or more bounces. Indirect illumination takes the biggest computational expense of the light transport calculation. However, it is very important, because it adds the light contribution to the shadowed regions in the scene and it increases the photorealism of the rendered result.

- **Local illumination** - Many real-time applications only use the local illumination calculation due to its low computational cost. However, the local lighting model is very limited and cannot include the light interaction with other surfaces in the scene. In this model, the lighting is calculated only locally on the specific surface point.
- **Global illumination** - Global illumination accounts for a full light transport in the scene including direct and indirect illumination. The light reflections on the scene surfaces are calculated. Global illumination is essential for realistic rendering.

2.1.2 Bidirectional Reflectance Distribution Function

The bidirectional reflectance distribution function (BRDF) describes the behavior of the light reflection on a particular material. The BRDF determines the ratio between the differential reflected radiance L_r outgoing to direction $\vec{\omega}$ and the product of the incident radiance L_i incoming from direction $-\vec{\omega}'$ and the cosine of the incident angle between $\vec{n}$ and $\vec{\omega}'$ [133]. It can be described by the following equation

$$f_r(x, \vec{\omega}', \vec{\omega}) = \frac{dL_r(x, \vec{\omega})}{L_i(x, \vec{\omega}') \mid \vec{n} \cdot \vec{\omega}' \mid d\omega'} \tag{2.3}$$

In practice, some general types of surfaces can be considered depending on their reflectance properties. Perfectly diffuse surfaces reflect the light equally into all directions on the hemisphere (Figure 2.3 left). Perfectly specular surfaces reflect the light only in one direction; the direction of perfect reflection (Figure 2.3 right). However, in the real world the perfectly specular reflection does not occur, therefore the rays scattered around the specular reflection direction are considered. This reflection is called glossy specular reflection (Figure 2.3 middle).

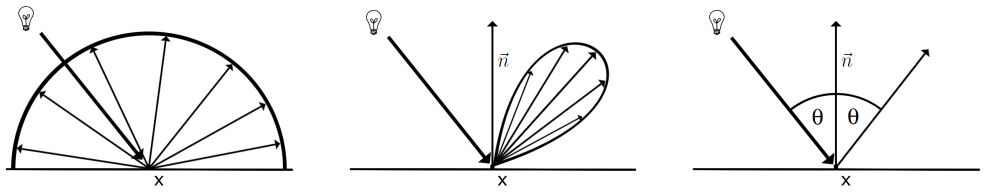

Figure 2.3: Left: Lambertian diffuse reflection. Middle: Glossy specular reflection. Right: Perfect specular reflection.

Physical Constraints. The set of all possible BRDFs is constrained by two physical laws. Should the BRDF be physically plausible, it has to conform to these two laws [96, 105]:

- **Helmholtz reciprocity**. Due to the reciprocal nature of the light reflection, the incoming direction $\vec{\omega}'$ and the outgoing direction $\vec{\omega}$ may be interchanged while the function will give the same result. This constraint can be described by:

$$f_r(x, \vec{\omega}', \vec{\omega}) = f_r(x, \vec{\omega}, \vec{\omega}') \tag{2.4}$$

- **Energy conservation**. The energy of light, reflected from a surface point, cannot be higher than the energy of incoming light to this point. Thus, the fraction of light coming in from any direction that is reflected within the entire hemisphere should be smaller than 1 to ensure the conservation of energy. This fraction is expressed by the total hemispherical reflectivity:

$$\int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \mid \vec{n} \cdot \vec{\omega} \mid \, d\omega \leq 1, \quad \forall x, \forall \vec{\omega}' \tag{2.5}$$

Many BRDF models were introduced in past research to simulate different kinds of materials. We can divide them to the three groups. First, physically based BRDFs were created to simulate the behavior of material in a physically correct way. These models include the Oren-Nayar diffuse reflection [123], the Torrance-Sparrow model [178], the Blinn model [13], and the anisotropic microfacet model [5]. Second, the models for measured sampled BRDFs were presented. A data-driven reflectance model and data format supporting an isotropic measured BRDF was presented by Matusik et al. [109, 110]. Moreover, the authors measured approximately 100 different materials, which can be used in physically based rendering. Third, the empirical BRDF models were presented. These models were derived from empirical observation of light reflection on materials and they have intuitive parameters. An empirical anisotropic BRDF model was presented by Ward [192] to fit measured BRDF data. This model has become widely used due to its capability of modeling various materials and its intuitiveness. Later, the efficient evaluation of Ward BRDF and the derivation of the probability density function for its associated Monte Carlo sampling was presented in [187]. The Phong model [134] is a widely known empirical reflection model. This model was later extended to create a physically plausible Phong BRDF model [96, 105]. The Phong BRDF model is described by Equation 2.6.

$$f_r(x, \vec{\omega}', \vec{\omega}) = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha \tag{2.6}$$

$$\vec{\omega}_r = 2(\vec{\omega}' \cdot \vec{n})\vec{n} - \vec{\omega}' \tag{2.7}$$

k_d and k_s are the respective diffuse and specular reflectivity representing the fraction of light which is reflected diffusely and specularly. n is the specular exponent representing the shininess of the surface. Higher values of n result in sharper specular reflections. α is the angle between the perfect specular reflective direction $\vec{\omega}_r$ (Equation 2.7) and the outgoing direction $\vec{\omega}$. Values larger than $\pi/2$ are clamped to $\pi/2$ in order not to get any negative cosine values. $\vec{n}$ is the surface normal at point x. The following condition has to be met to satisfy the conservation of energy:

$$k_d + k_s \leq 1 \tag{2.8}$$

Phong BRDF is widely used in real-time graphics due to its inexpensive evaluation and the intuitiveness of the parameters. This model is also accommodated in algorithms developed in this thesis, presented in Chapters 3 and 4.

2.2 Physically Based Global Illumination

High-quality photorealistic rendering has been one of the main research topics in the area of computer graphics. Thus, many algorithms were developed which simulate the light transport. They follow the models derived from optics and light transport in real world. An elegant numerical solution of the rendering equation evaluates the light contribution in a recursive manner from the camera position. Ray tracing [195] and path tracing [73] techniques are based on this principle. The forward light propagation is known as photon mapping technique [70]. In this thesis both forward and backward light propagation approaches are used. In addition to the mentioned global illumination methods other GI algorithms were presented in computer graphics research including Metropolis light transport [180], precomputed radiance transfer [164,189], and instant global illumination [80]. Some of these approaches are described in Section 2.3

2.2.1 Ray Tracing

Ray tracing is an efficient rendering algorithm that serves as a basis for many physically based rendering methods. The recursive ray tracing was originally introduced by Turner Whitted in [195] as a recursive algorithm for the rendering of highly specular surfaces. In ray tracing, the rays are propagated in a backward manner from the camera position towards the scene. Firstly, rays are generated from the camera position towards the directions of all pixels to calculate the radiance incoming to these pixels. These rays are called primary rays. Then, the additional rays are traced to calculate the contribution of light reflected from geometric surfaces. Two types of light are considered. The first type is direct light which is the light emitted from light sources and directly reflected on a scene surface towards the camera. The contribution of direct light is determined by visibility between the surface point (hitpoint) and the light source. This visibility is evaluated by shadow rays. The second type of light is indirect one which represents the light reflected from other scene surfaces to the hitpoint and then towards the camera. The reflected radiance rays are used to evaluate indirect light. The reflected radiance rays were firstly used to calculate the specular reflection only [195]. Later, this concept was extended to handle light reflection on all types of materials in distributed ray tracing [22]. The rays are recursively traced from each hitpoint of a ray and scene geometry. The ray tracing pipeline is depicted in Figure 2.4.

The image generation in ray tracing consists of the following steps:

- **Ray generation** - In this step the rays are generated and the origin and direction of each ray is set. The ray type is assigned to evaluate the specific light contribution.
- **Ray traversal** - The spatial accelerating data structure is traversed to find the intersection of the ray with scene geometry. This data structure is used to speed up the traversal phase from linear to logarithmic. The accelerating data structure subdivides the space into hierarchical subparts. The geometric primitives lie in the leaf nodes of the data structure. If the ray intersects the leaf node, the intersection of the ray and all geometry in this node is tested. Different accelerating data structures can be used depending on the requirements of the application. The most often used accelerating data structures are Kd-tree, bounding volume hierarchy(BVH), octree, etc. [61].

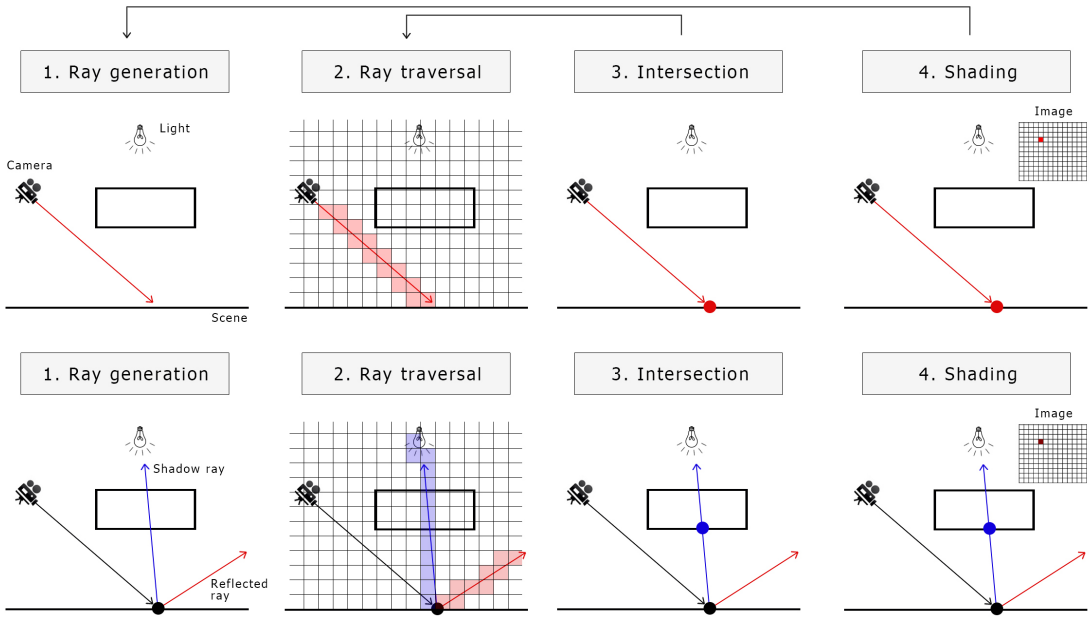

Figure 2.4: Ray tracing pipeline [165]. The first row shows the shooting of a primary ray. The second row displays the recursive shooting of a shadow ray and the reflected radiance ray from the hitpoint.

- **Intersection** - The hitpoint of the ray and scene geometry has to be found to calculate the resulting radiance from this ray. If the leaf node is found in the traversal step, the intersection with all geometric primitives lying in this node is tested.
- **Shading** - The final reflected radiance is calculated in the shading step, taking into account the material of the object. The material is represented by the BRDF function (Section 2.1.2).
- **Framebuffer** - The final color, calculated in the shading stage, is stored in a framebuffer. The image in the framebuffer can be converted from HDR to LDR and displayed on the screen.

The ray tracing pipeline contains two recursive cycles which are depicted in Figure 2.4 by upper arrows. The main recursion is caused by recursive shooting of reflected rays and shadow rays. This recursion is invoked during shading, when the radiance incoming to the hitpoint is needed as it is described by the rendering equation (Section 2.1.1). The second recursion is repeatedly invoking the ray traversal and intersection steps to calculate the intersection of ray with scene geometry. These two steps are recursive because the accelerating data structure is hierarchical.

Different types of intersection search are used for radiance rays and for shadow rays. The radiance ray has to evaluate the incoming radiance from the first surface along the ray. Therefore, the nearest hitpoint is searched in case of radiance rays. In the second case, the shadow rays are

used to evaluate the visibility between hitpoint and light sources. Thus, if any hitpoint between these two points is found the light is occluded. Searching for any hitpoint instead of the closest one increases the efficiency of the direct light calculation.

The problem of naive search for the intersection of ray and geometry is its linear complexity $O(n)$ depending on the number of surfaces in the scene. This test is executed several million times during the rendering of one frame. Thus it should be accelerated. By organizing the surfaces of the scene into an acceleration data structure, based on space partitioning, the nearest intersection is obtained far more efficiently by culling surfaces that cannot be intersected by a given ray. The complexity of search in the tree is $O(\log n)$, which introduces an important speedup in comparison to naive ray tracing. The naive and accelerated search for the intersection of ray and geometry is depicted in Figure 2.5.

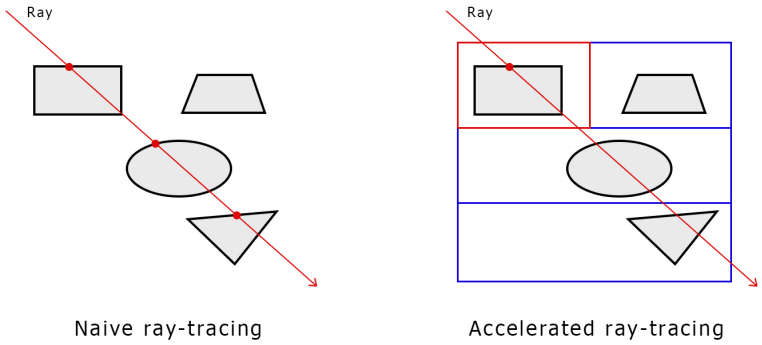

Figure 2.5: Comparison between naive ray tracing (left) and accelerated ray tracing (right) [91]. Four intersection tests have to be performed for naive ray tracing in comparison to one intersection test in case of accelerated ray tracing.

The ray tracing algorithm was later extended by Cook to distributed ray tracing [22]. This method traces many rays and distributes the parameters of multidimensional sampling space across them. The ray tracing improvement presented by Cook can simulate a wider range of phenomena by distributing the directions of the rays according to the analytic function they sample. These phenomena include motion blur, depth of field, penumbras, translucency, and glossy reflections. The concept of ray tracing is reused in many physically based rendering methods due to its elegance and universality of visibility calculation.

2.2.2 Path Tracing

The path tracing algorithm was introduced together with the rendering equation as its numerical solution by Kajiya [73]. In the path tracing, rays are used similarly to ray tracing to evaluate the reflected light. Distinctly to ray tracing, the reflection on any kind of material can be evaluated in a Monte Carlo fashion. The reflected rays are shot in many random directions from the hitpoint to evaluate the incoming radiance. The reflected radiance is then calculated by the following equation:

$$L_r(x, \vec{\omega}) \approx \frac{1}{N} \sum_{k=1}^{N} \frac{L_{i,k}(x, \vec{\omega}'_k) f_r(x, \vec{\omega}'_k, \vec{\omega}) \mid \vec{n} \cdot \vec{\omega}'_k \mid}{p(\vec{\omega}'_k)} \tag{2.9}$$

where L_r is the reflected radiance, N is the number of samples, $L_{i,k}$ is the incoming radiance of the k-th sample evaluated by the ray with the direction $\vec{\omega}'_k$. The radiance contribution from each ray sample is divided by the probability of sampling this ray $p(\vec{\omega}'_k)$.

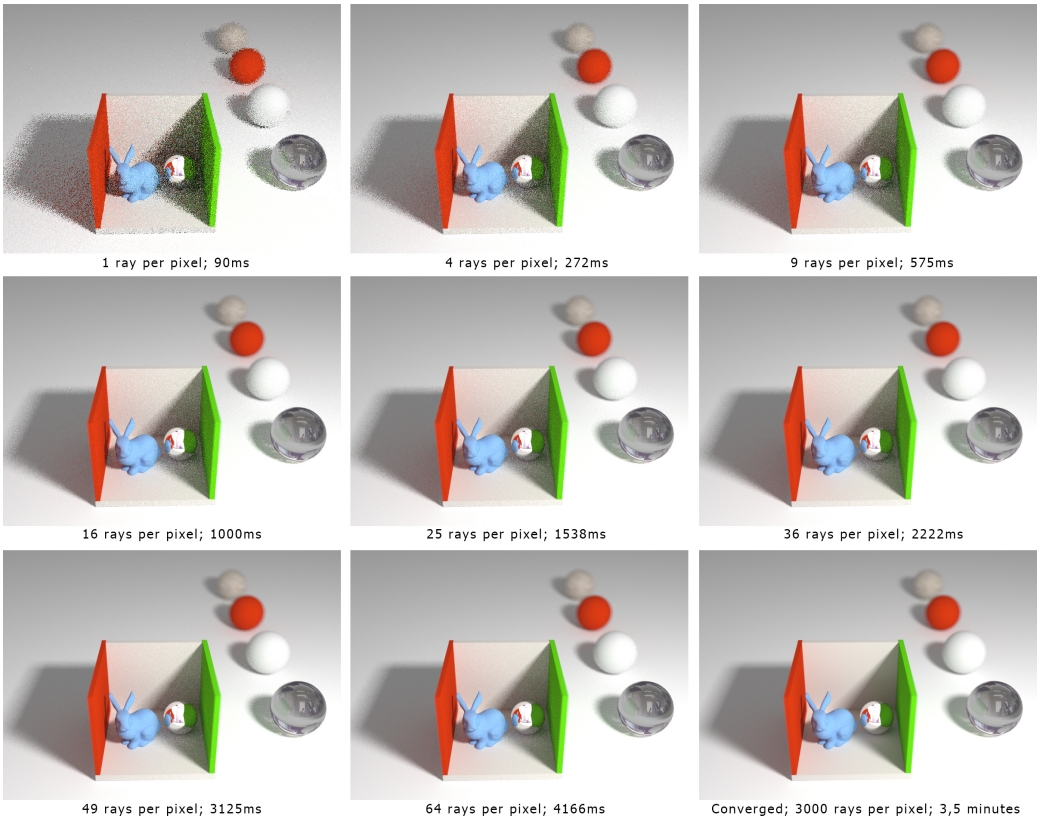

Figure 2.6: Convergence of the path tracing algorithm. The GPU path tracing implementation was used to generate the images on a PC with a Nvidia GeForce GTX 690 graphics card.

The path tracing algorithm calculates the unbiased solution of the rendering equation and converges to the physically correct result. The disadvantage of the path tracing algorithm is its slow convergence. It has been shown that the error in the Monte Carlo estimator decreases at a rate of $O(\sqrt{N})$ in the number of samples taken [133]. The convergence of the path tracing algorithm with an increasing sampling rate can be seen in Figure 2.6. Moreover, errors caused by a high variance appear as noise which is easily observable by the user. Faster convergence has been approached by improved sampling techniques. Importance sampling is variance reduction technique, which exploits the fact that the Monte Carlo estimator (Equation 2.9) converges more quickly if the samples are taken from a distribution $p(\vec{\omega}')$ that is similar to a function in the

numerator in equation 2.9 [93, 133]. If the random variables are sampled from a probability distribution that is similar in shape to the integrand, the variance is reduced faster.

Bidirectional path tracing, proposed by Lafortune and Willems [95], is the efficient improvement of the path tracing algorithm. In their method both forward (from light source) and backward (from camera) light propagation is utilized. Light paths from the camera and from light sources are randomly sampled by tracing rays. The intersections of the propagated rays and the geometry are stored. Finally, the light paths and camera paths are connected by shadow rays (Figure 2.7).

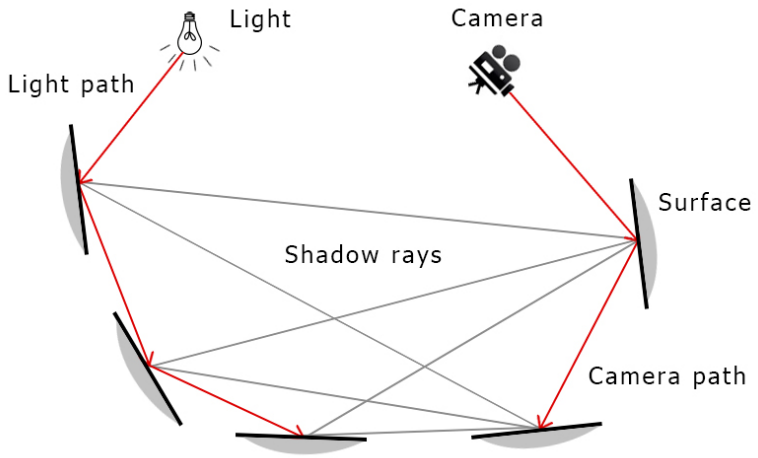

Figure 2.7: Bidirectional path tracing [95]. The rays of the light path and the camera path are traced separately and they are later connected by shadow rays to calculate the contribution of light to the camera image.

Many improvements of the path tracing algorithm were proposed in the past. Some of them focus on the effective sampling and reconstruction of calculated radiance values [58, 125, 150, 170, 176]. Recently, the path space regularization approach for calculating difficult light paths was presented [75]. It can be used to regularize the light paths which cannot be sampled in an unbiased way, especially for delta distributions.

2.2.3 Photon Mapping

Photon mapping, introduced by Jensen [70], is an efficient two-pass algorithm for the calculation of global illumination. It traces photons from the light sources into the scene. Each photon is traced through the scene using a method similar to path tracing. When a photon hits a surface it is either reflected, refracted, or absorbed. The photons are stored at their hit positions into a Kd-tree data structure called photon map. In the second step, named final gathering, the radiance information is reconstructed from the photon map at the points visible from the camera. If the radiance information at a specific scene point is requested, a set of rays is traced from this point to sample the incident radiance. If the rays hit the geometry, the radiance information is

estimated from nearby photons using k-nearest neighbors. The Kd-tree is traversed in order to find the photons contributing to a specified location. Both passes of photon mapping are depicted in Figure 2.8. Photon mapping can simulate full global illumination and it is especially efficient for the simulation of specular global illumination effects like caustics, which are difficult for other GI algorithms. An example of a rendering by photon mapping can be seen in Figure 2.9.

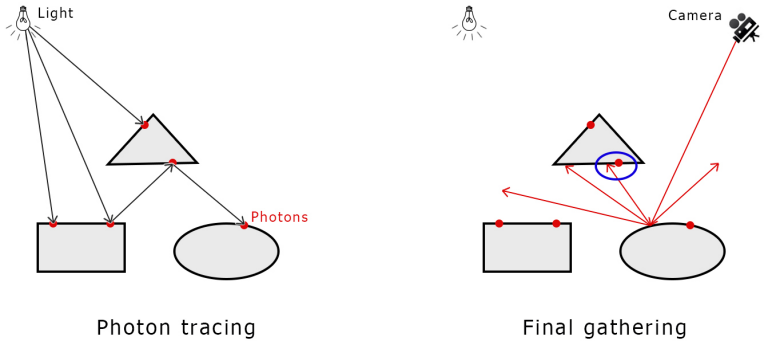

Figure 2.8: The two steps of the photon mapping algorithm: Photon tracing (left) and final gathering (right). Red dots are photons stored in a photon map. In the final gathering step, the red rays are sampled from the hitpoint of the camera ray with geometry to sample the incident radiance. The nearest photons are searched to estimate the radiance. The blue ellipse indicates that the stored photon, which is nearby to hitpoint, is taken into account in the radiance calculation.



Figure 2.9: Caustic rendered by photon mapping.

Since photon mapping was invented, many improvements have been published. Several solutions for interactive photon mapping were proposed. These solutions are described in Section 2.3.1. A special case of photon mapping reformulation is shown in splatting approaches. In these approaches, the energy carried by photons is splatted to the specified buffer and the photon's contribution and weight are accumulated to this buffer in the area of contribution. A scalable photon splatting algorithm was proposed in [99] and the photon ray splatting was proposed in [63].

Another kind of algorithmic improvement can be achieved by increasing the quality of the produced result. A high-quality photon mapping improvement was proposed by Spencer and Jones [168]. The authors focused mainly on the density estimation part of caustics rendering. They proposed the novel method of relaxing the initial distribution into one with a blue noise spectral signature. This improvement enables the use of low bandwidth kernels and increases efficiency of the rendering. The same authors also proposed hierarchical photon mapping [167] to increase the quality of the final gathering. A good overview of the photon mapping algorithm and its extensions can be found in [71, 72]

Progressive Photon Mapping. An important improvement of the photon mapping algorithm was proposed by Hachisuka et al. [59]. The authors presented a progressive photon mapping algorithm. The advantage of this algorithm over the previous photon mapping approaches is that it progressively converges to the accurate GI solution and thus produces an unbiased result. Progressive photon mapping is a multi-pass algorithm where the first pass is ray tracing followed by any number of photon tracing passes. Each photon tracing pass results in an increasingly accurate GI solution that can be visualized in order to provide progressive feedback. A new radiance estimate is used in this algorithm that converges to the correct radiance value as more photons are used. Progressive photon mapping was later extended in [18] by using the Metropolis sampling strategy. Authors applied a Metropolis-Hastings algorithm to effectively exploit local coherence among the light paths that contribute to the rendered image. A well-designed scalar contribution function was introduced as a Metropolis sampling strategy, targeting specific parts of the image with large errors to improve the efficiency of the radiance estimator.

2.2.4 Irradiance Caching

The irradiance caching, proposed by Ward [193], has become a widely used algorithm for GI calculation in production rendering. Numerous improvements were proposed in research, which made irradiance caching practical and efficient for high-quality offline rendering. The algorithm calculates the accurate irradiance in sparse scene locations and then exploits spatial coherence by interpolating between the samples. It is based on the assumption that indirect irradiance tends to change slowly over a surface. The contribution of an irradiance sample to an arbitrary scene point is determined by the weight w_i.

Irradiance caching algorithm calculates the irradiance in the positions of irradiance samples, called cache records, and it interpolates this irradiance between them. The irradiance cache records and rendering by irradiance caching are depicted in Figure 2.10. If the scene contains

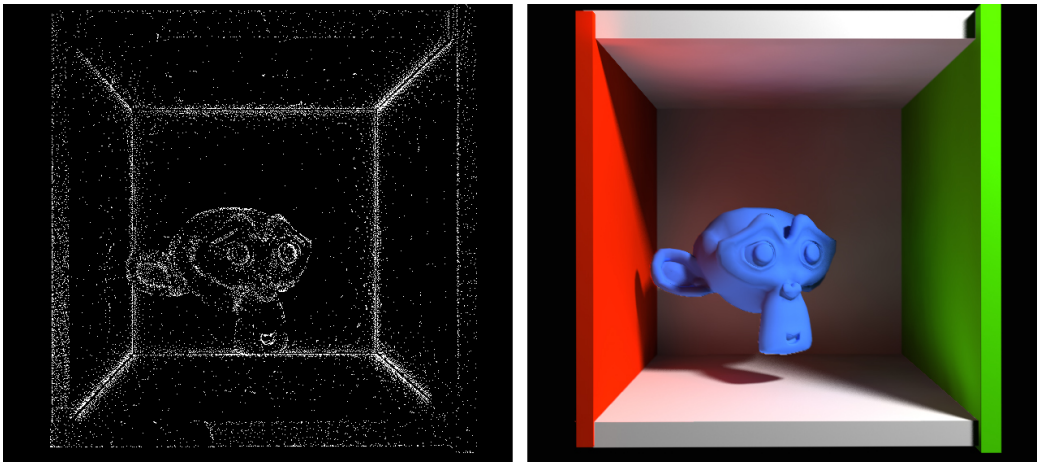

Figure 2.10: Left: Positions of irradiance cache records depicted by white dots. Accurate irradiances are calculated at these positions and they are interpolated at the nearby spatial locations. Right: Rendering by irradiance caching.

diffuse surfaces with constant BRDFs only, the diffuse term can be taken out from the integral in the rendering equation:

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \rho_d(x) \int_\Omega L_i(x,\vec{\omega}') \mid \vec{n}\cdot\vec{\omega}' \mid \, d\omega' \tag{2.10}$$

where $\rho_d(x)$ is the constant diffuse BRDF. The reflected radiance can be calculated as multiplication of irradiance and a constant diffuse term.

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \rho_d(x)E(x) \tag{2.11}$$

$$E(x) = \int_\Omega L_i(x,\vec{\omega}') \mid \vec{n}\cdot\vec{\omega}' \mid \, d\omega' \tag{2.12}$$

The irradiance is calculated at irradiance cache records positions and later interpolated. To calculate the irradiance, the incident light integral from equation 2.12 is numerically evaluated by Monte Carlo integration similarly to path tracing. The irradiance at point x is calculated by algorithm 2.1. **x** is the position, where the irradiance is evaluated and **n** is the surface normal in this position.

The interpolation of nearby irradiance cache records to a specific surface point x is calculated as a weighted average by the Equation 2.13. $E_i(x)$ is the irradiance stored in the i-th cache record, $w_i(x)$ is the weight calculated by Equation 2.14, and $S(x)$ is the set of all irradiance cache records contributing to the point x defined as $S(x) = \{i; w_i(x) > 0\}$.

$$E(x) = \frac{\sum_{i \in S(x)} E_i(x) w_i(x)}{\sum_{i \in S(x)} w_i(x)} \tag{2.13}$$

```
1  function IrradianceCaching(x,n)
2  begin
3      if one or more irradiance values can be used for interpolation at x then
4          return irradiance interpolated from the cached values.
5      else
6          Compute new irradiance value by hemisphere sampling.
7          Store the value as a new record in the cache.
8          return the new irradiance value.
9      end
10 end
```

Algorithm 2.1: Irradiance Caching [91].

The contribution of the i-th cache record to the irradiance at point x is given by the weight $w_i(x)$. This weight can be calculated by equation 2.14, or the alternative equation 2.15, proposed by Tabellion [173]. The weight function from equation 2.15 has the advantage that it gives smoother interpolation results due to its maximum value 1 in comparison to infinity in case of equation 2.14. The weight $w_i(x)$ depends on the distance between the cache record position and x, the difference in normals, the maximum allowed interpolation error α, and R_i, which is the distance to surfaces visible from the cache record position x_i. This distance can be computed as a harmonic mean or, alternatively, the minimum of the ray lengths in the hemisphere sampling [91]. The κ in equation 2.15 is the user-defined constant and it determines the overall caching accuracy. It is inversely proportional to the maximum allowed interpolation error α used in the original formulation (Equation 2.14). To perform a fast search of nearby cache records during rendering the cache records are stored in a spatial octree data structure.

$$w_i(x) = \frac{1}{\frac{\|x - x_i\|}{R_i} + \sqrt{1 - n \cdot n_i}} - \frac{1}{\alpha} \tag{2.14}$$

$$w_i(x) = 1 - \kappa \max\left\{ \frac{\|x - x_i\|}{R_i/2}, \frac{\sqrt{1 - n \cdot n_i}}{\sqrt{1 - \cos 10^\circ}} \right\} \tag{2.15}$$

Irradiance Cache Splatting. The irradiance interpolation problem can be efficiently approached by irradiance cache splatting algorithm [45]. Instead of starting from a scene point and looking for the nearby records, this method starts from a record and accumulates its contribution to the splat buffer. In each pixel of this buffer, to which the cache record contributes, the sum of weighted irradiance contributions and the sum of weights are accumulated. The irradiance can be efficiently splatted to the splat buffer by utilizing GPU rasterization. Firstly a quad with specific parameters, including position, normal, irradiance, and R_i, is sent to the rasterization pipeline. This quad is scaled in a vertex shader to fit the area of contribution of irradiance cache

record. The radius of this area can be derived from the interpolation equation (Equation 2.14) as $r_i = \alpha R_i$. Later, the irradiance contribution and weight are accumulated for each pixel in a fragment shader using equation 2.14 or 2.15. Finally, after the splatting step finishes, the accumulated irradiance is divided by the accumulated weight and multiplied by the diffuse albedo to calculate the outgoing radiance for each pixel. The irradiance splatting algorithm is depicted in Figure 2.11.

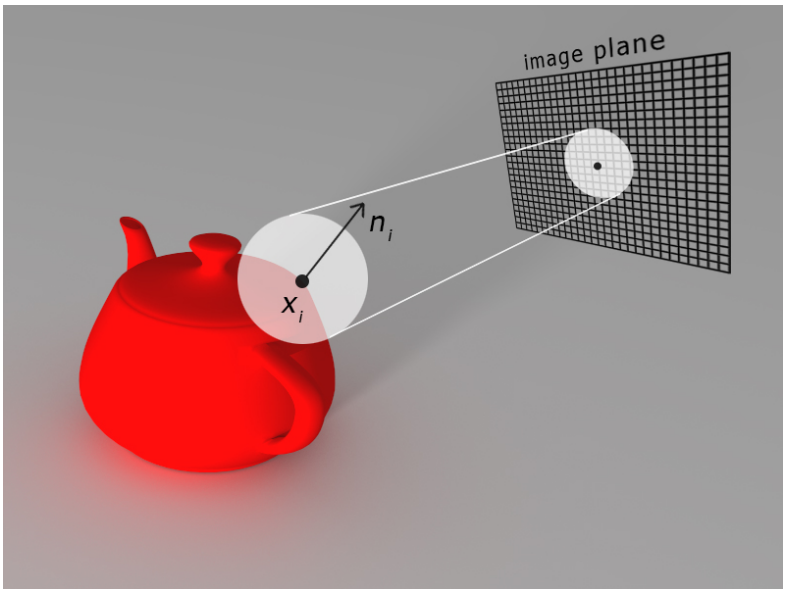

Figure 2.11: The sphere around position x_i of the record i is splatted onto the image plane. For each pixel within the projection of the sphere, the contribution of record i is accumulated into the irradiance splat buffer [91].

Improvements of Irradiance Caching. The quality of irradiance interpolation can be improved by using irradiance gradients proposed in [191] and [92]. Křivánek et al. proposed the improvement of irradiance interpolation by introducing neighbor clamping [89]. The authors clamp the average distance of nearby geometry R_i according to the R_i values of neighboring cache records. Arikan et al. [2] proposed a method decompositing the radiance field into near-field and far-field components, that can be approximated separately. Improvements of the error metric for irradiance caching, to allow for an efficient placement of cache records, were proposed in [69, 158].

The sampling of irradiance records is done in screen space and is therefore dependent on the camera pose. This dependence can cause temporal flickering in animations because new samples are permanently introduced. The temporal coherence in irradiance caching was addressed in [166] by introducing anchor points and in [44] by proposing temporal irradiance gradients. The work of Brouillat et al. [14] proposes the combination of irradiance caching and photon mapping. They compute a view-independent irradiance cache from a photon map. An extension

of irradiance caching to simulate glossy reflections by radiance caching was presented in [90]. The spherical harmonics are used in radiance caching to store directionally dependent radiance information for the simulation of glossy surfaces. An efficient way of storing and preconvolving the radiance is to use mip maps [154]. The cost of evaluation of spherical harmonic coefficients per pixel is then exchanged for a single constant-time lookup per pixel to the preconvolved mip maps. A detailed overview of irradiance caching can be found in [91] and [133].

2.3 Interactive Global Illumination

The calculation of global illumination is essential for high-quality computer generated images. Therefore, the need for GI appears also in augmented reality, computer games, simulation, design and other areas. One significant constraint in these fields is to preserve the interactive frame rates. Despite the massive parallel computational power of modern graphics hardware, the accurate solution of the rendering equation is still not available in real time. Therefore, approximative solutions of GI were proposed. Many interactive techniques for GI calculation were developed in previous research. The majority of them is based on rasterization, because modern graphics hardware is optimized for rasterization based rendering. The disadvantage of these methods is the rendering error caused by approximating the light transport. Different categories of methods are presented in this section. A good overview of interactive GI methods can be found in [146].

2.3.1 Ray Tracing Methods

With the increasing computational power provided by modern graphics hardware, some of the ray tracing based algorithms become available in interactive frame rates. Whitted ray tracing, calculating accurate reflections and refractions, can run in real time on modern GPUs. However, the full global illumination solution cannot be calculated in real time yet.

Interactive Ray Tracing. Early research on interactive ray tracing used computer clusters to provide the required computational power. Wald et al. [186] proposed an interactive ray tracing approach for rendering dynamic scenes. Their method separates the scene into independent classes of objects with common properties concerning dynamic updates. Three classes of objects are distinguished: Static objects with static acceleration data structures, objects undergoing affine transformations are handled by transforming rays, and objects with unstructured motion have the acceleration data structure rebuilt whenever necessary. Lauterbach et al. [98] proposed an efficient approach for interactive ray tracing of deformable or animated models. Their method dynamically updates the bounding volume hierarchy (BVH) acceleration structure. BVH used in this approach consists of axis-aligned bounding boxes. In practice, the rebuilding of BVH can be expensive, so these computations can be minimized by measuring BVH quality degradation between successive frames. The authors also applied the ray coherence techniques. Their implementation allows an interactive rendering on a single computer.

The first interactive ray tracing approach without using an acceleration data structure was proposed by Mora [116]. This approach uses a divide-and-conquer strategy to calculate intersections between rays and triangles efficiently. No spatial data structures are stored when spatial

subdivisions are performed, and intersections inside the scene are computed directly. Memory management is minimal and deterministic, which simplifies the ray tracing engineering, as spatial subdivision data structures are no longer considered in the graphics pipeline. This is possible with a modification of Whitted's naive ray tracing algorithm by using a divide-and-conquer approach, and by having a sufficient collection of rays in order to reduce the complexity of naive ray tracing. The divide-and-conquer algorithm consists of two nested loops computing every possible ray-triangle intersection in the scene. The intersections are calculated only on subsets of rays with subsets of triangles, which are determined by a divide-and-conquer scheme using spatial subdivisions. Firstly, the algorithm compares the number of primitives and the number of rays with two arbitrary constants. If one of the two comparisons is true, the algorithm just uses the naive algorithm. Otherwise, the space is subdivided and a recursive call is made for each subspace only including the primitives and rays intersecting the given subspace. This algorithm can calculate the ray tracing result interactively without using any acceleration data structure. A GPU implementation of ray tracing was proposed by Purcell et al. [139]. The authors explained how ray tracing can be mapped to graphics hardware and analyzed the performance of a ray casting implementation on the GPU.

Interactive Ray Tracing Frameworks. Several ray tracing frameworks have been developed which enable easy and fast implementation of ray tracing techniques on the hardware with parallel computational capabilities. A framework utilizing parallel CPU coprocessors is Embree [197]. It is designed for Monte Carlo ray tracing algorithms, where the vast majority of rays are incoherent. Moreover, this framework efficiently utilizes the wide SIMD vector units of CPUs and CPU coprocessors.

A fast ray tracing framework utilizing parallel GPUs is OptiX [128], which offers predefined structures for a programmable ray tracing pipeline. The user can fully implement the techniques for ray generation, ray-scene intersection, geometry representation, and other phases of the ray tracing pipeline. The user-provided code is compiled to the GPU-executable code. The following programs have to be specified by the user:

- **Ray generation** are the entry programs into the ray tracing pipeline. Many instances of these programs are created on the GPU to calculate the result per each sample. The user can define his own sampling strategy and can distribute the rays according to the requirements.

- **Intersection** programs are used to implement the ray-geometry intersection tests. As the acceleration structures are traversed, the system will invoke an intersection program to perform the geometric query. The program determines if and where the ray intersects with the object and additional attributes can be calculated. An arbitrary number of attributes may be associated with each intersection.

- **Bounding box** programs calculate the bounds of each geometry primitive. These bounding values are used to automatically build acceleration data structures by OptiX.

- **Closest hit** programs are invoked when the closest intersection of a ray with the geometry is found. Closest hit programs are used for shading calculations and interaction of light

with the surface material. The tracing of new rays can be invoked from closest hit program to calculate additional radiance values. For example, reflected radiance can be calculated by tracing new radiance rays, or visibility can be tested by casting shadow rays.

- **Any hit** programs are the second variant of hit programs. The user can decide if the closest hit, or any hit will be searched for a specific ray type. Any hit programs are invoked when the intersection of ray and geometry is found, similarly to closest hit programs. The any hit program can optionally terminate the ray using the built-in function, which will stop the traversal for a particular ray. This type of hit program is usually used to detect visibility by casting shadow rays. By terminating the traversal after any hit is found, the efficiency is increased.

- **Miss** programs are called when the ray does not intersect any geometry. They can be used to implement a background color or environment map lookup.

- **An exception** program is called when an exception occurs. The user can define his own visualization of exceptions, for example by writing a special color value to an output pixel buffer.

- **Selector visit** programs expose programmability for coarse-level node graph traversal. For example, the level of geometric detail may vary for parts of the scene on a per-ray basis.

OptiX achieves a high ray tracing speed due to its efficient utilization of massive parallel GPUs. OptiX is also used for the implementation of algorithms presented in this PhD thesis.

Interactive Photon Mapping. An interactive combination of photon mapping and ray tracing was proposed by Wang et al. [188]. In their work, the Kd-tree is dynamically calculated on the GPU [199] for every frame. Their method exploits the spatial coherence of illumination to reduce sampling cost. Samples are taken sparsely and the distribution of sample points conforms with the underlying illumination changes. Moreover, they presented the parallel implementation of adaptive sampling suitable for the GPU. Another interactive photon mapping approach was presented by Fabianowski and Dingliana [34]. The authors extended the concept of photon differentials [155] to efficiently handle diffuse reflections. Moreover they used density estimation with variable kernel bandwidths. An efficient BVH construction for acceleration of the ray tracing was reused to build photon maps that enable the fast retrieval of all hits relevant to a shading point. The authors also presented a heuristics that automatically tunes the BVH build's termination criterion to the scene and to illumination conditions. A GPU implementation of photon mapping was presented by Purcell et al. [140]. Their implementation uses breadth-first photon tracing to distribute photons using the GPU. The photons are stored in a grid-based photon map that is constructed directly on the GPU using one of two methods. The first is a multi-pass method which uses fragment programs to directly sort the photons into a compact grid. The second method uses a single rendering pass combining a vertex program and the stencil buffer to route photons to their respective grid cells, producing an approximate photon map. Moreover,

the authors presented an efficient method for locating the nearest photons in the grid. A real-time photon mapping approach, utilizing the OptiX framework, was presented in [56]. In this approach, the spatial hashing method was used to store and retrieve a photon map. The density of photon maps was reduced by storing photons selectively based on a local density criterion, while preserving the correct illumination. At some selected locations with high photon density, the irradiance was precomputed to speed-up the final gathering stage. An efficient photon mapping implementation combining GPU rasterization and CPU ray tracing was proposed by McGuire and Luebke [112]. They used photon volumes rasterization in order to avoid costly KNN queries. An interactive photon mapping implementation running on multiple computers is shown in [55].

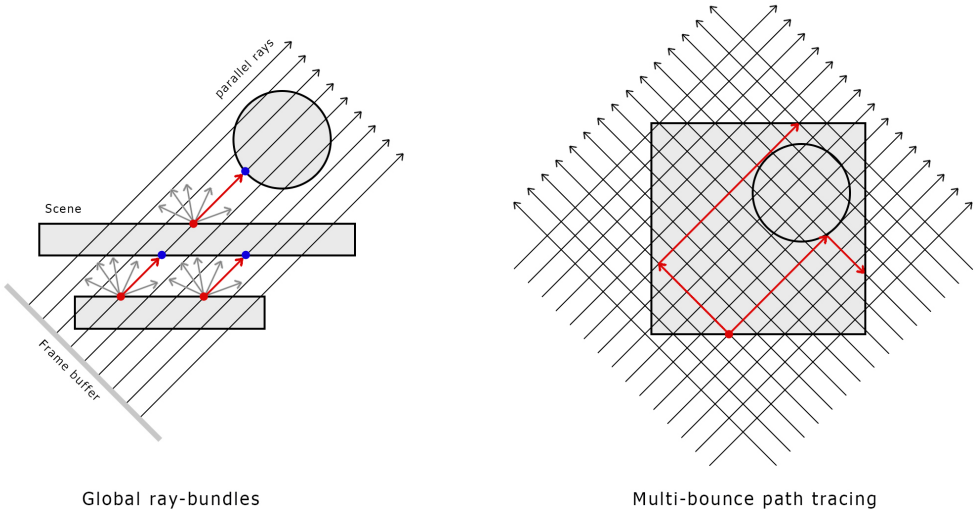

Figure 2.12: Left: Global ray bundles calculated by rasterization. All rays have the same global ray direction. Right: Multi-bounce path tracing by reusing omnidirectional global ray bundles [176].

Interactive Path Tracing. Due to the high computational cost of the path tracing algorithm, the converged rendering result from this algorithm is not achievable in interactive frame rates on current commodity hardware. Despite of this limitation, previous research has aimed to make path tracing interactive. Interactive path tracing methods either approximate the light transport, introduce constraints, or produce a biased GI solution. Tole et al. [177] proposed a progressive global illumination method based on path tracing. The authors presented a shading cache, an object-space hierarchical subdivision mesh with lazily computed shading values at its vertices. The interactive progressive rendering is generated with the help of the shading cache using hardware-based interpolation and texture mapping. An image space sampling scheme refines the shading cache in regions that have the highest interpolation error or those that are most likely to be affected by object or camera motion. Another progressive path tracing approach was presented by Novák et al. [121]. They proposed a path regeneration approach, in which the di-

vergence in execution flow was minimized to ensure full utilization by intelligently regenerating the paths. The convergent execution, suitable for modern graphics hardware, can be achieved by using the regeneration algorithm. The divergence is often created by termination of some threads. In path regeneration the terminated threads are restarted and they compute new paths, leading to higher overall performance. An approximative real-time bidirectional path tracing was proposed by Tokuyoshi and Ogaki [176]. The authors approximate path tracing from a camera by using global ray bundles. By focusing on a single global direction, the visibility for all fragments in a scene is computed in parallel as shown in Figure 2.12. This can be done by rendering the scene from the sample direction using a parallel projection, similar to rendering a shadow map from a directional light source. In this pass, fragment data (depth, normal, albedo and roughness) are saved into a buffer. This buffer is used in the next pass to calculate the radiance for each visible point. The concept of global ray directions was proposed by Szirmay-Kalos and Purgathofer [171]. The next GI method based on the concept of global ray directions is described by Hachisuka in [57]. The author describes a two-pass method utilizing the graphics hardware via rasterization, which produces a high-quality result.

2.3.2 Instant Radiosity

Instant radiosity was first proposed by Keller [80]. This method approximates the light reflected from surface points by creating virtual point lights (VPL) in the scene. In the next light bounce the light contribution is calculated from every VPL with respect to visibility. The positions of VPLs are calculated by quasi-random walk [79]. Instant radiosity calculates multiple bounces of global illumination in scenes with the assumption of diffuse surfaces by the following process. Firstly, the distribution of VPLs in the scene is calculated. Then, a shadow map is created for each VPL to calculate visibility. The indirect illumination is obtained by summing up the contribution of all VPLs. The contribution of one VPL is calculated by rendering the scene illuminated by this VPL from the camera's viewpoint with respect to visibility by using the corresponding VPL's shadow map. Finally, the indirect illumination is added to the direct illumination. The instant radiosity method employs rasterization on the GPU to calculate the indirect illumination and it uses the shadow mapping technique to quickly calculate visibility in the scene.

An extension of instant radiosity uses imperfect shadow maps (ISM) [148] to approximate visibility and to speed up the rendering. This method rapidly improves the time needed for shadow maps generation from the positions of VPLs by roughly approximating the geometry of the scene by a point cloud. Moreover, the shadow maps of all VPLs are calculated in parallel into one texture. The pull-push approach [52] is used to fill the holes in shadow maps caused by the point cloud representation. The ISM method requires a scene preprocessing to create the point cloud representation. In this step each point is created by randomly selecting a triangle with a probability proportional to its area and then picking a random location on the selected triangle. Barycentric coordinates of this point are calculated to handle dynamic scenes and to enable normal and reflectance calculation.

In addition to ISM, the imperfect reflective shadow maps (IRSM) can be used to calculate multiple diffuse and glossy bounces. This approach generalizes ISMs in the same way classic shadow maps generalizes to reflective shadow maps [26]. Given the VPLs for the first bounce, the VPLs for the second bounce can be created as follows. In the first step, the first bounce VPLs

are created and the imperfect reflective shadow maps are rendered from the point based scene representation. In the second step, the second bounce VPLs are created by importance sampling on rendered shadow maps and the illumination of these second bounce VPLs is rendered into the imperfect shadow maps. By using this procedure, multiple bounces can be created iteratively. Finally the contribution of all final bounce imperfect shadow maps is accumulated and the resulting image is produced.

2.3.3 Screen Space Methods

A screen space representation, including the surface position, normal and additional attributes per each pixel, is an efficient approximation of a 3D scene. This representation can easily be created by rasterization and it can make the complexity of rendering almost independent on the triangle count. The geometry is represented as 3D points rasterized by the GPU from a selected viewpoint. This discrete set of points with their attributes can represent the simplified representation of the scene with direct light and its visibility calculated by rasterization. Usually, screen space methods consist of two or more steps, where the scene is rendered from the position of the light source and the attributes of visible points are calculated in the first step.

Reflective Shadow Maps. In order to simulate the light transport in screen space, the original shadow mapping [196] technique can be extended to reflective shadow maps [26]. In this approach, the additional attributes are calculated in the shadow map rendering step. The additional attributes are stored in the render targets and they contain the information about the surface normals, the surface positions, and the radiant flux at the surface point. The illumination is calculated in the following two steps. In the first step, the scene is rendered from the light position and all attributes are calculated. In the second step, the scene is rendered from the camera position and the contribution of the reflected light to the rendered point is calculated by using the stored reflective shadow map values. An assumption that one-bounce reflected light comes only from the points stored in shadow maps is correct, because only this part of the scene is directly lit. This method suffers from its approximation error because it ignores visibility when calculating the contribution of reflected light.

Screen Space Ambient Occlusion. Screen space ambient occlusion (SSAO) is a method which calculates local visibility and the amount of occlusion using a screen space representation [10, 115, 160]. In this method the local shadowing calculation is based on the local occlusion of a point by nearby objects. The nearby objects are represented as sampled depth values in screen space. The ambient occlusion method creates a soft shadowing effect in the locations close to the nearby objects. SSAO was later extended to screen space directional occlusion (SSDO) [149]. SSDO extends the SSAO by directional and one-bounce light transport calculated in screen space which enhances the visual plausibility of the produced image.

Image Space Photon Mapping. A photon mapping extension, efficiently calculating GI, using screen space representation is called image space photon mapping (ISPM) [112]. In this method, the fact that the initial and final light bounces are the most expensive is exploited and

these steps are calculated by rasterization on the GPU. Other bounces are calculated by photon tracing on the CPU. Instead of the standard final gathering step the ISPM rasterizes photon volumes. This method can calculate high frequency indirect illumination like caustics and it handles dynamic scenes.

2.3.4 Volumetric Methods

Cascaded Light Propagation Volumes. Cascaded light propagation volumes is a fast algorithm for GI calculation based on light transport in a volumetric grid [74]. This method divides the 3D volume of the scene into a 3D grid and propagates the light through the cells of this grid. This approach is based on the discrete ordinate method [17], which discretizes the quantities in the radiative transfer equation in space and orientation. Moreover, the cascaded light propagation volumes approach combines lattice-based light propagation with the techniques for interactive GI like reflective shadow maps [26] and instant radiosity [80].

Two grids are used for calculating GI in cascaded light propagation volumes. The first one is used to store the indirect and low frequency direct lighting. The second grid is used as the coarse volumetric approximation of scene geometry. This grid is used for a fuzzy light blocking when traversing the volume. Indirect lighting calculation is done in the following four steps. In the first one, the light propagation volume (LPV) is initialized. In this step, the low frequency lighting is stored in the low number of VPLs, which use reflective shadow maps. The irradiance information of this light, sampled in the VPLs, is then converted into the spherical harmonics [65] and stored in the cells of the light propagation volume to be able to initialize the light propagation through the scene. In the second step, the scene is sampled from the camera and from multiple reflective shadow maps to create a coarse volumetric representation of the blocker geometry. In the third step, the light is propagated in the volume grid. In the last step, the scene is rendered with the indirect lighting contribution calculated from LPV. The advantages of LPV are that it calculates approximative GI in a few milliseconds, it uses a regular grid for the light propagation, and it can also simulate single-scattering participating media. On the other hand, LPV is limited to diffuse surfaces.

Voxel Cone Tracing. Another interactive GI algorithm, utilizing volumetric representation, is voxel cone tracing proposed by Crassin et al. [25]. The authors use a hierarchical voxel octree representation, generated and updated on the fly from a regular scene mesh, coupled with an approximate voxel cone tracing that allows for a fast estimation of visibility and incoming light energy. This approach can calculate diffuse and glossy reflections in a visually acceptable accuracy. A voxel octree representation is built and updated interactively by utilizing rasterization hardware. This makes the rendering step almost independent from the mesh representation and allows for a real-time performance. If the voxel structure is updated, the rendering consists of three main steps:

1. Firstly, the incoming radiance (energy and direction) is injected from dynamic light sources into the leaves of the sparse voxel octree hierarchy. This is done by rasterizing the scene from all light sources and splatting a photon for each visible surface fragment.

2. In the second step, the incoming radiance values are filtered into the higher levels of the octree (mipmap). A compact Gaussian-Lobes representation is used to store the filtered distribution of incoming light directions which is done efficiently in parallel by relying on a screen space quad-tree analysis.

3. Finally, the scene is rendered from the camera. An approximate cone tracing is employed to perform a final gathering, sending out cones over the hemisphere to collect illumination distributed in the octree. Then, the direct and indirect illumination are summed.

2.3.5 Precomputed Radiance Transfer

Various kinds of precomputation can be used to speed up the rendering. One of them is the precomputed transfer from direct to indirect illumination. This idea was firstly proposed by Sloan et al. [164] and it was later extended by Wang et al. [189]. In the latter, the direct lighting and precomputed diffuse indirect transfer are represented by a spectral mesh basis function set derived from an arbitrary scene model. The spectral mesh basis was proposed by Karni and Gotsman [76] for the purpose of geometry compression. They show, how the 3D geometry can be projected onto the set of basis functions and then stored as coefficients of these functions.

A mathematical equation of rendering with the precomputed radiance transfer can be derived by reformulating the rendering equation in the following way. Denote the function $T(x_o, x_i)$ as the differential form factor between two surface points x_i and x_o. The form factor is the proportion of all radiation that leaves one surface and reaches another one. The function $V(x_o, x_i)$ returns the visibility between two surface points.

$$T(x_o, x_i) \equiv V(x_o, x_i) \frac{\cos\theta_i \cdot \cos\theta_o}{|| \, x_i - x_o \, ||^2} \tag{2.16}$$

Assuming diffuse surfaces, the reflection part of the rendering equation can be rewritten:

$$L_r(x_o) = f_r(x_o) \int_{\mathcal{M}^2} L_i(x_i) T(x_o, x_i) \, dA(x_i) = f_r(x_o) E(x_o) \tag{2.17}$$

where $L(x_i)$ denotes the incoming radiance at point x_i, f_r is the diffuse albedo of the surface, and $E(x_o)$ is the total irradiance in point x_o. The domain of integration $\mathcal{M}^2$ is the entire surface area of the scene. In the static scenes, term T is purely geometric and it can be precomputed, projected into the spectral mesh basis, and stored as a matrix. Then the equation 2.17 can be written in the matrix form:

$$\mathbf{L_r} = \rho \cdot (\mathbf{T} \times \mathbf{L}) \tag{2.18}$$

where ρ is a vector of surface albedos. $\mathbf{T}$ denotes the precomputed matrix $T(x_o, x_i)$ and $\mathbf{L}$ is the vector $L(x_i)$. Both $\mathbf{L}$ and $\mathbf{T}$ can be projected onto the spectral mesh basis achieving the lighting vector $\mathbf{L^s}$ and the transport matrix $\mathbf{T^s}$. The radiance L_r is still computed in the same way because the spectral mesh basis S is orthonormal.

$$\mathbf{L_r} = \rho \cdot (\mathbf{T} \times \mathbf{L}) = \rho \cdot (\mathbf{T^s} \times \mathbf{L^s}) \tag{2.19}$$

$\mathbf{L^s}$ is the dynamic direct lighting computed on the fly and projected onto the spectral mesh basis S. $\mathbf{T^s}$ represents the transfer from direct to indirect lighting projected onto the spectral mesh basis. This transfer matrix is precomputed and the coefficients of the basis function obtained by projection are stored. It allows the efficient storage of radiance transfer and fast calculation of indirect illumination from direct illumination. The precomputed radiance transfer offers a fast GI solution. However, its limitation is that geometry has to be static.

2.3.6 Final Gathering

As described in Section 2.2.3, final gathering is the second step of photon mapping. Generally, final gathering refers to the process of calculating the amount of indirect illumination at a surface point. The final gathering step contains the integral part of rendering and it is, therefore, computationally expensive.

Irradiance Caching. An efficient approach for final gathering is irradiance caching, described in Section 2.2.4. An efficient GPU algorithm for interactive irradiance caching was proposed by Gautron et al. [45]. This algorithm utilizes splatting by GPU rasterization. Further details about irradiance cache splatting can be found in Section 2.2.4. An interactive combination of irradiance caching and photon mapping was presented by Brouillat et al. [14]. In this method a refined, view-independent irradiance cache is calculated from the photon map. GI is then rendered interactively using radiance cache splatting.

Point Based Global Illumination. Christensen [20] proposed a CPU based method which speeds up final gathering. This method uses a point cloud (surfel) representation of the directly illuminated geometry. Surfels are the directly illuminated sample points, from which the indirect illumination can be calculated. Here, the similarity with the screen space methods (section 2.3.3) can be seen. Surfels are organized in an octree and energy from each surfel is approximated with spherical harmonics [65]. The light from the surfels is added using three degrees of accuracy: ray tracing, single disk approximation, and clustering.

Micro-Rendering. An efficient GPU approach for final gathering is called micro-rendering [147]. This method traverses and rasterizes a hierarchical point-based representation of the scene. Importance-warped micro buffers are used, which allow BRDF importance sampling. This method is capable of calculating the final image in dynamic scenes, or it can be used in combination with a high-quality rendering algorithm like photon mapping to create interactive walkthroughs. Finally, this method is capable of calculating multi-bounce GI.

2.3.7 Comparison

Different kinds of approximation were used in interactive GI methods to speed up the rendering. Generally, these methods can be divided into two main groups. The first one is the group of high-quality rendering algorithms, which can reproduce almost any kind of light paths with high fidelity. These are mainly ray tracing based algorithms described in Section 2.3.1. A disadvantage of high-quality methods is their high computational cost. Therefore, they are impractical

for real-time applications. The second group of algorithms present a higher level of approximation with the benefit of fast rendering and the possibility of achieving real-time performance. These methods usually assume a low-frequency diffuse lighting and perform a sparse sampling. For example, instant radiosity, described in Section 2.3.2, can efficiently simulate light bounces using sparse samples at the VPLs locations. Screen space methods utilize a discrete approximative representation of the scene or lighting stored in a shadow map. This kind of approximation leads to visually plausible one-bounce indirect illumination, but it can cause artifacts due to the visibility approximation. A well suited method for GI in real-time applications is cascaded light propagation volumes discussed in Section 2.3.4. This method can render a multi-bounce solution of low-frequency GI in high frame rates. Moreover, it is capable of simulating light transport in the participating media. Some algorithms take different constraints into consideration to allow fast GI calculation. For example, the assumption of static scenes allows the precomputation as it can be seen in the precomputed radiance transfer. This method enables fast rendering by transfering the direct light to indirect light. However, the assumption of a static scene makes this method impractical in many situations. Generally, in interactive GI algorithms the trade-off between accuracy and rendering speed can be seen.

The main goal of this thesis is to provide a high-quality result of a composited image to increase the visual coherence of AR perceived by the user. Therefore, we focus on high-quality ray tracing based rendering techniques. An important requirement of AR is real-time performance. Due to this requirement, the algorithms have to be improved to achieve interactivity. In order to overlay real scenes with virtual objects in AR, fast compositing is needed.

2.4 Compositing

A main property of augmented reality is the combination of virtual objects with real scenes. Many AR applications simply overlay rendered virtual objects over a real image. However, this simple approach is insufficient for creating the proper light interactions in the scene. The light interactions of virtual and real objects are important for visual coherence in AR. Therefore, advanced composting has to be used.

High-quality rendering and compositing of virtual objects into the video of a real scene requires two solutions of GI. First, the rendering of a reconstructed model of the real scene (I_r) is required. Second, the model of the real scene is rendered together with the virtual models (I_m). As described by the differential rendering equation (Equation 2.20) [27, 77], these solutions can be combined with the real camera image (I_c) to obtain a final composited image (I_f):

$$I_f = M \odot I_m + (1 - M) \odot (I_c + I_m - I_r) \tag{2.20}$$

I_m is the rendered result of both real and virtual objects (mixed radiance), while I_r is the rendering result of real objects only (real radiance). The I_c is the source image captured by the camera. M is a mask which controls the amount of blending between differential radiance ($I_m - I_r$) and mixed radiance (I_m). Differential radiance is only used on real objects to introduce light changes caused by inserted virtual objects. Therefore, the values of M are 1 at pixel locations of virtual objects and 0 elsewhere. However, at object edges, values between 0 and 1

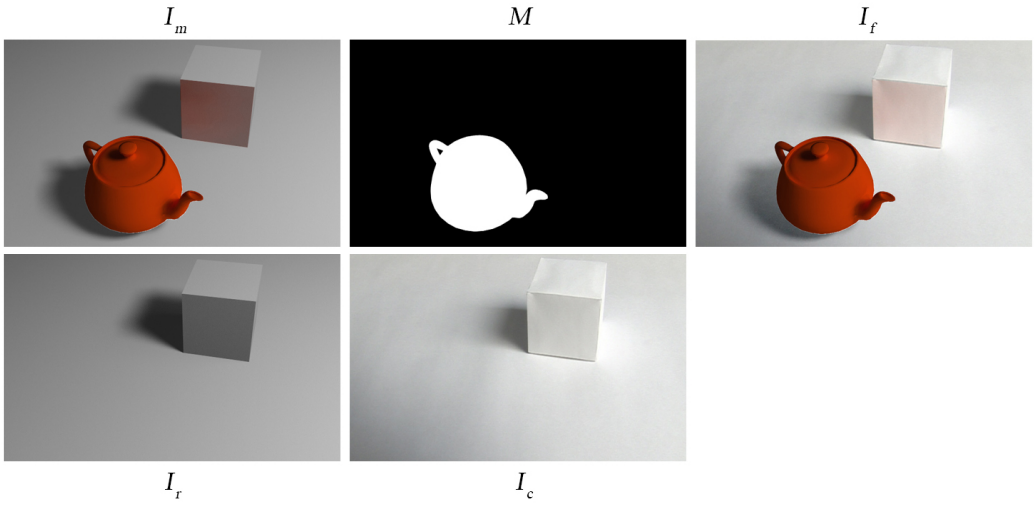

Figure 2.13: Terms of differential rendering from Equation 2.20. A real white cube on a real white table and a virtual teapot are visible in the image. Note the indirect red light introduced on the real white cube by adding the virtual teapot.

indicate blending between virtual and real content. Such a blending appears especially in depth of field rendering and for antialiasing. The terms of Equation 2.20 are depicted in Figure 2.13.

High dynamic range (HDR) rendering is essential for correct image synthesis. Common capturing and displaying devices work with low dynamic range (LDR) using only eight bits per pixel per color channel to store the light intensity information. This limited information is insufficient for reproducing the range of illumination intensities in natural scenes. Thus, the rendering has to be done in HDR to produce natural illumination effects. The importance of HDR rendering was highlighted by Debevec [27]. The limited capabilities of capturing and displaying devices pose a problem on using HDR rendering for AR applications. There are several solutions to approach this problem:

- The HDR radiance can be calculated from LDR images with multiple exposures. An approach for HDR radiance recovery from multiple images was proposed by Debevec and Malik [28]. They first reconstruct a camera response function from input images. In the second step, the HDR radiance is recovered using the inverse of the camera response function. This approach has the disadvantage that the scene and camera have to be static during multiple exposures capturing. Thus, this method is not practical for real-time applications like AR.

- The use of specialized HDR hardware, including HDR cameras and HDR displays, can completely avoid the need for LDR images. However, this special hardware is not easily available and is mostly used for research purposes only.

- The third solution is to use a conversion from LDR to HDR and vice versa. The HDR radiance values can be estimated from LDR by inverse tone mapping [8]. The authors propose

an approximate solution to reconstruct HDR from LDR that uses the mediancut to find the areas considered to have high luminance. Subsequently, they apply a density estimation to generate an expanded radiance map in order to extend the range in the high luminance areas using an inverse photographic tone reproduction operator. After the rendering step, the HDR radiance has to be converted back to LDR to be displayed in common display devices. Tone mapping methods can be used for this purpose. Several tone mapping operators were presented in past researches [4, 30, 31, 35, 130, 145]. The main functionality of them is to compress the high dynamic range values to the limited low dynamic range and to preserve details. These operators can work either globally or locally. Global operators process all pixels in the image with the same parameters, while local operators process a pixel depending on its neighboring pixels.

2.5 Global Illumination in Augmented Reality

A pioneering work on global light transport for AR, introducing differential rendering, was presented by Fournier et al. [36]. It was later extended by an image-based lighting approach to simulate the natural appearance of virtual objects [27]. Caustics and accurate refractions for AR were firstly presented in differential photon mapping [50]. In this approach, an efficient simulation of caustics and shadows was shown. Moreover, an image reprojection technique was proposed to obtain real radiance coming through refractive surfaces. Another approach for creating high-quality augmentations of photographs with virtual objects is to use user-driven light estimation, a quick scene modeling step, and physically based rendering [77]. In this approach, an offline light transport calculation is used to produce the final composited image. These methods are capable of creating high-quality final results, but do not run in real time. However, interactivity is required by AR applications. Thus, the advantage of the algorithms presented in this thesis (Chapters 3 and 4) is that they run at interactive frame rates while providing a high-quality final result.

Approximative rasterization based methods for light transport in AR can achieve real-time performance for a cost of an approximation error. Diffuse GI in real-time AR was proposed by Knecht at al. [85, 86] in a differential instant radiosity approach. They extended imperfect shadow maps [148] to work in AR and used single-pass differential rendering to create the final composite image. Their system is capable of simulating diffuse global illumination, including color bleeding between real and virtual objects, at interactive frame rates. This approach achieves real-time performance, but is limited to diffuse global illumination. Lensing and Broll [104] proposed a method for calculating fast indirect illumination in AR scenes with automatic 3D scene reconstruction. Another approach utilizes the light propagation volumes algorithm to calculate GI in real-time AR scenarios using volumetric grid. A modified delta light propagation volumes method [38] is used to calculate the global light interreflections between virtual and real objects. Instead of propagating the absolute radiance through the volume, the change of illumination, caused by the introduction of synthetic objects, is propagated. This approach can simulate the indirect illumination calculation using just a single rendering step. The disadvantages of mentioned approaches are that they introduce approximation errors and are limited to diffuse global illumination. Rasterization based rendering of specular GI effects like

refraction, reflection and caustics in AR was investigated in [87, 136]. The efficient GI method for AR, which can calculate diffuse, glossy, and specular global illumination was recently proposed in [39]. This method extends voxel cone tracing [25] to delta voxel cone tracing. The delta radiance estimation is used instead of radiance estimation to provide differential radiance. The volumetric structure is used to calculate the fast approximation of the indirect light reflection. The delta voxel cone tracing is suitable for AR due to its fast rendering speed. The limitation of this method are the voxelization artifacts visible in mirror surfaces.

Environment maps of real scenes can be used in AR rendering as light sources to increase the visual realism of the final rendering. The cosine weighted contribution from the environment map has to be calculated in order to simulate the diffuse, or glossy reflection of real light on artificial objects. This technique, called irradiance environment mapping, is described in [141]. A real environment map for image-based lighting of virtual objects was used in [1, 131, 132]. The disadvantage of these techniques is that they ignore visibility in the irradiance calculation and therefore introduce an approximation error. The novel algorithms, presented in this thesis, have the advantage that the interreflections between virtual and real worlds are calculated in both ways in a physically based fashion.

The irradiance volume [49] can be used to precalculate indirect illumination inside an AR scene for different directions of direct illumination [51]. The irradiance is stored in spatial locations arranged in a grid using spherical harmonics. A high number of irradiance volumes is calculated for all possible directions of incoming direct light. Indirect light could then be rendered dynamically in real time depending on incoming direct light. Moreover, a correct near-field illumination can be calculated using the predefined model of surrounding geometry in combination with a fish-eye lens camera. A disadvantage of this method is that a vast precomputation of irradiance volumes is required and big objects have to be static to not change the irradiance.

Believable material simulation can increase the amount of realism in AR. Pessoa et al. [132] proposed an approach for photorealistic rendering in AR based on rasterization. They use an extended version of the Lafortune spatial BRDF model [94] to properly simulate material properties. Additionally, they employ the irradiance environment mapping [141] to simulate GI coming from the real world to virtual objects. However, their solution cannot simulate GI coming from virtual objects to real ones. Furthermore, a static environment image is used in their solution and therefore no dynamic movement of reflected real objects or lighting change could be simulated. The rasterization based approaches usually do not accurately simulate the light paths reflected from specular surfaces and they use a high amount of approximation in order to achieve high rendering speed.

An approximation of self-shadowing in AR, calculated by ambient occlusion, was used by Franke and Jung [40]. They proposed a material reconstruction technique which employs genetic algorithms. Multi-pass rendering for AR was proposed by Agusanto et al. [1]. The authors used irradiance environment maps to simulate GI coming from real light to the virtual objects. However, their system does not simulate light reflected from virtual objects affecting the real scene.

Special hardware devices can be combined with the GI calculation in AR to enrich reality with artificial illumination. For example, the approach proposed by Cossairt et al. [24] uses a projector to project an artificial light field onto the real objects and the camera to capture the

light field from real objects. A lens array interface is used between the devices and real objects to properly align the light field. Some approaches focused on creating the perceptually correct AR result and they examined users' perception of different rendered phenomena in comparison to real scenes. Nakano et al. [41] studied the users' perception of shadows in AR and they developed a method for using a lower resolution light source map while still generating perceptually correct shadows. A good overview of illumination methods for AR and their categorization is given in [68].

Ray Tracing in Augmented Reality. Physically based algorithms developed in computer graphics often use ray tracing to accurately calculate light transport. However, the problem of high-quality ray tracing based rendering systems is limited performance. Offline rendering systems for adding virtual objects which simulate full global illumination were proposed in [27, 50, 77]. These methods achieve high rendering quality, however they run offline.

Ray tracing is an efficient algorithm for high-quality rendering. Its simplicity and robustness make it a preferable rendering method in many scenarios. Realistic rendering in AR by ray tracing was described by Scheer et al. [153]. Pomi et al. [137] demonstrated the use of ray tracing in a TV studio application where the video of a real character is inserted into the virtual world. This method uses a PC cluster to achieve interactive frame rates. Another method for occlusion calculation in AR environments by ray tracing was proposed by Santos et al. [29]. Former ray tracing based solutions for AR calculate only the specular global illumination or direct illumination. This thesis extends ray tracing based rendering for AR with innovative approaches for high-quality diffuse and specular global illumination.

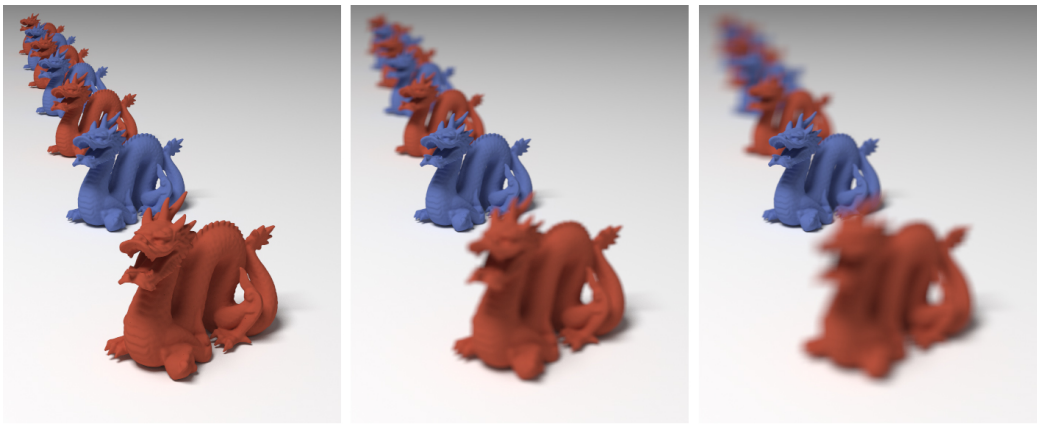

Figure 2.14: Comparison of DoF rendering with different aperture sizes. Left: The aperture size 0 corresponds to the pinhole camera model. Thus, all objects appear sharp. Middle: The objects out of focus are blurred depending on their distance from the focal plane. Right: A bigger aperture size causes a higher amount of blur in comparison to the middle image. The focus of the camera is on the second dragon in all images.

2.6 Depth of Field in Augmented Reality

Depth of field (DoF) is a natural feature of many optical systems such as the human eye or camera lenses. DoF can be defined as a range of distances near the focal plane, where the user perceives the image acceptably sharp [64]. To produce this effect in computer graphics, rendered objects have to be blurred according to their distance from the focal plane. The blurring can be simulated in two different ways.

First, in real-time rendering, DoF is calculated in image space by blurring the rendered image according to the depth values of virtual objects. However, this kind of DoF blur calculation is not physically correct and can introduce various visual artifacts.

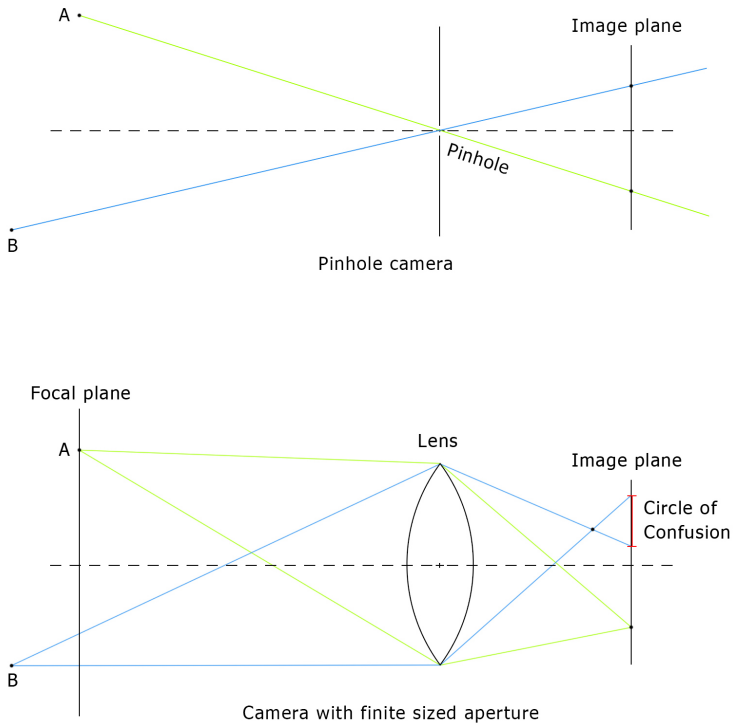

Figure 2.15: Comparison of the pinhole camera model (top) and a camera with finite sized aperture (bottom). Both points A and B are projected as sharp points to the image plane in case of a pinhole camera due to the infinitely small pinhole allowing only the differential ray to pass to the image plane from both points. In contrast to that, the lens with finite sized aperture projects only points lying in the focal plane (Point A) sharply as points onto the sensor. However, points out of the focal plane (Point B) are projected as areas onto the image plane. These areas are called circles of confusion. Thus, points out of focus appear blurred in the final image.

The second way of DoF simulation is based on camera optics having a finite sized aperture. Because the aperture has a finite area, a single point in the scene may be projected onto an area on the image plane, called circle of confusion. Correspondingly, a finite area of the scene may

be visible from a single point on the image plane, giving a blurred image (Figure 2.14). The size of the circle of confusion depends on the aperture size and distance between the object and the lens. The comparison of projections, using a simple pinhole camera model and a camera with finite sized aperture, can be seen in Figure 2.15.

Several methods were developed in the past to render the DoF effect in AR. However, techniques for physically based DoF in interactive AR have not been examined before. The majority of the proposed methods blur the rendered image according to blur parameters estimated from the camera. Park et al. [127] extended the efficient second-order minimization (ESM) method to handle blur in 3D object tracking. They render more intermediate images and combine them according to detected blur parameters. Their method is focused mainly on motion blur. Okumura et al. [122] estimated blur parameters according to a point spread function (PSF) detected on fiducial markers. They improved the tracking method to handle blurred markers and rendered the virtual image which was blurred according to the PSF. The disadvantage of their method is that virtual objects can be blurred only on the marker position and only a PSF of elliptical shape is used. A good overview of DoF algorithms in computer graphics can be found in [9].

2.7 Registration and Tracking in Augmented Reality

Camera tracking and registration are essential for spatial coherence in AR applications. The main problem in tracking is to estimate a precise camera pose in each frame. A camera pose consists of six degrees of freedom; three for position and three for orientation. The precise pose of the camera is required to set up the pose of a virtual camera and to render the virtual objects spatially aligned with real objects. By accurately tracking the camera pose, the virtual objects can be rendered to appear on a stable position in the real environment.

Several tracking technologies were developed in the past including mechanical, magnetic, optical, inertial, time-of-flight and hybrid tracking [15]. The most suitable tracking for AR applications is optical tracking, because it can calculate the accurate absolute camera pose just from image data. Moreover, if the same image is used for tracking and rendering, then no synchronization between tracking and display is required. The synchronization can be avoided, because both tracking and rendering use the same image data taken at the same time. Even if any movement of the objects appears in the time between the image capturing and display, the user only sees the image taken in the capturing phase with a little delay. But the virtual objects are aligned with the real video image correctly.

Based on the relation of tracking sensors (cameras) and tracked object, we can divide the optical tracking methods into two categories: outside-in and inside-out. In case of outside-in tracking, the sensors (cameras) are fixed to places in the environment and markers or tracking targets are fixed to the tracked object. Opposite to that, in inside-out tracking the sensors (cameras) are fixed to the tracked object and the markers are fixed to the surrounding environment.

The image-based tracking methods can be further subdivided into:

- **Marker-based tracking**. These methods use a printed marker as a visual target. The camera pose can be estimated from the projection of the marker to the image plane. A simple and efficient marker-based method for camera tracking was proposed by Kato and

Billinghurst [78]. The authors track the fiducial markers which are visible in the captured video. The pose of a marker is calculated from the orientation and position of marker's projection onto the image plane. The accuracy of this method can be improved by adding more markers to track. Non-square visual markers were also presented in the past. Cho et al. [19] used ring shaped fiducial markers and Vogt et al. [181] designed circular shaped marker clusters with various parameters. Markers can also be active or passive infrared targets.

- **Marker-less tracking**. Instead of tracking artificial markers, marker-less tracking uses the features of the real environment to estimate the pose of a camera. Natural feature tracking searches for features, such as points, lines, edges, or textures, in the scene and then tracks them. A camera pose calculation using natural features was presented in [21, 126, 179, 183]. A real-time approach for natural features tracking in AR running on mobile phones was presented by Wagner et al. [184]. Natural feature tracking can be also used for interaction in AR. For example the tracking of the human hand can be used to create a coordinate system for interaction with virtual objects in AR [101]. Detailed individual steps of a tracking pipeline for marker-less tracking are described in [62]. Feature descriptors, representing the tracked features in the real world, are the essential part of the natural feature tracking. Different types of feature descriptors were developed and used for tracking. A broad overview of feature descriptors and tracking methods can be found in [43]. Natural features are of high importance because they serve also as a basis for simultaneous localization and mapping approaches.

  Simultaneous localization and mapping (SLAM) methods were presented to create a map of the real environment simultaneously while tracking. These techniques simultaneously map the surrounding environment (e.g. the scene geometry) and track the pose of a camera in this environment. The parallel tracking and mapping approach [81] splits the tracking and 3D mapping into two separate tasks, processed in parallel. A 3D map of point features is built from previously observed video frames. By using this approach a vast amount of 3D feature points can be added to the map while still preserving interactivity. A robust real-time SLAM approach working in dynamic environments was presented in [174]. This method detects the changed features by projecting them from the keyframes to the current frame for appearance and structure comparison. An adaptive RANSAC algorithm is used to efficiently remove outliers from the set of features. Thus, a camera pose can be reliably estimated even in challenging situations. A good overview of SLAM methods can be found in [32].

2.8 Reconstruction of the Real Scene

An accurate geometric and radiometric model of a real scene is required for high quality rendering in AR. This model includes geometry, materials, and lighting of the real world and it is used to simulate the light interreflections between the objects of virtual and real worlds. Previous research in inverse rendering examined many approaches for reconstruction of geometry, materials, or light sources of the real scene [129]. Inverse problems can be ill-posed; there may be

no solutions or several solutions. They are also often numerically ill-conditioned, i.e. extremely sensitive to noisy input data [142]. Therefore the previous approaches posed constraints, at least one of the terms had to be known: Lighting, geometry, or materials. Moreover a special hardware was often needed to measure the scene properties. The previous research on real scene reconstruction can be divided into four main categories based on the term that is reconstructed: Geometry reconstruction, material estimation, light source estimation, and reconstruction of multiple terms.

2.8.1 Geometry Reconstruction

Geometry reconstruction is a widely known problem in computer graphics. In past research, many methods to reconstruct the 3D model of the real scene have been introduced. Specialized hardware was developed to scan the 3D geometry of the real world in a high sampling density to acquire an accurate result. A disadvantage of these methods is a high computational cost that cannot be handled in real time.

The problem of geometry reconstruction was approached in recent research by image-based methods. These methods take a set of input images of the scene and reconstruct the 3D model using multiview stereo [88, 159, 182]. Recently, real-time image-based geometry reconstruction methods have been presented. These methods are suitable for AR because they preserve interactivity while delivering the reconstructed model for each frame to perform occlusion calculation, physical interaction, rendering, and compositing.

A real-time 3D reconstruction method using a single camera is described in [119]. This method provides a dense 3D model reconstruction using every pixel of the camera image instead of sparse features. Moreover, a camera tracking is performed in parallel with the scene reconstruction. During reconstruction, the quality of the model is improved when more frames are captured. A non-convex optimization framework is used to optimize the reconstructed geometry. Recently, an approach for real-time 3D model reconstruction on mobile phones was proposed by Tanskanen et al. [175]. This method generates dense 3D models with absolute scale. Inertial sensors of mobile devices are used to make the tracking and mapping process more resilient to rapid motions and to estimate the metric scale of the captured scene. Moreover, an efficient scheme for dense stereo matching is presented, which allows interactive processing. Another method for real-time 3D reconstruction, utilizing a single camera, is MonoFusion [138]. This method firstly estimates the camera pose by sparse tracking. The estimated pose is then used for efficient dense stereo matching between the input frame and a previously extracted key frame to create depth map. Depth maps are directly fused into volumetric data structure and surfaces are extracted in each frame. MonoFusion is implemented on the GPU and it achieves real-time performance. Moreover, it is capable of recovering from tracking failures and it can filter out geometrically inconsistent noise.

Approaches which utilize a depth camera can reconstruct a precise 3D model of the real scene during tracking and mapping. Newcombe et al. [118] presented a method for accurate real-time mapping of indoor scenes using a moving depth camera. A dense volumetric model of the 3D scene is reconstructed in their approach. The pose of the camera is estimated by calculating the pose of the depth frame, relative to the global model, using a coarse-to-fine iterative closest point (ICP) algorithm [12]. In the presented method the reconstructed volume is limited by

the GPU memory. This problem was approached in [194] by converting the volumetric data out of the mapping region to triangular mesh representations. This improved algorithm works as follows. The volumetric representation, enriched by the truncated signed distance function (TSDF), is employed to map the particular region. As new regions of space enter the TSDF, previously mapped regions are extracted into a triangular mesh representation. A disadvantage of infrared depth cameras is their limitation to indoor environments. Another method based on a depth camera which reconstructs 3D geometry in real time was presented by Bylow et al. [16]. The authors used the signed distance function (SDF) to represent the geometry similarly to Newcombe et al. The tracking accuracy was improved in this paper by estimating the camera pose by direct error minimization of the depth images on the SDF.

In this thesis a predefined geometry of the real scene is used. However, any of the described geometry reconstruction methods can be employed in the future to extend the developed system.

2.8.2 Material Reconstruction

Material reconstruction estimates the radiometric properties of the objects in the scene. Many inverse rendering algorithms were developed in the past to approach the problem of material reconstruction. In order to reconstruct the material of the object with high accuracy, a special setup can be used which controls the angular sampling of light and camera positions [6, 46, 107, 110, 117, 198]. Several measurements are taken and a BRDF is estimated by optimizing the parameters to fit the sampled data. These methods provide accurate results of reconstructed material. However, disadvantages of them are the requirements of special hardware, dark environment, and long scanning time. Therefore, these methods can only be used in the preprocessing and not during a real-time AR session.

Approximative solutions which reconstruct the material with natural illumination were developed. Knecht et al. [84] proposed a method for real-time material estimation for augmented reality utilizing an RGB-D camera. In their approach, an additional camera with a fish-eye lens is used to capture the illumination of the real scene. Then, the highlights are removed from an input image and inverse diffuse shading is applied to calculate material parameters. Oxholm and Nishino [124] proposed a method for joint BRDF and geometry estimation with known but uncontrolled illumination. A method for reconstruction of shape and spatially-varying BRDFs from photometric stereo was proposed in [47]. The limitation of this approach is that the illumination must be known. Ramamoorthi and Hanrahan [142] proposed a signal-processing framework which describes the reflected light field as a convolution of the lighting and BRDF; and expresses it mathematically as a product of spherical harmonic coefficients of the lighting and BRDF. Inverse rendering can then be viewed as a deconvolution. Their approach can simultaneously reconstruct materials and lighting under the assumption of known geometry. A survey of problems and solutions in inverse rendering can be found in [129].

2.8.3 Light Source Estimation

AR has to use the simulation of real lighting in rendering to create the realistic result. Generally, three different approaches have been reported in literature to accommodate real lighting in AR rendering:

Image-Based Lighting. In the case of image-based lighting the whole environment image is used as a light source. This image is captured either by camera with fish-eye lens or by taking photo of a reflective sphere. Each pixel of the environment image can be used as directional light source. Image-based lighting [27,85] samples the environment map according to the probability density function (pdf) similar to the intensities of the pixels. This approach builds a cumulative distribution function from the environment image, allowing for random sampling from the probability distribution, which equals the intensity of the incoming light from different directions on the hemisphere [133].

Light Source Estimation from an Environment Map. An image-based light reconstruction from an environment image can be used to precisely specify the light positions and intensities. Frahm et al. [37] used an image processing approach to obtain information about light sources. The authors search for regions with high saturation in all channels and then apply a segmentation. They approximate the real light by point light sources and calculate their positions by processing two images from two fish-eye lens cameras. High-quality light reconstruction from a single-camera image was proposed in [77]. The authors used a user guided algorithm which processes the light sources, marked by the user, to find the optimal positions, shapes, and sizes of them.

Light Source Estimation from a Single Camera. The light sources can be estimated from the viewer's camera images. This approach analyzes the images from the main camera and finds features like shadows, reflections, or highlights to estimate the positions and intensities of light sources. The estimation of directional illumination distribution from shadows was presented by Sato et al. [152]. The authors analyzed image brightness inside shadows, cast by an object of known shape in the scene, and reconstructed the illumination on a series of patches on the sphere. Estimation of multiple directional light sources for AR from shadows and shading, using a single image, was described in [190]. This approach assumes a known geometry and Lambertian reflectance of real objects. A contour-based approach for directional light source reconstruction was presented by Nillius and Eklundh [120]. The requirement of their approach is that there exists a segment of an occluding contour of an object with locally Lambertian surface reflectance in the image. Shim [161] estimated light sources separately for high-frequency and low-frequency lighting utilizing spherical harmonics and point light sources. The spatial reconstruction of the positions of light sources was presented by Hara et al. [60]. Their technique requires a single image and locations of specular reflections as an input. Additionally, they use the series of multiple polarization images to remove the constraints on the diffuse reflectance property. This method simultaneously recovers the reflectance properties and the light source positions by optimizing the linearity of a log-transformed Torrance-Sparrow model. A light source estimation with an RGB-D camera which assumes Lambertian surfaces was proposed by Gruber et al. [54]. The authors used the estimated environment map to render virtual objects into AR with coherent illumination. Another approach based on an RGB-D camera which captures high dynamic range (HDR) light fields for AR was presented in [113]. In this approach, the HDR environment maps can be acquired directly from a dynamic set of images in real time. A moving RGB-D camera is used as a light field sensor, based on a dense 3D tracking and mapping, that avoids the need for a light probe. The HDR radiance map is then computed from a stream of low dynamic range

(LDR) images. This HDR radiance data can be used to create an arbitrary number of virtual omni-directional light probes that will be placed at the positions of virtual objects in AR for rendering.

2.9 Presence in Augmented Reality

The sense of presence has been a topic of research in VR and AR for many years. Presence can be defined as *"a psychological state in which virtual objects are experienced as actual objects in either sensory or nonsensory ways"* [100]. Presence in a virtual environment is a sense of *"being there"*. However, in augmented reality presence is a sense of *"it is here"* referring to the presence of virtual objects in the real environment [106, 144].

Several experiments have been conducted to study the effect of different parameters on presence. The influence of rendering quality on presence was studied in [108, 200], but the results could not confirm that presence depends on rendering quality. Rendering quality in VR was also studied in an environment that displays a precipice, a pit that the participant looks over [163]. The results of the experiment suggested that the participants observing the VR with ray tracing rendering reported a higher level of presence than participants observing the VR with ray casting rendering. Ray casting uses only the primary rays from the camera in comparison to ray tracing, where the reflected light rays are traced as well. An overview of research on presence in VR can be found in [157].

Presence in AR was investigated by Sugano et al. [169]. They conducted an experiment, which examined the effect of a shadow representation of virtual objects in AR. The results showed that a shadow representation is important for increasing the presence of virtual objects. An experimental framework for studying perceptual issues in photorealistic AR was presented by Knecht et al. [83]. The authors performed a study to investigate the influence of different shadows and lighting calculation methods on user performance in five different tasks. Their results indicate that there are no significant effects of the studied rendering conditions on task performance. The questionnaires, measuring the presence in AR, were developed by Regenbrecht et al. [143, 144]. The authors describe the development of a presence questionnaire, a data collection strategy and the first results of the application of the questionnaire.

CHAPTER 3

Ray Tracing in Augmented Reality

Ray tracing is the preferred algorithm in high-quality rendering. The main goal of this thesis is to achieve a high quality of the composited image while preserving interactivity. Thus, we developed an AR rendering system based on real-time GPU ray tracing. This chapter describes the design of our core rendering and one-pass compositing system together with the developed algorithms for physically based camera simulation, light source estimation, and antialiasing in AR.

3.1 System Overview

The developed rendering system combines real-time ray tracing with the differential rendering compositing algorithm. The input data to the system include the following sources:

- **Offline input data:** The system needs reconstructed geometry and materials of the real world to properly simulate the light interaction between real and virtual objects. Moreover, the relations of specific AR markers to the geometric objects and the camera have to be defined. A requirement is to use at least one marker for camera tracking.

- **Real-time input data:** The system requires one live video stream from camera at the position of the viewer. This stream is augmented with the virtual objects in real time. Additionally, a second live video stream can be used to capture the illumination of the real world. A camera with a fish-eye lens is used for this purpose.

The running system takes the input data in each frame and calculates the output image including virtual and real objects together with proper light interaction between them. An important property of the developed AR system is that all algorithms run in real time to preserve the interactivity of the AR application. The rendering and compositing pipeline consists of the following subparts (Figure 3.1):

- **Camera Tracking** is the essential part for achieving spatial coherence in AR. The tracking system takes an input frame and calculates a camera pose from the position of the marker in the image. The calculated camera pose is later used for rendering to set up the virtual camera at the correct position and orientation. Stable and accurate tracking can guarantee correct spatial relations between virtual objects and the real scene.

- **Light source estimation** is needed to correctly illuminate the scene with virtual lights, similar to the real ones. If the lights are accurately estimated, the illumination of virtual objects and shadows are consistent with reality.

- **Rendering and one-pass compositing** is the most important part of the developed system. The core rendering algorithms, using GPU ray tracing and one-pass compositing, are described in Section 3.3. Additionally, advanced rendering techniques are implemented to calculate realistic visual effects. In this thesis we add the following additional effects: Physically based camera simulation (Section 3.4), antialiasing (Section 3.5), and global illumination calculation (Chapter 4).

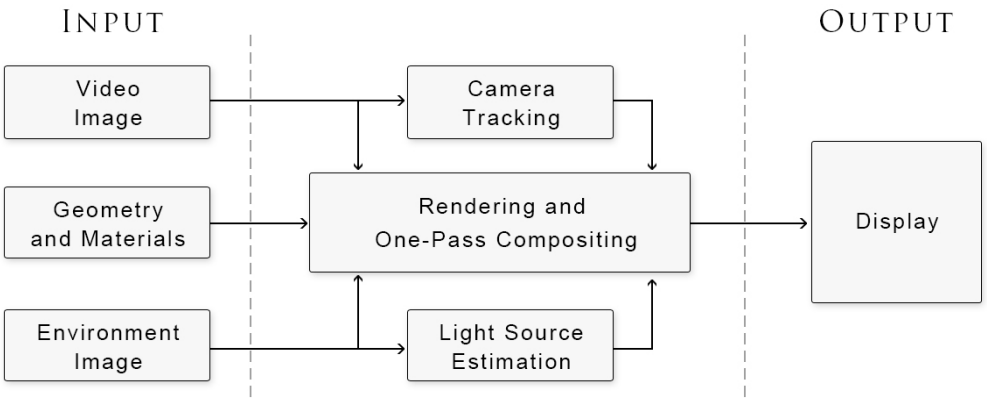

Figure 3.1: Overview of the system design.

In every frame, an image is taken by the video camera and the environment camera. The captured images are used to calculate the camera pose in tracking and the light sources in light source estimation. The rendering and compositing part is the main part of the system. Data from tracking and light source estimation are sent to the rendering system to set up the virtual camera and lighting. The images from both real cameras are sent to the rendering as well to composite the final image and to calculate the reflections or refractions. The rendering system calculates the mixed radiance I_m and the real radiance I_r together and composits the final image. This image is then displayed on the screen. Additionally, the system can be extended by any real-time 3D reconstruction technique (Section 2.8.1) to avoid the need of using predefined geometry.

Ray tracing based rendering in AR has many advantages. One of them is the natural simulation of reflections. For example, the virtual objects can be reflected in a real mirror as we can see in Figure 3.2. The next advantage is the possibility to naturally simulate the depth of

field effect (DoF) in a physically correct way. This thesis presents an approach for a physically based DoF simulation in AR in Section 3.4. Moreover, antialiasing can be natively calculated by supersampling the pixel area by multiple rays. Additionally, advanced high-quality GI methods based on ray tracing can be applied in AR (Chapter 4).

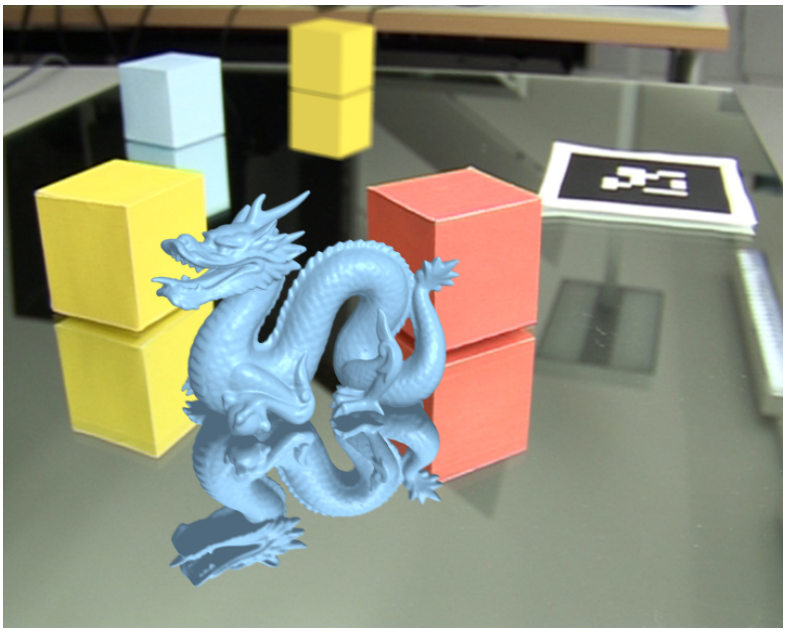

Figure 3.2: Ray tracing in AR naturally provides features like specular reflection on mirror. Moreover, the physically based DoF technique can improve visual realism of the produced image. The scene contains a real mirror, real front yellow, blue, and red cubes. The dragon and the back yellow cube are virtual. Note the correct reflection of virtual objects in the real mirror.

3.2 Light Source Estimation

Light source estimation is employed in the developed system to calculate the positions of light sources in the real scene. This information is later used in the rendering step to render the virtual objects with coherent illumination. The light source estimation is running in a separate thread to not slow down the rendering. Two lighting methods are included in our system: Light source estimation by processing of the environment map and image-based lighting. An additional camera with a fish-eye lens is used to capture distant light in real time (Figure 3.3).

3.2.1 Light Source Estimation from an Environment Camera

An image processing approach is employed in our system to estimate the positions of light sources from the environment image. This image is captured by a fish-eye lens camera every frame and the following procedure is applied to calculate the light positions. Firstly, a threshold-

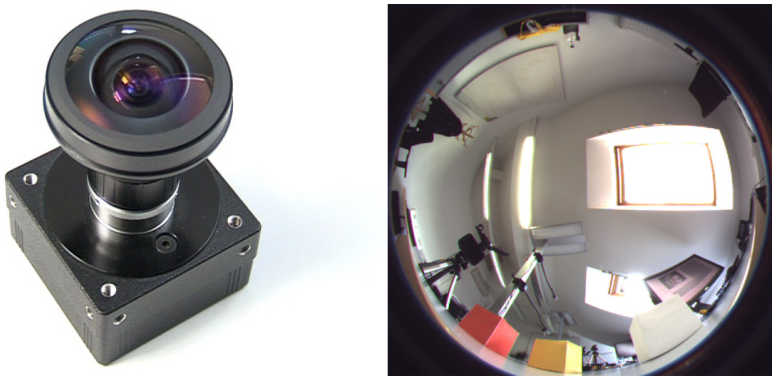

Figure 3.3: Left: Camera with a fish-eye lens used for real-time environment map capturing. Right: The image captured by this camera.

ing is applied to highlight the areas with high radiance values. In the next step, a blob detection is used on the binary image to detect the biggest sources of high incoming radiance. Connected component analysis, provided by OpenCV[1], is utilized and a contour tracing approach [48] is used to find the contours of radiance blobs. The area of every blob is calculated and the biggest blobs are selected as light sources (Figure 3.4). The direction of incoming light is estimated according to the blob center position in the environment image by reprojecting the blob center onto the hemisphere. The position of a light source is then estimated by using the user supplied average room size constant. The light is positioned at a point in the reconstructed light direction at a distance of room size from the origin. The origin of the coordinate system is at the marker position. Currently, only the position of the biggest light source is estimated. A future extension of this method will allow an arbitrary number of light sources to be extracted and bigger area light sources to be sampled by more point light sources. The environment image capturing and light source estimation run asynchronously in a separate thread. This makes the rendering speed almost independent of the light source estimation and the last calculated values are always used.

3.2.2 Image-Based Lighting

The whole environment image is used as a light source in the case of image-based lighting. We capture the environment image in real time by the fish-eye lens camera. In the path tracing extension (Section 4.4) of the developed rendering system an image-based lighting approach is available to render virtual objects with natural illumination. In this approach, a whole environment map, captured by a fish-eye lens camera, is used as a source of light. Firstly, an inverse tone mapping is applied to the environment image in every frame to obtain HDR radiance values. This radiance is later accessed in the ray tracing pipeline. If a reflected ray does not intersect with any geometry, the environment map is accessed to calculate the real light coming from a particular direction of that ray. This method simulates natural lighting but requires more samples to calculate the converged solution.

[1] www.opencv.org

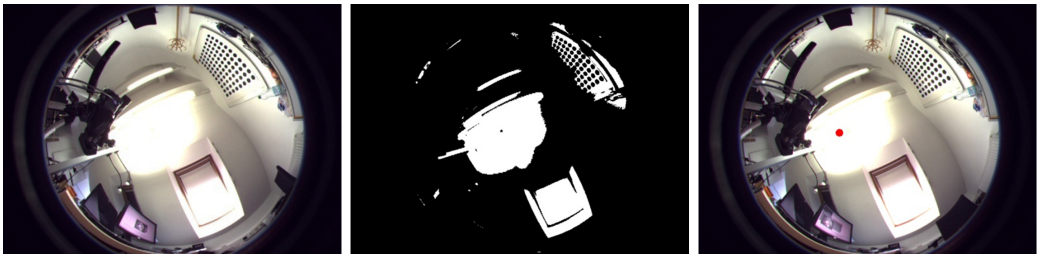

Figure 3.4: Left: An environment image captured by the fish-eye lens camera. Middle: Binary environment image after thresholding. Blobs are detected in this image by a contour tracing approach. The biggest blob is selected as a light source. Right: The estimated position of the biggest light source in the environment map is marked by the red dot. This position is calculated as a center of the biggest blob.

3.3 One-Pass Differential Rendering

Compositing is an essential part of an AR system because it inserts virtual objects into the real image. The differential rendering algorithm introduces the light interreflections between real and virtual objects. The problem of this algorithm is that it requires two rendering solutions to calculate the final composited image. To overcome this problem, a novel one-pass differential rendering algorithm in ray tracing is presented in this section which significantly improves the performance by calculating both mixed and real radiance together in one ray tracing pass. The calculated radiances are stored in a per-ray data structure. This algorithm is implemented directly in a ray tracing pipeline.

In order to calculate both mixed radiance I_m and real radiance I_r together, we define more complex ray types than in traditional ray tracing. In contrast to traditional ray tracing, using only the radiance ray type and the shadow ray type, we divide rays in our engine into four different types:

- **Mixed radiance ray** - this ray type is used to evaluate both mixed radiance and real radiance. It can intersect both virtual and real objects.
- **Real radiance ray** - it can evaluate the real radiance only. Thus, it overpasses the virtual objects and it can intersect only with the real ones.
- **Mixed shadow ray** - it is used to calculate the visibility between the light source and the current point for a mixed scene. Both virtual and real objects are taken into consideration for intersection calculation.
- **Real shadow ray** - the visibility between the light source and the current point, for a real scene only, is evaluated by this ray type. Therefore, it can intersect only with real geometry.

The ray tracing pipeline starts by shooting the mixed radiance primary rays from the camera position. Then, the radiance rays can possibly change their type after they hit a virtual object.

Shadow rays are only used to evaluate the visibility between the scene point and light source. They cannot change to other ray types. The per-ray data (PRD) structure of mixed radiance and real radiance rays contains four variables: **mixed radiance**, **real radiance**, **mask**, and **path depth**. We use four different rules for ray type changing and shooting depending on ray type and object type (Figure 3.5):

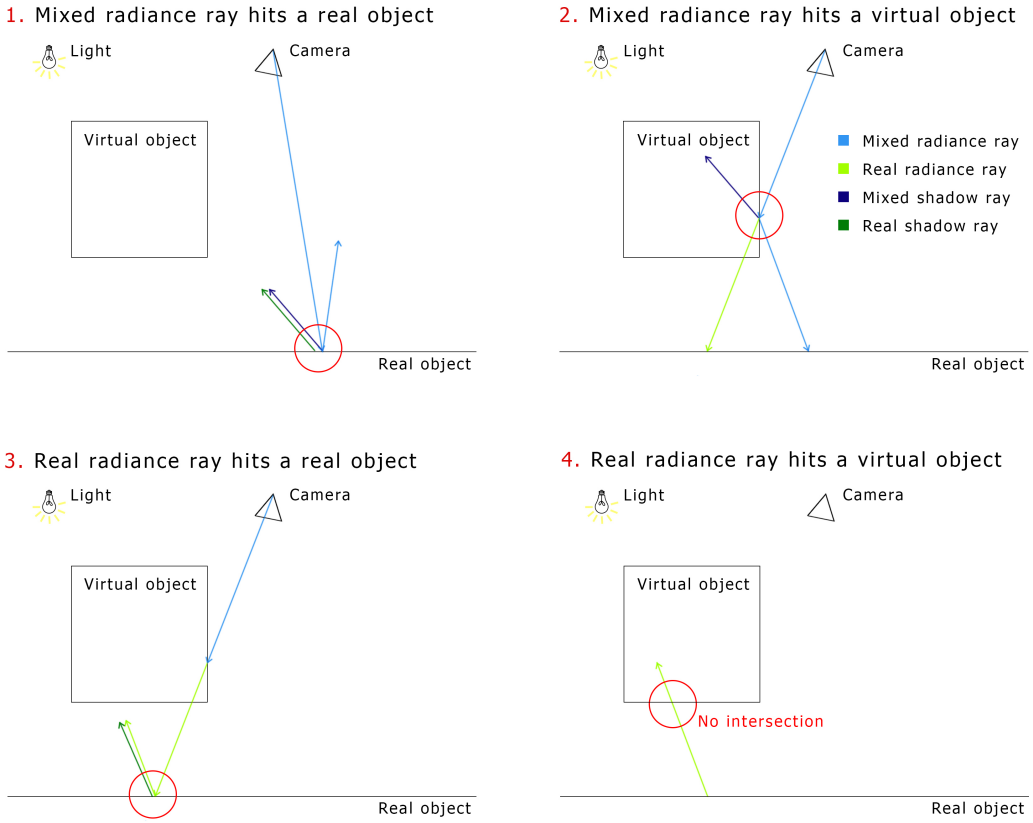

Figure 3.5: Four different cases of handling the ray-scene intersection depending on the ray type and the geometry type. The ray type can change after hitting virtual objects as we can see in case 2 (top right). The color coding of the ray types is depicted on top right.

1. **Real object is hit by a mixed radiance ray** - Real and mixed shadow rays are shot to calculate the visibility from the light sources for real and mixed scenes separately. A mixed reflected ray is shot to evaluate the reflected light coming from the sampled direction. In the case of a perfect specular surface, a mirrored reflected direction is used to shoot the mixed reflected ray. In case of path tracing (Section 4.4), a random ray direction is sampled on the hemisphere.

2. **Virtual object is hit by a mixed radiance ray** - A real radiance ray, continuing the primary direction, is shot. This ray is used to evaluate the real radiance without taking the

virtual object into account. Moreover, a mixed radiance reflected ray is shot to evaluate the mixed reflected radiance. In addition, a mixed shadow ray is shot. No real shadow ray is shot because the point on the virtual object does not contribute to the real radiance.

3. **Real object is hit by a real radiance ray** - A real shadow ray and a real radiance reflected ray are shot. Real radiance rays evaluate the real radiance only. Therefore, no mixed rays are shot.

4. **Virtual object is hit by a real radiance ray** - This case cannot happen because real rays do not intersect with virtual objects. Thus, the ray-triangle intersection routine (Section 3.7.3) prevents this case.

The corresponding shading results for each ray are stored as appropriate variables in a per-ray data (PRD) structure. The mask value in PRD is set to 1.0 when a ray hits a virtual object and to 0.0 otherwise. This value is used in the differential rendering equation (Equation 2.20) to composite the virtual objects with the real image. The compositing is performed directly on the GPU at the end of the ray generation program (Section 3.7.3). Our method is more efficient than two-pass differential rendering because all types of rays do not have to be shot in all cases.

3.4 Physically Based Depth of Field in Augmented Reality

A correct camera simulation in AR is of high importance for visual coherence between virtual and real objects. If a simple pinhole camera model is used, the real objects, which are out of focus, are blurred while the virtual ones are always sharp. This poses an inconsistency on the visual appearance of an AR scene. Thus, a correct camera model has to be used to produce a consistent depth of field (DoF) effect. Previous methods for DoF simulation in AR (Section 2.6) are not physically correct and they can simulate the DoF only on the marker position. In this section a physically based DoF method for AR is presented which uses a camera model with finite sized aperture.

A lens with finite sized aperture is the natural type of optics for many lens systems. In order to follow the finite sized aperture model and to reproduce a physically correct DoF effect, the visibility of objects has to be calculated from many points on the aperture. Our approach is similar to the one described in [133] and it is developed especially for AR.

Many rays have to be sampled on different aperture points to obtain the final blurry effect of out of focus objects. Stratified jittered sampling [114] is used in our method to sample both the 2D domain of aperture and the 2D domain of the pixel area. The sampling algorithm performs the following steps. At the beginning of ray sampling, the algorithm starts at the fixed center of projection O. First, the pixel area at the image location is sampled to create a ray direction $\vec{d}$. Then, the intersection point I of a ray with direction $\vec{d}$ and the focal plane is calculated by the following equation:

$$I = O + \vec{d} * d_f \tag{3.1}$$

where d_f is the distance from the center of projection O to the focal plane. The vector $\vec{d}$ is not normalized to make Equation 3.1 valid. The new ray direction $\vec{d'}$ is calculated by subtracting

the sampled aperture point O' from the intersection point I. The aperture point is randomly sampled in the planar area around origin O. The size of the sampled area is defined by the aperture size.

$$O' = O + r_x \cdot \vec{X} + r_y \cdot \vec{Y} \tag{3.2}$$

$$\vec{d'} = I - O' \tag{3.3}$$

The r_x and r_y in equation 3.2 are two randomly sampled numbers from domain (-1,1) scaled according to the aperture size. Vectors $\vec{X}$ and $\vec{Y}$ are normalized coordinate axes of the image plane. The sampling point of the image plane remains the same because the image plane has to shift according to the origin's offset. If a sufficient number of aperture samples is used, a high quality DoF result is calculated. The aperture sampling is depiced in Figure 3.6.

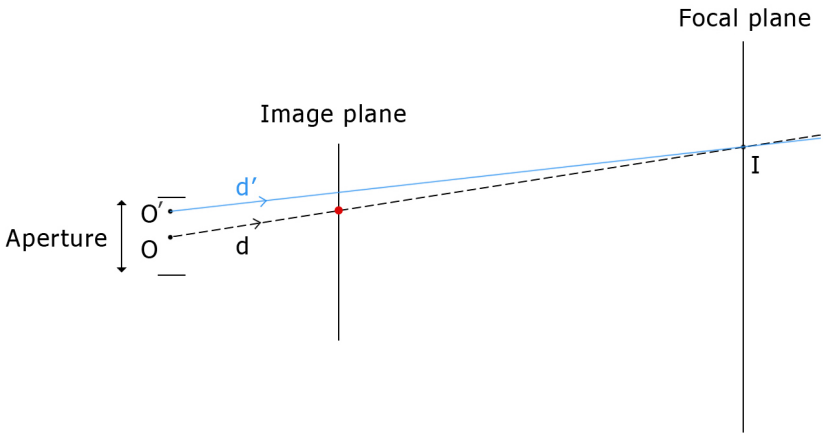

Figure 3.6: The sampling of the finite sized aperture is done by creating a new origin O' and calculating a new ray direction d'. The original pixel, marked by the red dot, is used to write the final radiance value.

Our one-pass differential rendering is used to composite the virtual objects, rendered with a physically correct camera model, to the real scene. Correct blending between the blurred virtual and real objects is ensured by averaging the mask values from our ray tracing pipeline. If some ray samples per pixel belong to the virtual objects and some belong to the real ones the average mask will have the value between 0.0 and 1.0. This real number defines the amount of blending between the virtual and real objects used in the differential rendering equation.

The result of the developed physically based DoF technique and a comparison to a rendering with the pinhole camera model can be seen in Figure 3.7. The virtual cubes, rendered with the pinhole camera model, are visually inconsistent with the real objects which are out of focus. Using the physically correct camera model increases the visual coherence by producing similar blur to the real camera.

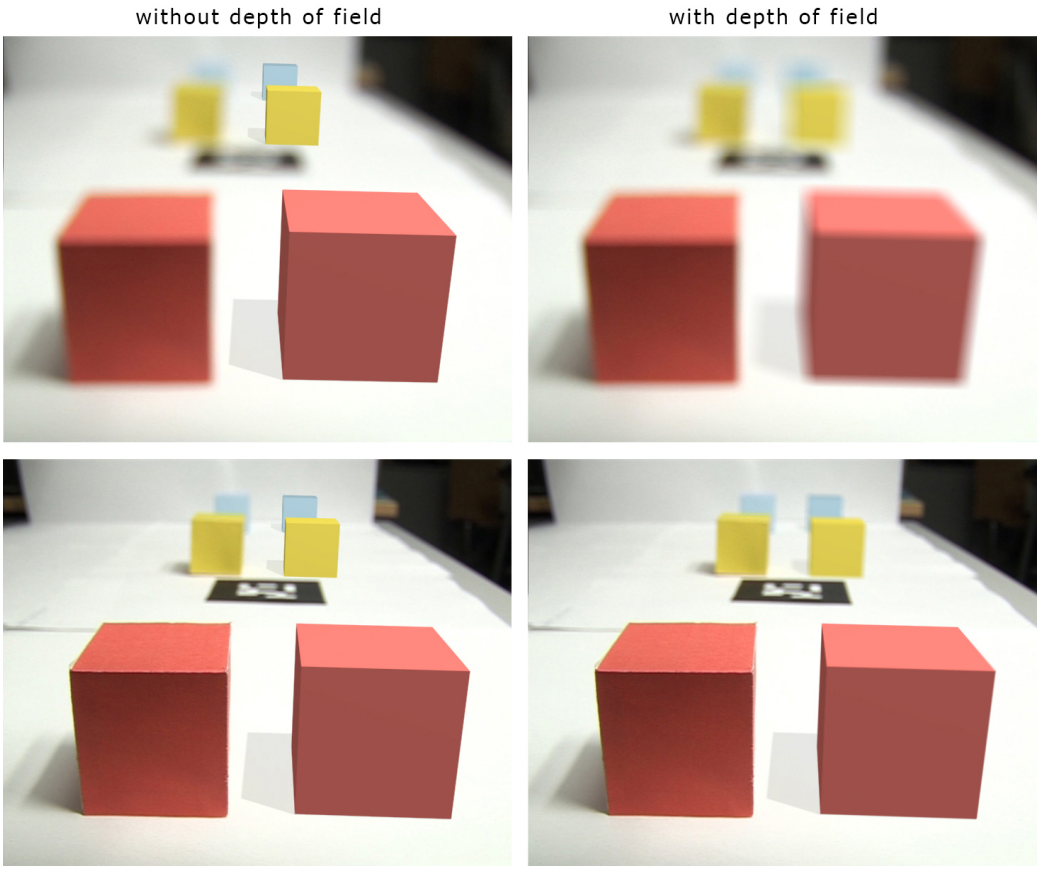

Figure 3.7: Results of our DoF technique. Left column: rendering without DoF. Right column: rendering with DoF. First row shows defocused image and second row shows image with camera focused at closest cube. Left cubes in each image are real and right cubes are virtual.

3.5 Antialiasing

As we show in our evaluation in Chapter 5, the antialiasing of rendered objects is an important feature for visual realism in AR. It reduces artifacts, caused by insufficient sampling density in high-frequency parts of the image function, such as discontinuities on the edges of virtual geometry (Figure 3.8). Aliasing artifacts can reveal to users that objects are virtual and therefore decrease the overall realism of the composited video. Distributed ray tracing [22] offers a very elegant and natural method for antialiasing by supersampling the pixel area. It is a robust method since various random variables can be distributed over multiple rays shot per pixel. For example, supersampling can be used to sample the 2D domain of the pixel area together with the 2D aperture to get the DoF effect. We use stratified jittered sampling [114] to achieve a good distribution of samples. A disadvantage of supersampling is that more samples require higher computational time. Nevertheless, a tradeoff between quality and speed can be achieved.

In order to reduce aliasing, we increase the number of rays shot through each pixel. The

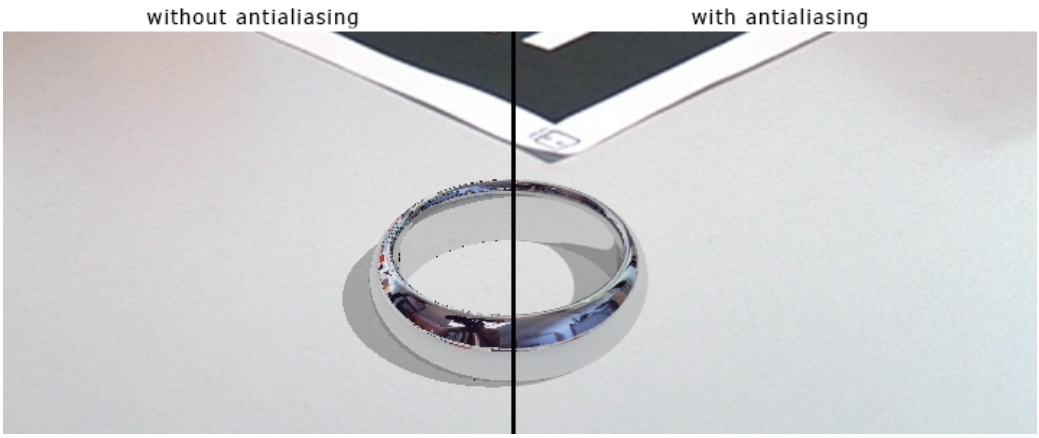

Figure 3.8: Comparison of rendering without (left) and with (right) antialiasing. Note the quality difference of reflections on the virtual ring between the two images.

final pixel color is then calculated by filtering the calculated radiance values which are obtained by shooting the rays. The reduction of aliasing on edges of virtual objects can be achieved by filtering the mask value in the differential rendering equation as well. Moreover, this blending strategy can also be used with the DoF calculation, where blurred edges of out-of-focus objects should be blended with the real background. The comparison of DoF quality with different sampling rates can be seen in Figure 5.1.

3.6 HDR Rendering

HDR rendering is important to simulate the wide range of light intensities in the real scene. Virtual objects should be rendered by using similar radiance values as in the real world. Our system calculates the rendering and compositing result in HDR. The light transport calculation handles all radiance values as float numbers. In order to display this HDR result on standard display devices, a tone mapping operator is applied to the final radiance values to convert them from HDR to LDR. We use an inverse tone mapping operator to convert LDR input images from cameras into HDR radiances [8] to be used in rendering. These HDR radiances are accessed directly in the ray tracing pipeline as the source of incoming light captured by the cameras.

3.7 Implementation

This section describes the implementation of the core AR system developed in this PhD thesis. The main part of the system is the GPU rendering and compositing. Implementation details of this part are discussed in Section 3.7.3. Additionally, the details about the used hardware configuration, included libraries, and virtual content creation are shown.

3.7.1 Hardware

A commodity PC with a GPU supporting NVIDIA CUDA[2] can be used to run our system. In our experiments, we use a PC with hexa-core CPU. Three cores are utilized by our system. Details about cores utilization by different subsystems are provided in Section 3.7.2.

Our rendering and compositing system was tested on two commodity graphics cards: The dual-core NVIDIA GeForce GTX 590 and the the dual-core NVIDIA GeForce GTX 690. Any video camera connected to the PC can be used to capture the input image for compositing in real time. We use the Sony HVR-Z1E camcorder. This camera provides an image resolution of 720x576. Thus, this resolution was used also for rendering and compositing in the majority of our tests. For the experiments with DoF a low aperture value (equals to a bigger aperture size) is used to get a stronger DoF effect. In order to capture the environment image in real time we use the Basler A312fc camera with a fish-eye lens. The lens has a wide 185 degrees field of view. Thus, it is possible to capture the whole upper hemisphere of incoming light. Both cameras are connected to the PC by the firewire interface to achieve high data transfer rates.

3.7.2 System Implementation

Our system is implemented as a multithreaded application allowing for asynchronous data processing. The following three threads are running in the system. The first one and the most important is the rendering thread which invokes rendering on the GPU in each frame. This thread also handles the input events including keyboard and mouse input. The second thread is used for camera capturing and camera pose tracking. Each frame, captured by the camera, is used to estimate the camera pose. The last captured image and tracking data are used for rendering. The same input image is always used for rendering and tracking. This allows the system to avoid the synchronization issues between rendering and tracking. The third thread captures the environment image by the fish-eye lens camera. Additionally, the light source estimation is performed in this thread.

The OpenCV[3] library is used to process the environment image in real time and to estimate the positions of light sources as described in Section 3.2.1. This library is also used to capture images from both cameras. At the beginning of each frame, the captured images from both cameras are transferred to the GPU to make them available for the ray tracing pipeline. For camera tracking within the scene, the marker based library ARToolKitPlus [185] is used. The printed marker is located in the real scene and the pose can be estimated. Then, the same pose is set for the virtual camera to align virtual objects with the real world. The orientation of the marker determines the rotation of the world coordinate space. The environment image retrieved by the fish-eye camera is orientation-dependent, therefore the fish-eye camera has to be aligned with the marker. Proper alignment ensures correct shadows, reflections, and refractions. In our experiments, we aligned the printed marker with the fish-eye camera sensor manually by rotating the fish-eye camera (xy plane of the camera sensor) to fit the rotation of the paper marker (xy plane in the marker coordinates).

[2] www.nvidia.com
[3] www.opencv.org

The correct setup of virtual camera parameters is required to achieve the coherent appearance of virtual and real objects. The parameters of the virtual camera have to be same as the parameters of the real one. The aspect ratio and field of view are calculated by the tracking system. The focal distance and the aperture size are set manually to be the same on the real and virtual camera. Therefore, autofocus is disabled on the real camera. It can be enabled in the future by transferring data about focus distance and aperture size to the PC in real time.

3.7.3 Ray Tracing

Ray tracing is highly suitable for parallel execution. Therefore, all rendering computations in our system are performed on modern massive parallel GPUs. We use the OptiX [128] GPU-based ray tracing engine to implement tracing of mixed and real rays as well as to composite virtual objects with the real scene. Stratified jittered sampling, used in our rendering, needs a uniform random number generator. We use a pregenerated array of random numbers in our solution. This array is generated on the CPU at the initialization and it is reused in every frame. Reusing the same random parameters for every frame has the advantage of temporal coherence. If new random numbers were generated every frame, the noise caused by an insufficient sampling rate would tend to change in time causing temporal artifacts. These temporally changing artifacts are easily observable by the user. Therefore, the predefined random numbers are of high advantage to avoid this problem. We use the BVH acceleration data structure to accelerate the ray-geometry intersection calculation. This structure is built automatically by OptiX in real time on the GPU. Both camera image and environment image are transferred to the GPU in real time as textures. When the compositing result is calculated, it is displayed by OpenGL[4].

Our rendering and compositing pipeline is implemented in OptiX programs which are executed on the GPU. The correspondence between OptiX programs and ray tracing pipeline can be seen in Figure 3.9. The algorithms in our system are implemented in the following programs:

- **Ray generation program** - This program is invoked at the beginning of the ray tracing pipeline. We implemented the physically based camera model, stratified jittered sampling strategy, and compositing in this program. Firstly, the area of the pixel and the area of the aperture are sampled. Then the mixed radiance ray is shot from the sampled origin towards the sampled direction to calculate both real and mixed radiances for each sample. When the ray tracing calculation finishes, the program uses the differential rendering equation to composite the ray tracing results with the camera image

- **Intersection program** - The intersection between ray and triangle is calculated in this program. In our implementation, special attention is paid to the ray types and geometry types. If the real radiance ray hits a virtual object, no intersection is reported according to the rules described in Section 3.3.

- **Closest hit program** - This program implements the shading of the object. It can optionally shoot reflected rays and it shoots shadow rays to test the visibility of a current point

[4] www.opengl.org

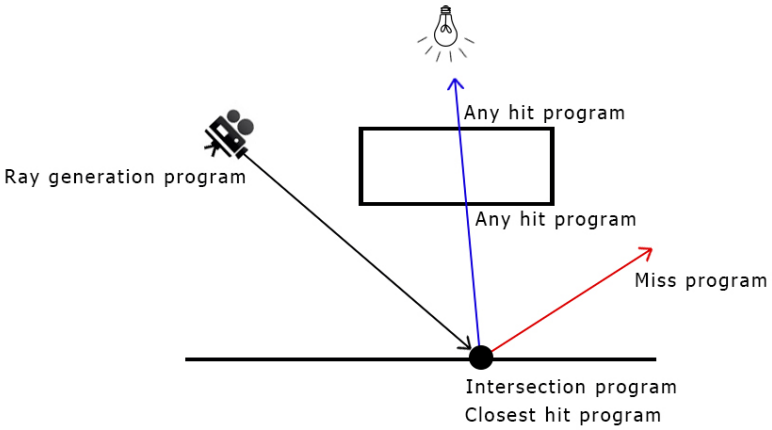

Figure 3.9: Invocation of different OptiX programs during ray tracing. The pipeline starts with the ray generation program. In this program, mixed radiance rays are shot towards the scene. If a ray enters the leaf of the acceleration data structure, the intersection program is invoked on primitives in this leaf. If the closest intersection is found, the closest hit program is invoked. The reflected rays and shadow rays can be generated in the closest hit program. If a shadow ray hits a geometry, the any hit program is invoked. Finally, if the ray misses any geometry, the miss program is executed. The whole pipeline is running on the GPU.

from the light source. The material behavior and proper light reflection is implemented in this program. The four rules for ray type shooting and changing (Section 3.3), essential for our one-pass differential rendering, are implemented here. Additionally, this program implements the reprojection technique described in Section 4.1.1.

- **Any hit program** - It is invoked if a shadow ray hits any geometry. In this case the light contribution from the particular direction is set to zero because the light source is occluded. If the real shadow ray is shot, the occlusion is calculated with real objects only.

- **Miss program** - If a ray does not hit any geometry, the miss program is executed. If the last hit surface was a specular one, the miss program accesses the environment texture to provide data for reflection calculation. Additionally, the environment texture is accessed if image-based lighting is used.

3.7.4 Scene Representation

Our system uses the collada [3] file format to load the geometry and materials of real and virtual objects. Both geometry and materials are transferred to the GPU before rendering. The geometry and materials of the scene are created manually in Blender[5]. Some models, used in our evaluations, come from public repositories. The collada file format is used to store real and virtual objects together in one file. The real objects are marked by a flag indicating that these

[5] www.blender.org

objects should be handled as real ones in differential rendering. This flag is set in Blender and then correctly interpreted in our ray tracing pipeline. In the future, the system can be extended by automatic geometry reconstruction of the real scene to avoid the need of using predefined real objects. Dynamic geometry, lighting, and materials are supported by our system as OptiX can rebuild the acceleration data structure in real time.

CHAPTER 4

Physically Based Global Illumination in Augmented Reality

Global illumination provides important visual features which increase the visual realism of AR. Examples of such features are reflections, refractions, caustics, or diffuse indirect light. This chapter focuses on the calculation of global illumination in a physically based manner to provide a high-quality result. Novel algorithms for specular light transport in AR (Section 4.1) and diffuse light transport in AR (Section 4.2) are introduced in this chapter. Furthermore, a differential progressive path tracing algorithm is presented for previsualization and relighting in AR (Section 4.4). This algorithm can calculate the unbiased result of global illumination in AR in a progressive manner.

4.1 Specular Light Transport in Augmented Reality

This section presents novel algorithms for high-quality light transport in AR. The focus is especially given to the light interaction with specular surfaces causing reflections, refractions, and caustics. Accurate reflective/refractive material simulation is presented together with proper refraction of the real world in virtual objects. The core system, described in Chapter 3, is extended by interactive GPU photon mapping and by additional algorithms. Interactive photon mapping on the GPU is used for caustics rendering in AR and it introduces global light transport between virtual and real objects. An example of high quality specular light transport is the rendering of realistic glass material (Figure 4.1). It is simulated in our ray tracing system by Fresnel reflection [133] and proper refraction. To overcome the problem of acquiring incoming radiance from the real world which is refracted in glass objects, we use a camera reprojection method similar to [50]. This method can find the correct radiance in the camera image. Furthermore, the real environment map is used to render proper reflection/refraction of the real world in specular virtual objects. In our implementation, we exploit the parallel nature of photon mapping and utilize the massive parallel power of modern GPUs. The advantage of the presented methods over

previous work is that we can naturally render specular surfaces, like glass or mirrors, in high quality. Moreover, a method for interactive caustics calculation in AR is presented. With this method, caustics can be created by reflecting or refracting light on both real and virtual specular objects.

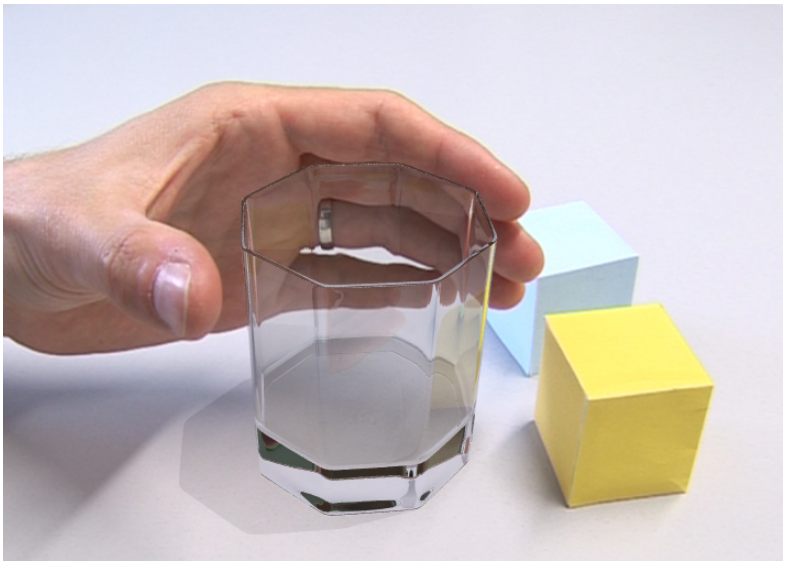

Figure 4.1: Refractive virtual glass surrounded by the real environment. The correct refraction of the hand in the glass is obtained by the reprojection method.

4.1.1 Specular Refraction and Reflection

Fresnel Reflection. Ray tracing is capable of properly simulating reflection and refraction on specular surfaces. The following rules are used in the developed system to trace the reflected and refracted rays. If a ray hits a reflective surface, the reflected ray direction is calculated and a new ray is shot. If a ray hits a refractive surface, both reflected and refracted rays are shot in order to properly simulate the refraction and reflection on transparent materials described by Fresnel equations [133]. We use Schlick's approximation of the Fresnel term described by the following equation [156, 172]:

$$F(\theta) = F_\perp + (1 - F_\perp)(1 - \cos\theta)^5 \tag{4.1}$$

where θ is the angle between the incident ray direction $\vec{\omega}$ and the surface normal $\vec{n}$. $F_\perp$ is the Fresnel term at the perpendicular direction. The Fresnel term, calculated by Equation 4.1, is used to interpolate between the refracted and the reflected light. The refracted ray direction $\vec{\omega}'$ is calculated according to Snell's law [103, 133] by the following equation:

$$\vec{\omega}' = \left(\frac{\eta_1}{\eta_2} \vec{n} \cdot \vec{\omega} - \sqrt{1 - \left(\frac{\eta_1}{\eta_2}\right)^2 (1 - (\vec{n} \cdot \vec{\omega})^2)} \right) \vec{n} - \frac{\eta_1}{\eta_2} \vec{\omega} \tag{4.2}$$

This equation is derived from the original equation described by Snell's law:

$$\eta_1 \sin\theta = \eta_2 \sin\theta' \tag{4.3}$$

η_1 is the index of refraction of the material that the light is leaving and η_2 is the index of refraction of the material that the light is entering. The refraction of light is depicted in Figure 4.2.

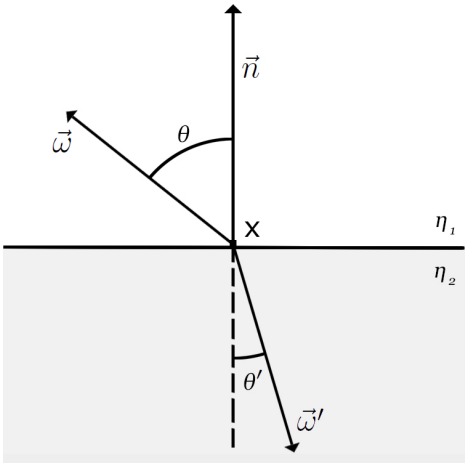

Figure 4.2: The angle of incidence θ and the angle of transmission θ' are related by Snell's law given in Equation 4.3 .The refracted ray $\vec{\omega}'$ is calculated by Equation 4.2.

Reprojection. A problematic situation arises if the refracted ray hits a real surface; because if the synthetic result of the rendering is used, information about the refracted real world image is missing. We solve this problem by using the image reprojection method similar to [50]. The per-ray data structure contains a flag *wasSpecular*. If the ray hits a specular object, this flag is set to true. If a diffuse real geometry is hit by a mixed radiance ray, the *wasSpecular* flag is checked. If the flag is set to true the diffuse object was hit after the specular reflection/refraction. In this case we need to use the outgoing radiance from the real object. The measured radiances from real objects are stored in the image obtained by the camera. If a diffuse surface is assumed, the outgoing radiance is the same for every outgoing direction. This fact allows us to use the radiance measured by the camera as the outgoing radiance from the real diffuse surface to the virtual refractive one. In order to obtain the correct measured radiance, the hitpoint of the diffuse surface is reprojected onto the image plane and its reprojected position in the image space coordinates is calculated. Then the video image, previously sent to the GPU memory, can be accessed and the measured value can be converted to radiance by inverse tone mapping. If the hitpoint contains a glossy material, the radiance obtained by the reprojection is still a good approximation of the outgoing radiance. The reprojection method is depicted in Figure 4.3 and the results of the refractive material rendering are shown in Figures 4.1, and 4.4.

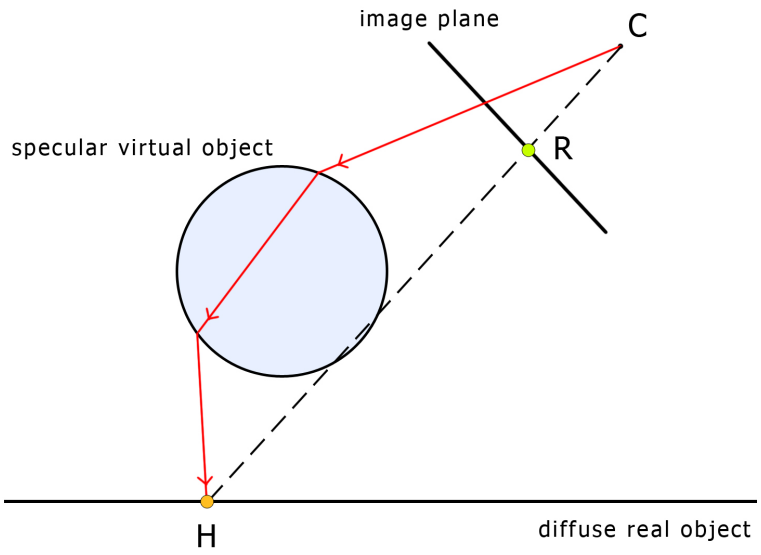

Figure 4.3: The reprojection method obtains the outgoing radiance from the diffuse real surface, seen through the refractive virtual object. Red rays are the rays sent from the camera to calculate the radiance coming from a certain direction. Point C is the center of projection. Point H is the hitpoint of the refracted ray with the diffuse real surface. In order to retrieve the outgoing radiance from point H, it is reprojected to the image plane to the point R. The outgoing radiance is then read from the camera image at point R.

Reflection of the Real Environment. In order to properly display the reflected/refracted environment in reflective specular objects, the developed system uses the hemispherical environment map obtained by the fish-eye camera. The surrounding environment is approximated as a distant light, coming from infinity, whenever reflected or refracted radiance is calculated. This assumption allows accessing the environment texture according to the ray direction. The environment texture is used as incoming radiance only if no real or virtual geometry has been hit. The texture is accessed in the miss program in the ray tracing pipeline and the brightness value is converted by inverse tone mapping to the radiance. An asynchronous image capturing, independent from the rendering thread, is used to capture the environment map. The captured image is updated to the GPU as a texture. The majority of the rays, pointing to the lower hemisphere, hit the surface of the scene. However, it can happen that these rays also miss any geometry. In this case, we reuse the captured environment image of the upper hemisphere in the lower hemisphere. This mirroring provides visually acceptable results of the radiance from missed rays.

4.1.2 Caustics

Caustics are important visual features that increase the amount of visual realism. They are created by the light reflected from specular surfaces to diffuse surfaces and then to the camera. The photon mapping [71] algorithm can efficiently calculate caustics in the scene. A new GPU implementation of photon mapping, utilizing the OptiX ray tracing engine, is presented in this thesis to achieve interactive frame rates while keeping quality of the created caustics high.



Figure 4.4: The resulting image produced by our AR rendering system. The image was rendered by shooting 9 rays per pixel to obtain a high-quality result and to reduce aliasing artifacts. 1M photons were shot in the photon shooting phase. The scene was rendered at 1fps, and the rendering rate can achieve 15 fps by decreasing the number of samples per pixel. There is one virtual glass monkey, casting caustics onto real objects, and three real cubes in the image.

In our implementation, a two-pass caustic generation algorithm is used. In the first pass, photons are emitted from the light source into the scene. If photons hit a specular virtual surface, they are reflected or refracted in the direction of specular reflection or refraction. If a photon hits a diffuse surface after reflection from a specular object, it is recorded in an array of photons. This array is later processed on the CPU, and a Kd-tree is created to allow faster search for near-by photons. This Kd-tree represents the photon map. In the second pass, the rays are traced from the camera through the image plane to obtain the radiance incoming from the scene. If a ray hits a non-specular surface in the scene, direct illumination is calculated and indirect caustic illumination is reconstructed from the photon map. An example of a caustic rendering can be seen in Figure 4.4.

In order to reconstruct the indirect illumination from the photon map at a certain scene point, density estimation techniques are applied. There are three main approaches for density estimation: k-nearest neighbor search, using a histogram, or kernel density estimation [162,170]. The K-nearest neighbor (KNN) search is the method that is often used in combination with photon maps. This technique reduces the variance while keeping the bias low. However, a high number of K has to be used to obtain accurate results. Because many samples are required, the

KNN search is a bottleneck of the radiance estimation from photon maps. Therefore, our system uses a kernel method to estimate illumination from the photon map. It allows us to perform a fast calculation of visually correct results. With kernel methods, there is always a tradeoff between bias and noise. A standard kernel method estimates the probability density function (pdf) $p(x)$ given N samples x_i by the equation [162, 170]:

$$\hat{p}(t) = \frac{1}{Nh^d} \sum_{i=1}^{N} \mathcal{K}(\frac{t - x_i}{h}) \tag{4.4}$$

$\mathcal{K}$ is a kernel function, h is the kernel bandwidth, d is the dimension of the domain of p, and t is the current position of the estimation. The accuracy of the kernel density estimation technique depends on the shape and bandwidth of the kernel. A kernel bandwidth selection is an important step. If the kernel is too wide, more bias is produced and if it is too narrow, more variance can be observed. Kernel estimation techniques with adaptive bandwidth were proposed in previous work [63, 170]. They use a different bandwidth for every sample according to its correctness, previously estimated density, or the number of surrounding samples. These techniques are often iterative and require additional computational time to find a good bandwidth. Our implementation uses a density estimation with a fixed kernel size. This approach can potentially produce bias and blur the discontinuities in caustics. However, we solved this problem by selecting a narrow kernel width. The variance is then reduced by increasing the number of photons that are shot into the scene. Using a fixed kernel size with a narrow kernel enables very fast photon search in a Kd-tree as well as fast density estimation. We use the Epanechnikov kernel (Equation 4.5), which is a standard in density estimation [170].

$$\mathcal{K}(x) = \begin{cases} \frac{2}{\pi}(1 - |x|^2) & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

A comparison of rendering with and without caustics as well as with other rendering features can be seen in Figure 4.5. In the top left image of the figure, the rendering of specular refraction and reflection is enabled. However, in this image only 1 ray per pixel was shot and aliasing artifacts can be observed. The top right image shows how aliasing artifacts disappear when supersampling is enabled. In the bottom images caustics rendering was added. Note the small caustic created under the glass sphere. In the lower left image the real yellow cube is blurred by DoF of the real lens. Therefore, the virtual sphere is visually incoherent because it is sharp. The DoF effect is enabled in the bottom right image.

An important part of the photon mapping is the sampling algorithm used when the photons are emitted. A well designed algorithm should sample the directions of photons which are likely to hit the specular objects. Jensen proposed a solution to this problem which projects specular objects to a hemisphere, centered at light position, to obtain only directions from which specular objects are seen [71]. Another approach is to approximate the geometry of specular surfaces by bounding volumes and to shoot rays into the resulting primitives. In our system, specular objects are approximated by bounding spheres for the purpose of photon shooting. In the scene loading phase, all geometry is traversed with the goal of finding caustic generators (specular objects). If a caustic generator is found, a bounding sphere, including all geometry of this object, is created.

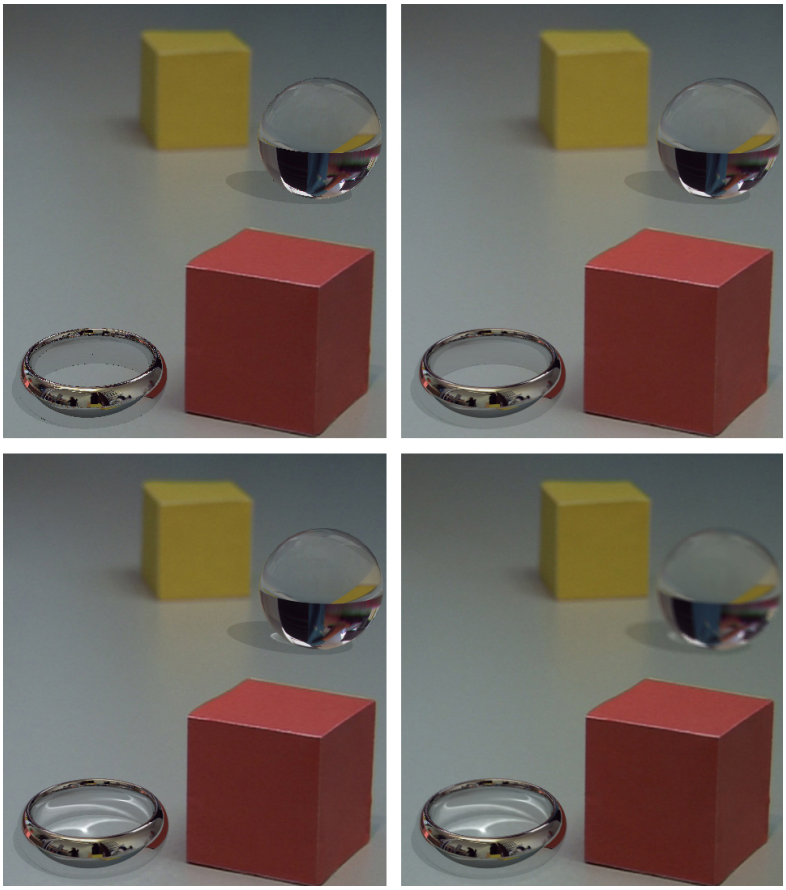

Figure 4.5: Comparison of rendering with different features. The scene contains a real red and yellow cube, a virtual refractive sphere and a virtual metal ring. (Top left) Rendering refraction and reflection using 1 ray per pixel and no caustic simulation at 27 fps. (Top right) Antialiasing is added using 25 rays per pixel at 7 fps. (Bottom left) Caustics are enabled using 150K photons. Rendering speed is 3 fps. (Bottom right) Depth of Field effect is enabled. Frame rate is 2.5 fps. Differences in images can be seen better in closeup.

If the projection of two bounding spheres on the sphere around the light source overlaps, they should be merged to one bounding sphere to avoid sampling the same area twice. The center and the radius of each bounding sphere is used in the photon shooting program as follows. First, one of the caustic generators is randomly selected according to the projected area of the bounding sphere. Then a disk, perpendicular to the direction from the center of the object to the light source, is sampled by stratified jittered sampling. A direction of the photon is then calculated as the direction from the light source position to the sampled point. By this sampling process, some photons may miss the caustic generator, however a high number of hitpoints is achieved. Our system can simulate caustics reflected on both real and virtual surfaces. The reflection of the virtual object and its caustic reflected on the real mirror surface can be seen in Figure 4.6.



Figure 4.6: Caustics can be created by both virtual and real specular objects. The virtual glass teapot casts a caustic to the virtual diffuse cube. Part of this caustic was created by the reflection of light on the real mirror. The generated caustic is correctly reflected in the real mirror. Moreover, the real environment is refracted in the teapot.

4.2 Diffuse Light Transport in Augmented Reality

This section presents a novel method for light transport simulation on diffuse surfaces in AR. The presented method utilizes the irradiance caching (IC) [193] algorithm. Our novel algorithm based on irradiance caching and differential rendering [27, 36] runs at interactive frame rates and produces a high-quality result of diffuse light transport in AR scenes. The parallel GPUs are utilized and ray tracing is combined with rasterization to achieve high-quality results while preserving interactivity. The developed method takes a frame of a live video stream and superimposes virtual geometry onto it, introducing light interreflections between real and virtual worlds, in real time (Figures 4.7, 4.8). Direct illumination is calculated by ray tracing from the camera using the system presented in Chapter 3. The indirect light is calculated by random hemisphere sampling by path tracing at a sparse set of points (cache records) in the scene. Rasterization is used to interpolate the indirect light between cache records by the irradiance cache splatting algorithm [45].

The limitation of single-bounce global illumination in irradiance cache splatting is overcome by employing recursive path tracing to evaluate the irradiance at the locations of cache records. This naturally enables multi-bounce global illumination and the irradiance in cache records is evaluated in an unbiased fashion.

The problem of splatting approaches is the limited information stored in screen space. No data are available when information from other locations is required, for example when computing depth of field or refraction, which require information behind the objects. We solve this problem by using a reprojection technique to access the indirect illumination splat buffer in the

ray tracing pipeline. This enables a high-quality depth of field effect (Figure 4.12) and screen space information reuse on refractive and reflective surfaces.

Our differential irradiance cache is a linear data structure which fits well to the memory capabilities of modern GPU hardware. We avoid any search in the irradiance cache by using irradiance cache splatting for rendering and a novel miss detection technique to find the places of new irradiance cache records. All steps of differential irradiance caching are described in the following sections.

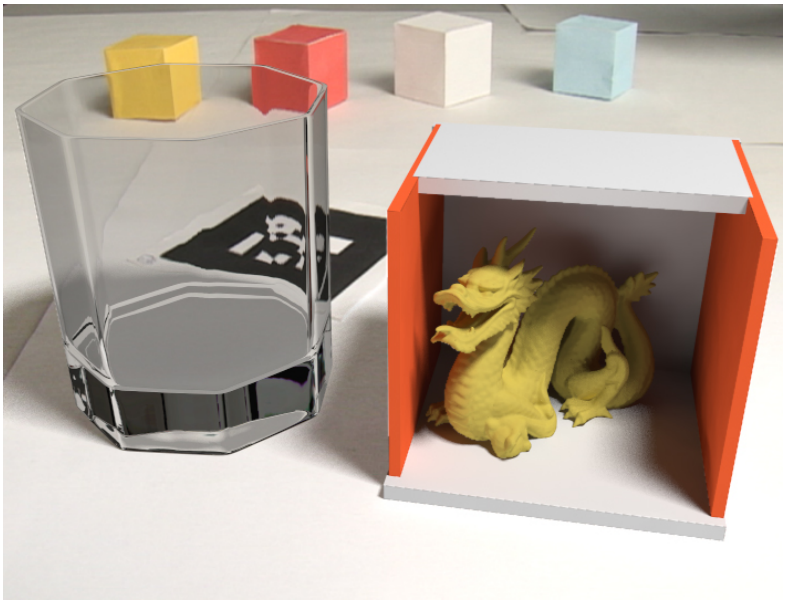

Figure 4.7: Virtual objects are placed into a real scene. Rendering is performed at interactive frame rates by differential irradiance caching in combination with ray tracing. The virtual glass correctly refracts the real cubes behind it. Diffuse indirect illumination on the virtual box with the dragon is calculated by the differential irradiance caching algorithm.

4.2.1 Differential Irradiance Caching

The indirect light component of the terms I_m and I_r from the differential rendering equation (Equation 2.20) is calculated by the presented differential irradiance caching algorithm. Both indirect I_m and I_r are calculated together. The differential irradiance caching algorithm evaluates the accurate differential and mixed irradiances at sparse scene locations and then interpolates the results in screen space to introduce diffuse indirect illumination. The calculation of irradiances at irradiance cache records is performed by Monte Carlo numerical integration. The hemisphere above the location of the cache record is sampled by random rays and the light integral is evaluated by tracing the random light paths. The calculation of differential and mixed radiances in one step and their integration into irradiances are described in Section 4.2.2. The

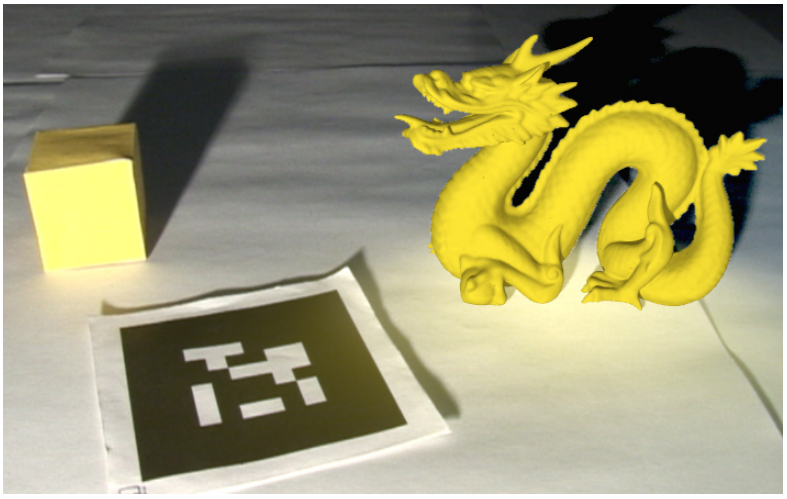

Figure 4.8: Indirect light transport between real and virtual worlds. The virtual dragon causes color bleeding on the real table similarly to the real cube.

input : Vector <IrradRecord> IrradCache, virtual and real scene
output: Buffer IndirectLight, Vector<IrradRecord> IrradCache

```
1  begin
2      UpdateTemporalData(IrradCache)
3      RecalculateIrradiance(IrradCache)
4      IndirectLight = SplatIrradiance(IrradCache)
5      Vector <IrradRecord> NewRecords = DetectCacheMiss(IndirectLight)

6      while NewRecords.IsNotEmpty() do
7          EvaluateIrradiance(NewRecords)
8          IndirectLight += SplatIrradiance(NewRecords)
9          IrradCache.Append(NewRecords)
10         NewRecords.Clear()
11         NewRecords = DetectCacheMiss(IndirectLight)
12     end
13 end
```

Algorithm 4.1: Differential irradiance caching. This algorithm calculates indirect light for each pixel of the output buffer.

steps of differential irradiance caching are depicted in Algorithm 4.1 which runs every frame. It updates the irradiances in the cache records and splats them to image space.

Different steps of the differential irradiance caching algorithm have the following functionality. Procedure $UpdateTemporalData$ updates the irradiance cache and removes old records. For this purpose, irradiance cache record has the variable t, which denotes the number of frames since it was last updated. If $t = t_{max}$, the record is reevaluated or removed from the cache.

In our implementation, we use a constant t_{max} value, however it can be changed in the future to an adaptive value inversely proportional to the speed of changes in geometry and lighting. The procedure $RecalculateIrradiance$ recalculates the irradiance for outdated cache records. The same sequence of random ray parameters is used every time the record is recalculated to achieve temporal coherence. The function $SplatIrradiance$ splats all the valid irradiance cache records to the $IndirectLight$ buffer by rasterization. In our method, the irradiance cache is a linear data structure. Our algorithm does not search in the irradiance cache. Therefore, we can avoid complex non-linear data structures. In addition, a linear irradiance cache fits well to the GPU architecture which has a small GPU cache. The limited GPU cache is causing latency if memory is accessed intensively. Thus, our method is very advantageous, because it avoids the search in the irradiance cache data structure. The function $DetectCacheMiss$ processes the resulting splat buffer and finds places where the irradiance information is missing. The new cache records are initialized at these locations. This function runs on the CPU. An asynchronous memory transfer between GPU and CPU is utilized to transfer the splat buffer. Cache miss detection is described in detail in Section 4.2.4. The procedure $EvaluateIrradiance$ calculates the differential and mixed irradiances at locations of cache records in parallel on the GPU. The differential irradiance calculation is described in Section 4.2.2. Algorithm 4.1 iteratively calculates indirect illumination for given camera pose. The $IndirectLight$ buffer is iteratively filled in the $while$ cycle.

4.2.2 Differential Irradiance Calculation

The differential irradiance in irradiance cache records is evaluated in parallel by utilizing path tracing on the GPU. The light interaction between the real and virtual objects is calculated in both ways. Both mixed and real irradiances are evaluated in a single pass. Real irradiance is computed as the integral over the product of real incoming light from a hemisphere above a surface point and the cosine function. The hemisphere is rotated with respect to the surface normal. The resulting vector contains a single value for each spectral component (color channel). This irradiance calculation takes only real objects into account:

$$E_r(x) = \int_{H^+} L_{ir}(x, \vec{\omega}') \cos\theta' \, \mathrm{d}\vec{\omega}' \tag{4.6}$$

$E_r(x)$ is the real irradiance in point x calculated by integrating the incoming real radiance L_{ir}. This radiance is calculated using only the geometry of the real scene. θ' is the angle between incident light direction $\vec{\omega}'$ and the surface normal. Analogously, the mixed irradiance is the integrated product of incoming light with the $\cos\theta'$. However, in this case both virtual and real objects are taken into calculation. The mixed radiance L_{im} is integrated:

$$E_m(x) = \int_{H^+} L_{im}(x, \vec{\omega}') \cos\theta' \, \mathrm{d}\vec{\omega}' \tag{4.7}$$

We can calculate the differential irradiance $E_d(x)$ by subtracting the real irradiance $E_r(x)$ from the mixed irradiance $E_m(x)$:

$$E_d(x) = E_m(x) - E_r(x) \tag{4.8}$$

The results is a difference in indirect lighting, caused by adding virtual objects. After multiplication with diffuse albedo, the result can directly be added to the differential solution of direct lighting to calculate the differential solution of global lighting. We add this difference to the video image to introduce light changes.

In order to evaluate the integral from Equations 4.6 and 4.7, we sample the integration domain H^+ by shooting random rays. Monte Carlo numerical integration is used to estimate the irradiance [133]:

$$E(x) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(x, \vec{\omega}_i') \cos \theta_i'}{p(\vec{\omega}_i')} \tag{4.9}$$

where $\vec{\omega}_i'$ is the random direction on the hemisphere, $p(\vec{\omega}_i')$ is the probability of selecting $\vec{\omega}_i'$ in the sampling process, and N is the number of samples. If the rays are sampled from a cosine-weighted distribution, $\cos \theta'/\pi$ [91], Equation 4.9 can be rewritten to:

$$E(x) \approx \frac{\pi}{N} \sum_{i=1}^{N} L_i \tag{4.10}$$

Mixed and real radiances are calculated by equations above. Multiple ray types are used to calculate them together. We use four different ray types similarly to the one-pass compositing described in Section 3.3. The four ray types are: **mixed radiance ray, real radiance ray, mixed shadow ray, and real shadow ray**.

Mixed radiance and mixed shadow rays can intersect both real and virtual geometry while real radiance and real shadow rays can intersect real geometry only. Mixed radiance rays return both mixed and real radiances while real radiance rays return only real radiance. The per-ray data structure contains four variables: mixed radiance, real radiance, path depth, and R_i, which is the minimum distance to objects hit by cast rays. R_i is later used in irradiance splatting to specify the area of influence of the irradiance cache record. Tabellion and Lamorlette [173] proposed to use the minimum distance to nearby objects to avoid missing important geometric details. Primary rays, shot from the locations of irradiance cache records, are always mixed radiance ray types because both mixed and real radiances are required. If a ray hits a surface, four different cases can happen similarly to the cases described in Section 3.3:

1. **If real geometry is hit by a mixed radiance ray** then a reflected mixed radiance ray is shot in a random direction on the hemisphere above the surface, mixed shadow rays and real shadow rays are shot towards all light sources to evaluate visibility. If lights are visible, light contribution is calculated to both mixed and real radiances and the contribution from the reflected ray is added.

2. **If a virtual object is hit by a mixed radiance ray** then a real radiance ray which continues in the direction of current ray is shot. This ray will evaluate the real radiance ignoring the virtual object at the current location. Additionally, the reflected mixed radiance ray is shot in a random direction towards the hemisphere above the surface. This ray will evaluate the reflected mixed radiance. The mixed shadow rays are shot towards positions of light sources and the direct contribution of the mixed radiance is calculated.

3. **If a real object is hit by a real radiance ray** then real shadow rays are cast and the direct light contribution is calculated. The reflected real radiance ray is shot in a random direction on the hemisphere above the surface and the reflected real radiance is evaluated.

4. In the fourth case **virtual geometry is hit by the real radiance ray**, however this case is prevented in the ray-triangle intersection routine. It would not contribute to the result since the real radiance is not influenced by virtual objects.

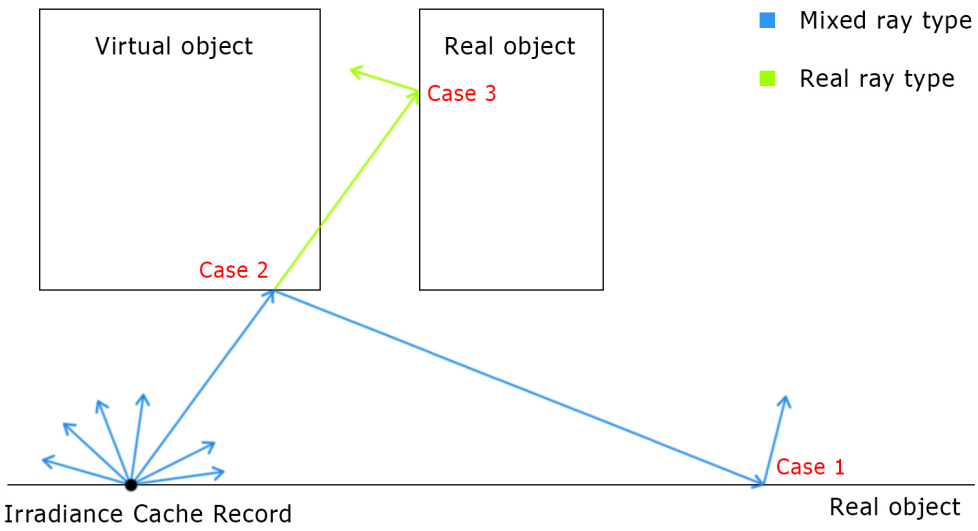

Figure 4.9: Differential irradiance calculation at the irradiance cache record. Mixed rays are shot in random directions towards the hemisphere above the surface. The mixed rays calculate both real and mixed radiances coming from specified directions. Mixed and real radiances are integrated to calculate corresponding irradiances. The shadow rays are omitted in this figure for clarity. Nevertheless, they are included in the calculation.

The evaluation of differential irradiance in a cache record by shooting a high number of mixed rays is depicted in Figure 4.9. Since both real and mixed irradiances are calculated by applying Equation 4.10 to evaluate both Equations 4.6 and 4.7, the differential irradiance can be calculated by Equation 4.8. Differential and mixed irradiances are stored in each irradiance cache record. These irradiances are later used in irradiance cache splatting. Differential irradiance is splatted if current pixel belongs to the real object. Differential irradiance represents the change of indirect lighting which is caused by adding virtual objects. Mixed irradiance is used if current pixel belongs to the virtual object. It represents the indirect illumination reflected from virtual and real objects. The Monte Carlo evaluation of irradiance by recursive path tracing with random hemisphere sampling has the advantage of calculating multiple bounces of indirect illumination. In our experiments, we shot from 100 to 4000 mixed rays for the evaluation of irradiance in each cache record.

4.2.3 Differential Irradiance Splatting

The indirect illumination, calculated at cache records, is splatted by the irradiance cache splatting algorithm (Section 2.2.4). The differential and mixed irradiances are splatted in a fragment shader in the GPU rasterization pipeline. The quads centered at irradiance cache records' locations are sent to the vertex shader. Quads are scaled proportionally to the minimum distance to surrounding objects R_i in a vertex shader. The pixel shader uses the weight w_i to accumulate the differential or mixed irradiances in each pixel. The weight for contribution of each record to each pixel is given by the equation proposed in [173] (Equation 2.15). If a pixel belongs to a real object, differential irradiance is added. Mixed irradiance is added for pixels belonging to virtual objects. Finally, the accumulated irradiances are addressed in the ray tracing pipeline by the reprojection technique (Section 4.2.5). The accumulated values are divided by accumulated weights and the final indirect irradiance contribution is calculated by Equation 4.11. This irradiance is multiplied by surface albedo in ray tracing to calculate the reflected indirect light.

$$E(x) = \frac{\sum_{i \in pixel_{xy}} E_i(x) w_i(x)}{\sum_{i \in pixel_{xy}} w_i(x)} \tag{4.11}$$

4.2.4 Cache Miss Detection

Irradiance caching typically stores cache records in a nonlinear data structure, e.g. in a tree. The set of contributing irradiance cache records is searched for every pixel. If no record is contributing to a pixel, a cache miss is detected and a new record has to be calculated. The search for a set of contributing records has the complexity $O(log(n))$, where n is the number of irradiance records. However, this search is invoked many times and non-linear memory access is not efficient on current GPU architectures. Therefore, we propose an irradiance cache as a linear vector of cache records without any spatial relationships. Cache records are added sequentially whenever they are created. This cache can be easily rendered. During rendering, the irradiance cache splatting algorithm splats all records directly to the framebuffer independent of the order of cache entries.

In case of linear cache organization, the problem of cache miss detection arises. Linear search in a cache has complexity $O(n)$ and is, therefore, slow if invoked for every pixel. In order to solve this problem and make cache miss detection independent of the number of cache records, we propose to use the irradiance splat buffer to detect cache misses. Splat buffer locations with zero weights can be used to select the positions of new cache records and the image can be filled in an iterative manner. Our cache miss detection algorithm works as follows. The irradiance splat buffer is asynchronously transferred from GPU to CPU by direct memory access (DMA). The splat buffer is divided into tiled rectangular regions and pixels are traversed within each tile. If at any point the accumulated weight, calculated by the irradiance splatting algorithm, is zero, a new cache record is created. In this case the search for cache misses stops at a tile where a cache record was created to avoid oversampling by multiple neighboring cache records. The rectangular portions of the splat buffer are processed and pixels with zero weight w_i are found in a linear fashion. Iterative refinement is needed to cover the whole image by cache records, but it can be performed in a progressive manner by exploiting temporal coherence. In order to

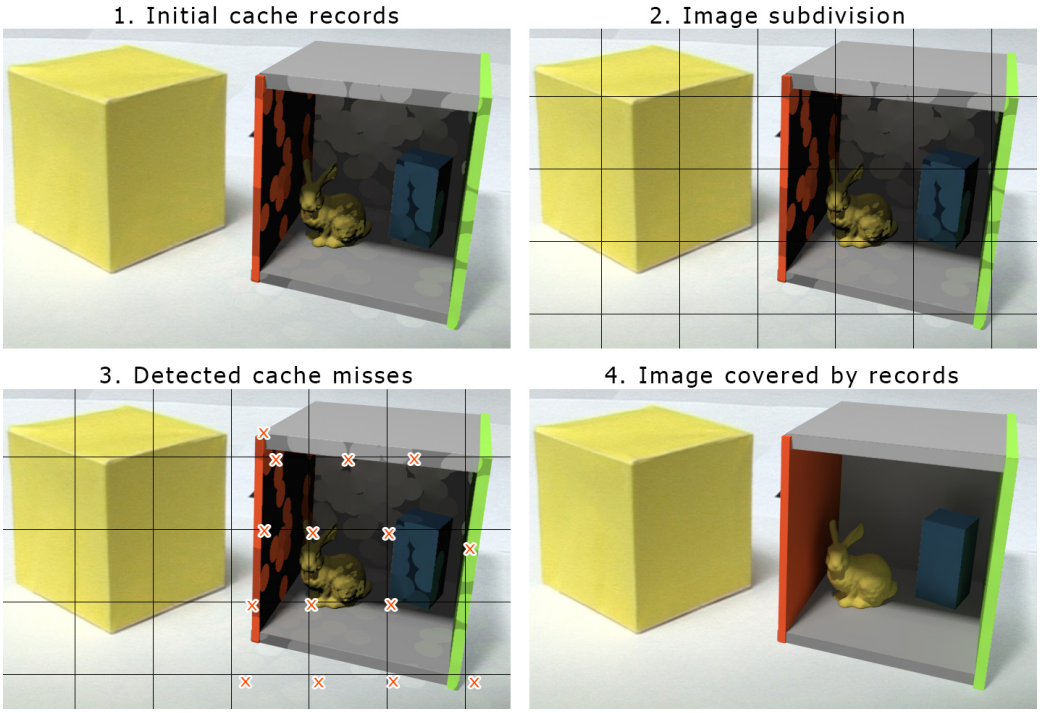

Figure 4.10: The steps of cache miss detection. The first image shows the rendering with initial cache records. In the second step, the image is subdivided into the tiled regions. The third image shows the positions of new irradiance cache records initiated at locations of first pixel with zero weight in each tile. By this procedure, the whole image is iteratively covered by the cache records. The fourth image shows the rendering with full cache.

avoid repetitive search in the same pixels within a tile, the position of the last checked pixel can be stored and used in the next iteration. A speedup of cache miss detection can be achieved by running the search for each tile in parallel. The cache miss detection procedure, using the splat buffer, is depicted in Figure 4.10.

4.2.5 Reprojection to the Splat Buffer

Splatting approaches have been useful in computer graphics for fast energy projection from world positions to the specified buffer. Different problems were solved by introducing splatting [45, 99]. The irradiance cache splatting method enables fast projection of cache records' energy into the framebuffer. The drawback of splatting methods is that the information is stored per pixel in screen space and therefore the depth of field, antialiasing, or refraction of a splatted value in refractive objects cannot be calculated from this one-pixel information.

We propose to use a reprojection method and reuse the information from different locations of the splat buffer when needed. This method is also used in Section 4.1.1 to introduce information from real scenes on reflective and refractive objects. We changed the reprojection to

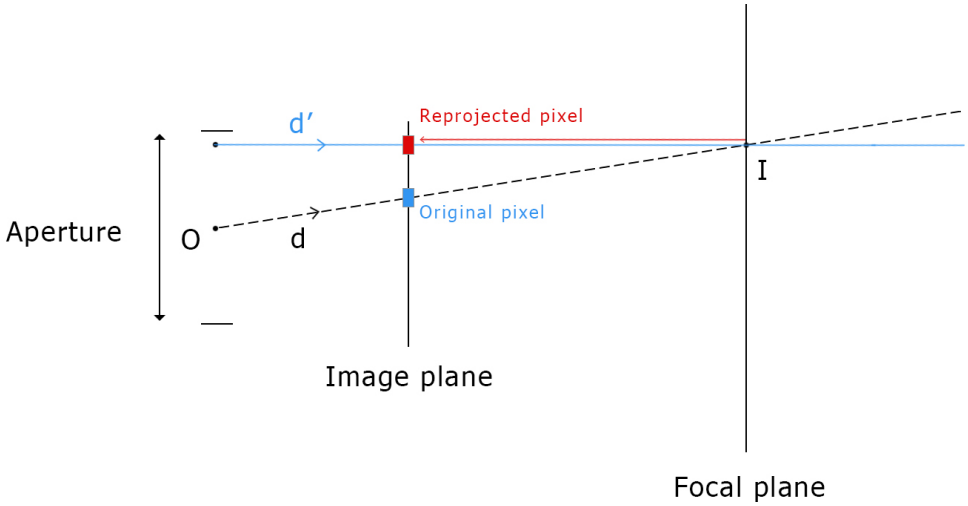

Figure 4.11: The finite sized aperture model, described in Section 3.4, uses the original pixel (marked as blue) to write the final color and to read the indirect irradiance from the splat buffer. The reprojection technique calculates the reprojected pixel (marked as red) and reads the indirect irradiance from this position. The new reprojected pixel provides valid indirect irradiance in the case of depth of field, refraction, or reflection. The final color value is still written to the original pixel, marked as blue. Vector d' is the new ray direction created by sampling the aperture area.

operate on a splat buffer and to access irradiance information of a splat buffer by multiple rays. If indirect lighting information is needed, we reproject the ray hitpoint back to the splat buffer and then read the information from the reprojected position. If the reprojected position is outside of the splat buffer, the splat buffer is repetitively tiled in the 2D plane to provide the data. The reprojection technique is depicted in Figure 4.11. Reprojection allows addressing more pixels in the irradiance splat buffer by shooting rays with different aperture samples. These multiple values of indirect illumination are finally blurred by filtering ray samples to create the depth of field effect. Visually plausible results of depth of field and refraction can be created. A comparison of rendering with reprojection and without reprojection to the splat buffer can be seen in Figure 4.12. Our method is the first using reprojection to access splat buffer data and it can be applied to any splatting approach. Furthermore, the reprojection method can be generalized to improve access to data in any rendering approach that operates in screen space.

4.3 Global Illumination System for Augmented Reality

The algorithms, presented in this chapter, are included into our global illumination system for AR to simulate complex light transport. The core of our system is described in Chapter 3. Additional extensions calculate the specular light transport in AR scenes, including effects like specular reflection, refraction, and caustics (Figure 4.13), at interactive frame rates. The caustics are calculated by interactive photon mapping described in Section 4.1.2. Moreover, differential irradiance caching calculates diffuse indirect illumination and stores it in the splat buffer. This

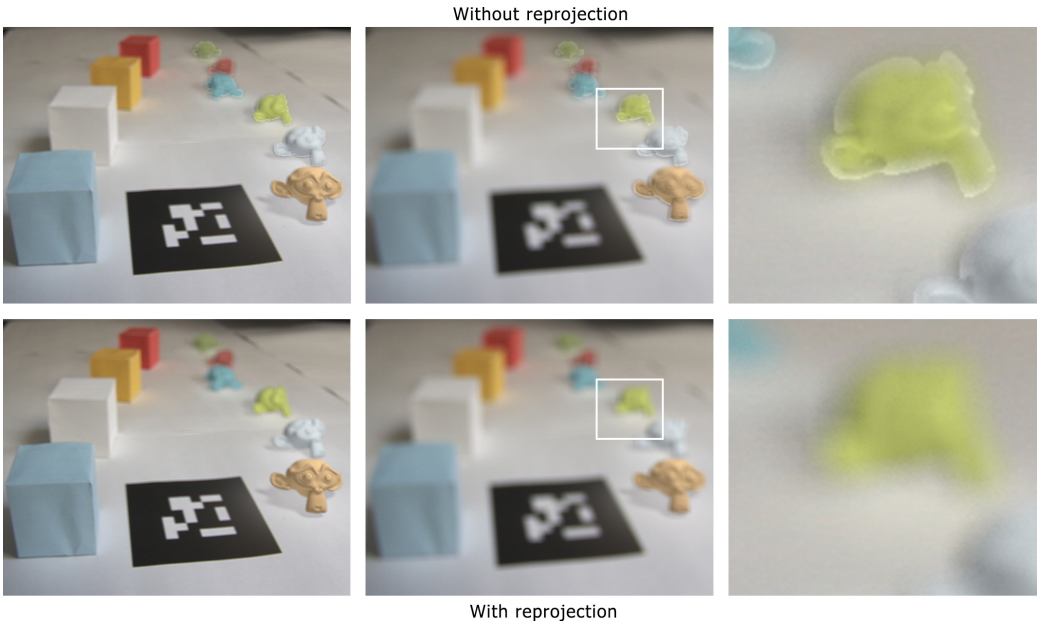

Figure 4.12: Rendering without (top row) reprojection to the splat buffer and with (bottom row) reprojection. Virtual monkeys are rendered in the scene together with real cubes. Both real and virtual cameras have opened aperture causing a depth of field effect. Artifacts, caused by direct reuse of irradiance information from the splat buffer in every pixel, can be seen in the top row. Properly smoothed edges are achieved by introducing reprojection (bottom row). In the left column the camera is focused to the frontmost monkey. Middle column shows a defocused camera. Right column is the close-up from rectangles in the middle column. Note the incorrectly sharp ghost edges which appear in images without reprojection.

buffer is later addressed in the ray tracing pipeline by reprojection. The difference of indirect mixed and indirect real illumination is added to the result of differential direct illumination.

The differential irradiance caching method can be extended to differential radiance caching by storing directionally dependent radiance instead of irradiance in every cache record. Radiance caching approaches [89, 90] use spherical harmonics functions to store the radiance in an efficient way. The extension to differential radiance caching can lead to fast and high-quality light transport in glossy scenes. Our implementation does not include radiance caching, but it can be added in the future.

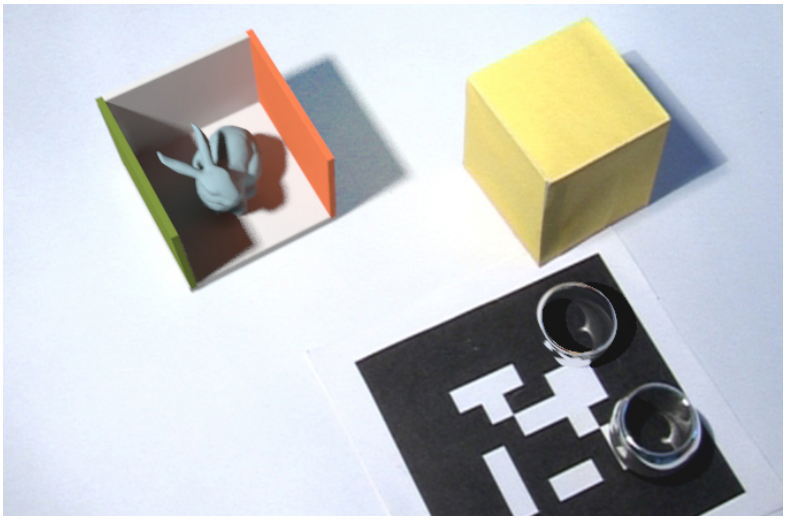

Figure 4.13: Result of a rendering with our GI system for AR calculating diffuse and specular light transport. The virtual ring (top) casts a caustic to the real table. The real ring (bottom) casts a similar caustic. Diffuse GI in the virtual box with the bunny is calculated by differential irradiance caching.

4.4 Differential Progressive Path Tracing

Physically based path tracing algorithm, which simulates global illumination in an unbiased manner, is of high interest in AR and other applications. However, the high sampling rate, required for converged solution, is not feasible to achieve in real time. We propose a solution of this problem that uses a real-time preview and progressive rendering in AR. Our solution can be especially beneficial for cinematic relighting and movie previsualization.

This section presents a rendering technique which uses a physically based path tracing to provide an unbiased solution of image synthesis for AR. The composited video can be rendered in preview quality during interaction. Additionally, progressive refinement can be used to converge to the accurate solution. Modern GPUs are employed to increase the speed of the path tracing algorithm. We use an AR scenario to allow users to interact with virtual objects, inserted into the real world. This scenario can be especially useful during movie production where virtual and real content is mixed. A novel one-pass differential progressive path tracing algorithm is introduced which quickly calculates two illumination solutions needed for compositing. Moreover, a rendering framework for previsualization and relighting in AR is presented. This framework operates in two main modes allowing interaction and high-quality convergence: An interactive preview mode and a progressive refinement mode. The problem of noise in the interactive preview mode is solved by allowing the user to increase the quality of the result by increasing the sampling rate. The user can switch to the progressive refinement mode any time to see the full quality solution (Figure 4.14).

Rendering synthetic objects into a real scene requires the estimation of real lighting. For this purpose we use a camera with a fish-eye lens to capture the environment illumination. Two

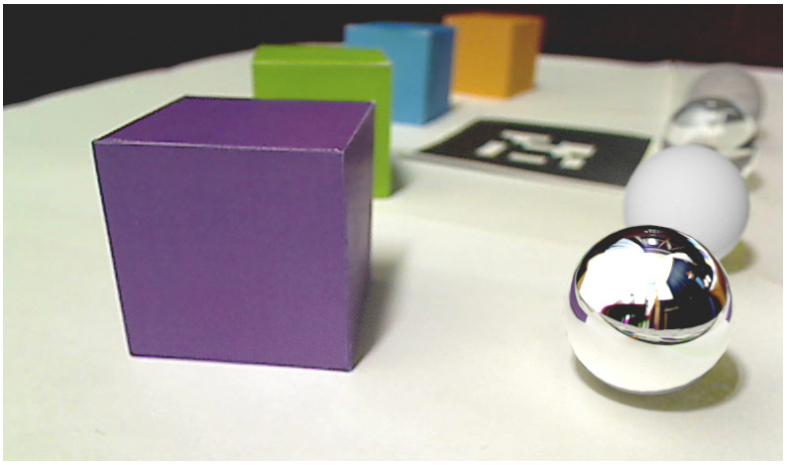

Figure 4.14: Converged AR image in progressive refinement mode. Cubes on the left side of the table are real and spheres on the right are virtual. The converged image was progressively rendered within 1 minute. Note the similar depth of field effect on real cubes and virtual spheres. The real environment is correctly reflected in the virtual metal sphere.

lighting algorithms are available in our system: (1) Light source estimation by processing an environment map or (2) image-based lighting in which the whole environment map is used to light the scene. In the presented framework, a physically based camera model with finite sized aperture is used which enables the simulation of a high-quality depth of field (Figure 4.14). Light paths needed to generate caustics are difficult, or in some cases impossible, to simulate by path tracing. We overcome this problem by using photon mapping to handle these light paths separately (Figure 4.15). Thus, our framework is capable of simulating complex global lighting between real and virtual scenes. Our framework naturally supports reflection and refraction on specular surfaces. Moreover, antialiasing can be enabled by supersampling an area of pixels and it is inherently supported in progressive refinement mode. Artificial lights can be added to the rendering to support artistic lighting in the process of movie production. If a high-quality result is needed, the captured video, lighting and camera position can be recorded during interaction. Afterwards, the scene can be rendered in post-processing to create the full movie quality.

The core of the presented previsualization framework is the differential progressive path tracing algorithm running on the GPU. This algorithm uses Monte Carlo integration to evaluate the global light transport in an AR scene. In order to calculate the light exiting from a scene point x in a direction ω towards the camera, the integration of incoming light at this point is necessary (Equation 2.2). The recursive path tracing algorithm [73] (Section 2.2.2) is used to perform this numerical integration. In original differential rendering, the integral from the rendering equation has to be evaluated two times to be used in the compositing equation (Equation 2.20). Our one-pass differential rendering algorithm can produce both solutions together. Four ray types are used to calculate mixed and real radiances. Additionally, four rules, described in Section 3.3, are applied to handle the ray types. The algorithm starts by shooting the mixed radiance rays from the camera towards the scene. The numerical integration is performed at the hitpoints of

primary camera rays with geometry to evaluate the contribution of light reflected from these surface points. The random light paths are sampled from these points and the visibility of a light source is tested in every hitpoint to calculate the light contribution. Mixed radiance rays evaluate both mixed and real radiances which are needed for the differential rendering equation. The user can set up the number of samples, used during interaction, to control the performance and quality of the live preview. If the user requests the full quality image, the progressive algorithm starts to run and the samples are accumulated to obtain the final value.

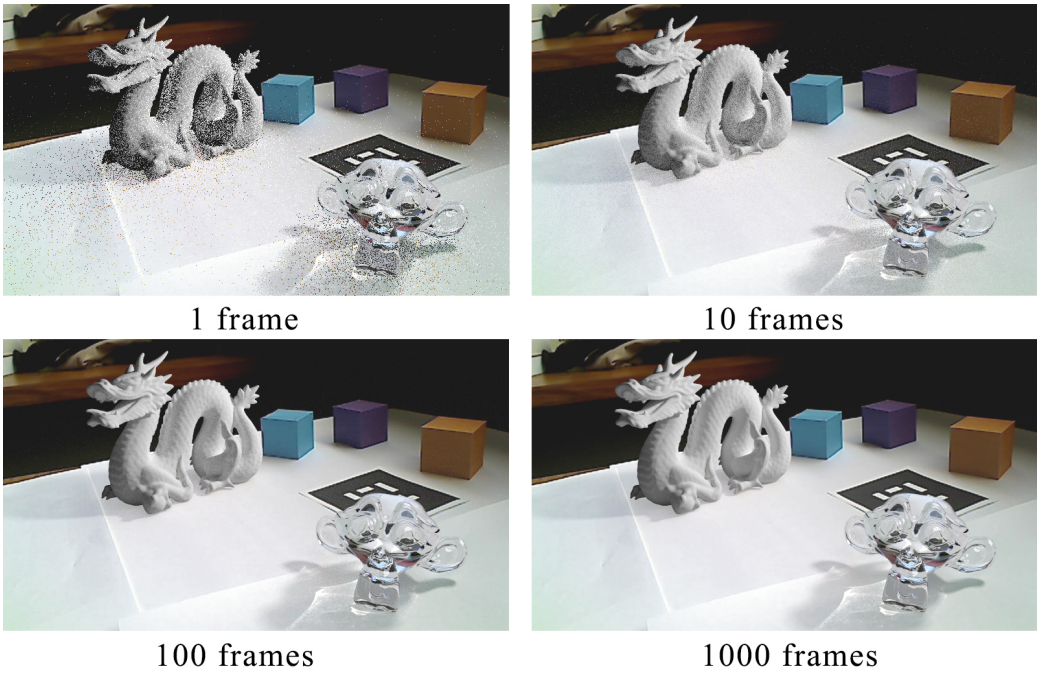

Figure 4.15: Rendering by differential progressive path tracing. The scene contains a virtual monkey and a virtual dragon (27K triangles). The solution after 1, 10, 100 and 1000 frames of refinement is shown. Caustics are simulated by photon mapping.

If the user requests to see the converged AR image, the system stops tracking and camera capturing and progressively converges to a high-quality image without noise. The last camera image, pose and lighting are used for progressive rendering. Newly calculated frames are accumulated to the accumulation buffer and the average solution of all samples is progressively calculated. The new calculated radiances are blended with the accumulated values:

$$L_{acc} = \frac{(n-1)L_{acc} + L_o}{n} \tag{4.12}$$

where L_{acc} stands for the accumulated radiance value. This value is calculated as an average of all samples from previous frames. L_o is the result of the compositing equation of the differential rendering algorithm (Equation 2.20) and n is the number of frames in progressive refinement mode. The results of progressive rendering in AR can be seen in Figure 4.15.

4.4.1 Previsualization Framework

We created a novel previsualization and relighting framework capable of delivering a high-quality rendering result by utilizing the differential progressive path tracing algorithm. The presented framework can operate in two different modes:

- **Interactive preview mode:** The 3D pose of the camera is tracked and the user can interact with the scene in real time. The quality of the output can be controlled by the number of samples per pixel. The preview mode suffers from noise caused by the high variance in Monte Carlo integration.

- **Progressive refinement mode:** This mode is used when a high-quality converged image is required. The interaction is paused and the last captured camera image is used for compositing. The synthetic image is progressively refined as more samples are used. In our experiments, a fully converged image could be obtained within 2 minutes in average.

The following steps are performed per frame if the interactive preview mode is enabled:

- The environment lighting is captured by the camera with a fish-eye lens and the positions of light sources are estimated if light source estimation is enabled.

- The observer camera image is captured and a 3D pose is calculated by the tracking system.

- Data from the previous steps are sent to the path tracing running on the GPU.

- Mixed radiance rays are sent from the camera position towards the scene to evaluate both mixed and real radiances and to composite virtual objects with the real camera image. The direct illumination is evaluated at hit points of rays with the geometry. The indirect reflected light is estimated by shooting rays in random directions within the hemisphere above the hit point. This calculation uses Monte Carlo numerical integration.

A low number of ray samples (1-50) is used during interaction which leads to a noisy preview image. The sampling rate is set by the user depending on the required quality and speed. Temporal coherence between successive frames is achieved by using the same random parameters in each frame. A video can be recorded during the interactive session and the full quality movie can be generated in post-processing. The quality and speed evaluation of differential progressive path tracing is shown in Section 5.1.4.

When the user switches to the progressive refinement mode, the tracking of the camera, light source estimation, and camera capturing are paused. The path tracing uses the data from the last interaction frame and starts to accumulate the calculated light values. The progressive refinement is displayed to the user. The user can either wait until the image is fully converged to see the final result or can interrupt the progressive refinement to continue in interactive mode.

4.4.2 Interaction

During the interactive preview mode the user is able to interact with virtual objects to see the rendering result in preview quality. The user can switch between interactive preview mode and progressive refinement mode by pressing a button. We use marker-based visual tracking to estimate the 3D pose of the camera although any other tracking system could be used. Our rendering framework supports dynamic geometry, materials, lighting and camera. Therefore, interaction between virtual and real worlds can be achieved in real time. Our current implementation does not use the movement of real objects, because predefined real geometry is used. The system can be extended by automatic scene reconstruction.

4.5 Implementation

This section describes the implementation of advanced global illumination algorithms presented above. The core implementation of ray tracing based rendering in AR is extended by adding the photon mapping and irradiance caching to simulate complex GI effects. Moreover, the physically based reflection and refraction which employs the Fresnel term is added.

4.5.1 Photon Mapping

The main steps of the rendering and compositing with photon mapping are depicted in Figure 4.16. For every frame, the images from the video camera and the fish-eye camera are captured and sent to the GPU. The environment image is processed and light source positions are found (Section 3.2.1). A visual marker is detected in the video image and position and orientation of the camera are estimated. The first GPU step is photon mapping. Photons are then processed on the CPU and a Kd-tree is created. The final rendering step is GPU ray tracing from the camera position. All calculated results, together with geometry and materials of real and virtual scenes, are used in this step. Additionally, density estimation is performed on the GPU at every ray hit of diffuse surface to estimate the indirect illumination from the photon map. Finally, the result of the rendering is composited with the real video image and displayed on the screen.

4.5.2 Differential Irradiance Caching

The implementation of differential irradiance caching utilizes the parallel power of modern GPUs. The core of this algorithm is the path tracing on the GPU which evaluates the differential irradiance at locations of cache records. The direct illumination and specular indirect illumination are calculated by Whitted style ray tracing [195]. Similarly to the implementation of the core system, the OptiX [128] engine is also used for the irradiance caching implementation. Multiple GPU cores can be automatically utilized by OptiX. We implemented the irradiance cache splatting in OpenGL using GLSL. The irradiance cache splatting needs geometry information for each pixel. Therefore, the positions and normals are firstly rendered to the textures in screen space. Then, these textures are used in the irradiance cache splatting shader to calculate the contribution of a particular cache record to the current pixel. The irradiance cache splatting

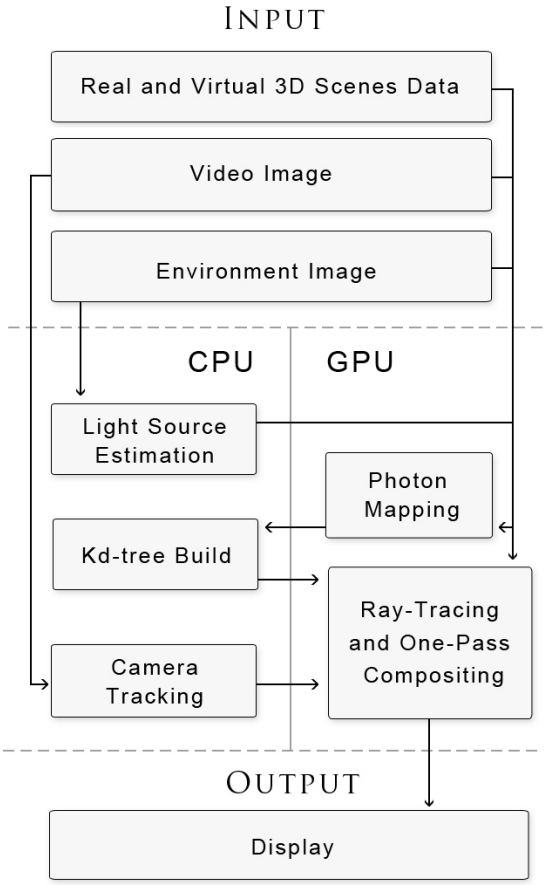

Figure 4.16: Overview of rendering and compositing with photon mapping. Arrows indicate the data flow between different components.

shader is invoked on the pixels which belong to the quad centered at the position of the specific irradiance cache record.

The light source estimation step is performed in each frame and therefore dynamic lighting is supported in our system. As the Optix engine allows for a dynamic Kd-tree rebuild, we also support dynamic geometry and materials. The geometry of the real scene is required in differential rendering approaches. We use a predefined model of the real scene. However, since our method fully supports dynamic geometry and materials, the implementation can be extended with a real-time 3D scene reconstruction approach.

CHAPTER 5

Evaluation and Results

This chapter evaluates the algorithms for high-quality rendering and compositing in AR, presented in previous chapters. Different methods are used for this evaluation. The rendering time of the presented algorithms can be measured to asses their speed. The rendering speed of our methods is evaluated in Section 5.1. Moreover, the rendering result of different methods is compared to the reference image to asses their quality. The reference solution can be either rendered by unbiased offline rendering method or can be taken as photograph of real objects corresponding to the virtual ones. A difference to the reference solution is used in our evaluation to compare our algorithms with differential instant radiosity. Quality comparisons of our methods are described in Section 5.2.

In addition to the exact measurements of rendering speed and quality, the user perception plays an important role for evaluation in AR. The results of several user studies are described in this PhD thesis. The effects of different visual features, rendered by our system, on visual realism perceived by the users are described in Section 5.3. In addition to visual realism, an essential attribute in AR is the sense of presence perceived by the user. It expresses the feeling of the user that virtual objects are present in the real environment. We studied the effect of global and direct illumination on the sense of presence in AR. This study is described in Section 5.4.

5.1 Rendering Speed

In order to evaluate the efficiency of the rendering algorithms, the rendering speed can be measured. This measure depends on used hardware, but it can give a reasonable estimate of the efficiency of a particular algorithm. This section shows the results of the rendering speed for the presented algorithms. Different factors impact the rendering speed in ray tracing. Some of them are: geometry count, number of rays shot per pixel, used materials, output resolution, and others. In this section, the rendering speed is measured with different parameters to show dependency of speed on these factors.

5.1.1 Depth of Field

We evaluated the performance of rendering with the depth of field effect and the physically correct camera model. The rendering with different quality levels was measured. The results of rendering speed can be seen in Table 5.1. The rendering speed is denoted in frames per second (fps). Our method is capable of running at interactive frame rates even at high-quality settings. Lower quality previews can be rendered in real time. The quality and speed of DoF rendering with variable sampling rates can be seen in Figure 5.1.

Rays per pixel	Primary rays count	fps
1	0.4 M	59
25	10.4 M	12
49	20.3 M	7

Table 5.1: Frame rates of DoF rendering in AR by ray tracing at the resolution of 720x576.

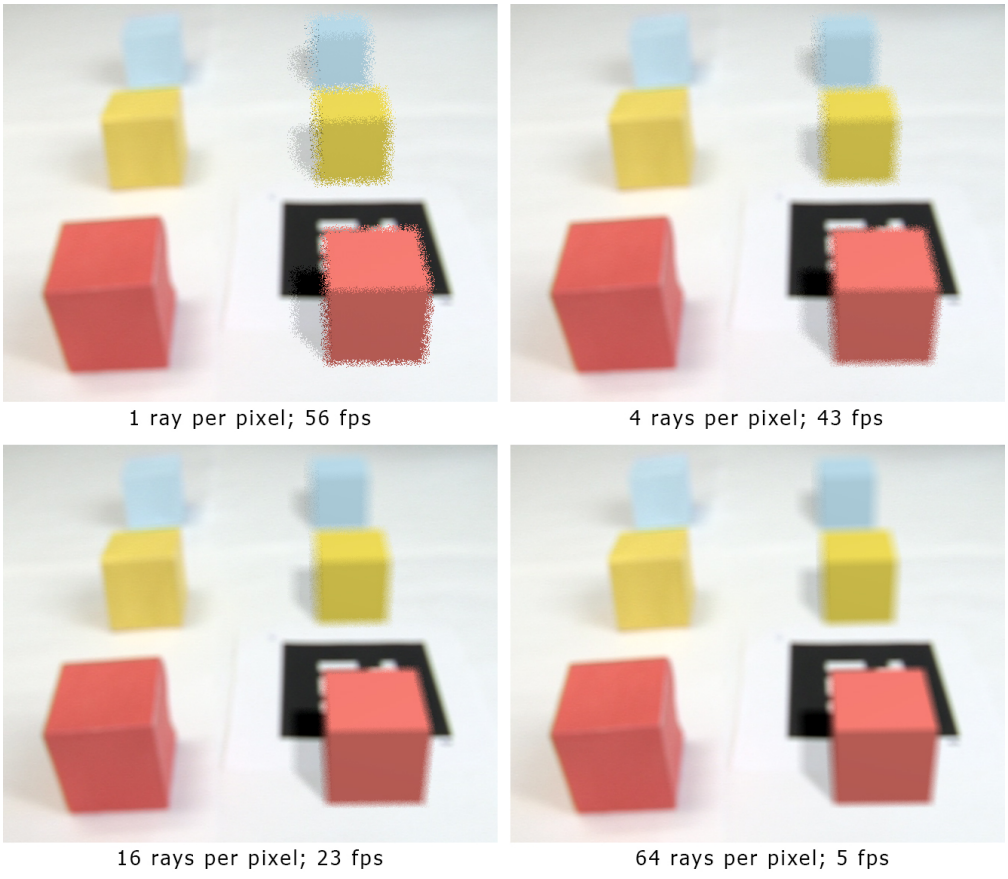

Figure 5.1: Effect of supersampling on DoF quality. More samples produce higher quality but also take more computational time. Left cubes in each image are real and right ones are virtual.

5.1.2 Specular Global Illumination

The speed of rendering with specular global illumination (Section 4.1) was measured at different sampling rates. Table 5.2 shows that our system is capable of calculating a high-quality result in interactive frame rates. The rendering time of different rendering steps is measured in milliseconds (ms). The speed of the following rendering steps was measured: photon emitting, Kd-tree build, and ray tracing with density estimation. In average, it took 20 ms to copy the texture to the GPU memory in our experiments. Our system can preserve interactivity even with complex scenes like the glass happy Buddha which consists of 855K triangles. Increasing the sampling rate improves the quality of the produced result, however it decreases rendering speed (Table 5.2), thus it offers a tradeoff between quality and speed.

Scene	**Tri.**	**Photons**	**Rpp**	$\mathbf{t}_p$	$\mathbf{t}_b$	$\mathbf{t}_r$	$\mathbf{fps}_r$	$\mathbf{fps}_n$
Metal Ring	9K	280K	1	45 ms	196 ms	21 ms	4	**20**
		280K	4	47 ms	196 ms	94 ms	3.3	**10**
		280K	9	47 ms	196 ms	206 ms	2.7	**5.7**
		No caustics	1	0 ms	0 ms	6 ms	-	**31**
Glass Monkey	15K	100K	1	20 ms	31 ms	24 ms	11	**20**
		1M	1	167 ms	619 ms	45 ms	1.6	**17**
		1M	4	167 ms	618 ms	138 ms	1.4	**6**
Glass Dragon	201K	240K	1	28 ms	47 ms	36 ms	10	**17**
		240K	9	29 ms	48 ms	269 ms	2.8	**3.6**
Glass Buddha	855K	240K	1	16 ms	17 ms	42 ms	7.5	**14**
		240K	4	16 ms	17 ms	165 ms	4.1	**6.5**

Table 5.2: Performance of our rendering system calculating specular global illumination. t_p is the duration of the photon shooting phase in ms, t_b is the time for Kd-tree building, and t_r is the time of ray tracing from the camera which includes the density estimation at hitpoints. fps_r is the frame rate with photon map rebuild enabled and fps_n is the frame rate with photon map rebuild disabled. Rpp represents the count of primary rays shot per pixel. All measurements were taken at the resolution of 720x576. The GeForce 590 GTX graphics card was used to render all measured scenes.

5.1.3 Diffuse Global Illumination

Diffuse GI is calculated in our rendering system by the differential irradiance caching algorithm. A performance evaluation of differential irradiance caching can be seen in Table 5.3. Our method provides interactivity even for complex scenes. The processing time of different rendering steps is displayed in Figure 5.2. The GPU ray tracing from the camera and rasterization based irradiance cache splatting are the fastest parts of our algorithm. These parts are running in real time even for complex scenes. The most computationally expensive part is the irradiance evaluation. The speed and quality of irradiance evaluation can be controlled by the number of ray samples which sample the hemisphere in Monte Carlo numerical integration. The interactivity of this step is achieved by a parallel GPU implementation in OptiX. The user can set the allowed inter-

polation error α to control the speed and rendering quality. This value determines the radius of interpolation around a cache record and the maximum allowed normal difference. A higher allowed interpolation error will lead to fewer cache records and then to higher performance. Other parameter, controlling the quality and performance, is the number of the splat buffer rectangular search regions in cache miss detection. Our method can be used both in real-time and offline scenarios resulting in different quality settings.

Scene	**Triangle count**	**fps**$_r$	**fps**$_m$	**fps**$_n$
Bunny	16K	19	10	**31**
Dragon	201K	14	7	**19**
Happy Buddha	886K	7	5	**8**

Table 5.3: Performance measurements of differential irradiance caching. The resolution of 720x576 and 1 ray per pixel was used. fps$_r$ denotes the frame rate with irradiance cache recalculation but new cache records are not created (miss detection is turned off). The fps$_m$ stands for frame rate with miss detection enabled, and fps$_n$ is the frame rate with no irradiance recalculation and disabled miss detection. 128 path samples were used per each cache record to evaluate differential irradiance.

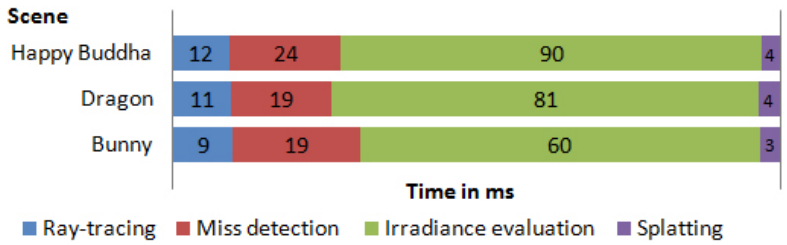

Figure 5.2: Duration of different rendering steps of differential irradiance caching.

Figures 5.3 and 5.4 show the dependence of rendering time on the triangle count and resolution. In Figure 5.3, the logarithmic dependence of rendering time on triangle count, caused by the nature of ray tracing, can be seen. The rendering time without irradiance cache rebuild has sublinear behavior caused by the combination of ray tracing and rasterization. In Figure 5.4, the sublinear dependence, caused by the parallelization of our algorithm, can be observed. The visible linear jumps are caused by exceeding the number of CUDA cores by the number of tasks which leads to additional processing.

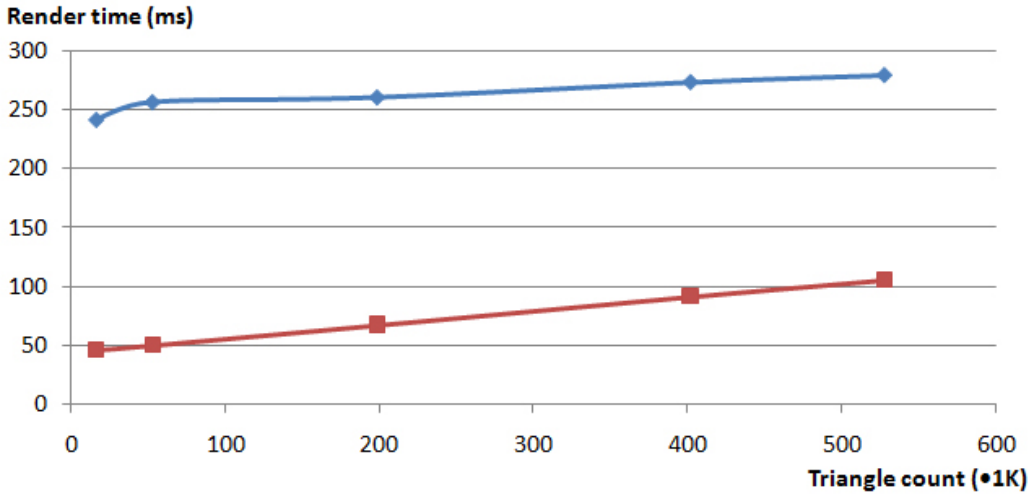

Figure 5.3: Dependence of rendering time on triangle count with differential irradiance caching. The red curve (bottom) plots the rendering with precomputed irradiance cache. The blue curve (top) shows rendering with irradiance cache recalculation. The triangle count is stated in thousands.

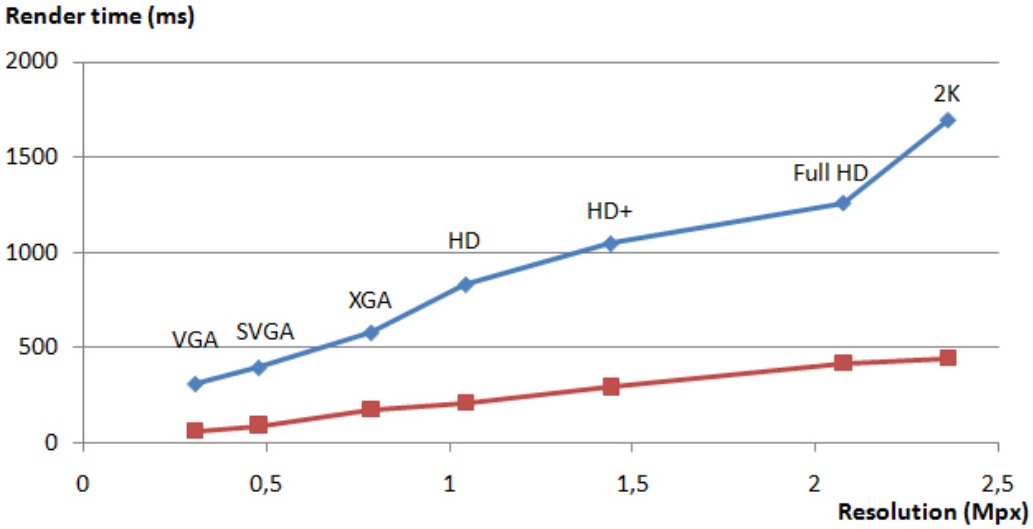

Figure 5.4: Dependence of rendering time on output resolution with differential irradiance caching. The red curve (bottom) represents rendering with precomputed irradiance cache and the blue curve (top) shows rendering with irradiance cache recalculation. The measurements were taken with a model consisting of 15K triangles. 256 sample rays were used to evaluate irradiance.

5.1.4 Differential Progressive Path Tracing

The quality and speed of the differential progressive path tracing can be assessed by comparison of the results produced with variable sampling rates. The rendering results of this algorithm, with different sampling rates, are shown in Figure 5.6. Each row displays a different scene. Frame rates are depicted below the images. The first column shows interactive rendering with 1 ray sample per pixel. The second column contains interactive results with 9 rays per pixel and the third column shows the converged rendering solution in the progressive refinement mode. The scene in the first row contains three real paper cubes on a table; and a virtual box with a metallic ball and a bunny (17K triangles). The image-based lighting approach was used to lit the scene. The scene in the second row contains real cubes on the left and virtual spheres on the right side (4K triangles). Light source estimation by image processing was used here. Note the very fast convergence when simplified lighting conditions are used (10 s). The third row shows a real scene with virtual Buddha statues (581K triangles). The image-based lighting approach was used. Interactivity can be achieved despite the very high complexity of this scene. The results show the interaction of light between virtual and real objects. Additionally, we analyzed the dependence of rendering speed on output resolution. The results of this dependence are depicted in Figure 5.5. Our path tracing based rendering is interactive even for full HD resolution. The results show that our system is well scalable both in terms of triangle count and output resolution. All tests were performed on a laptop with quad-core CPU and a GeForce GTX 680M graphics card. A resolution of 800x600 was used in all evaluations except Figure 5.5.

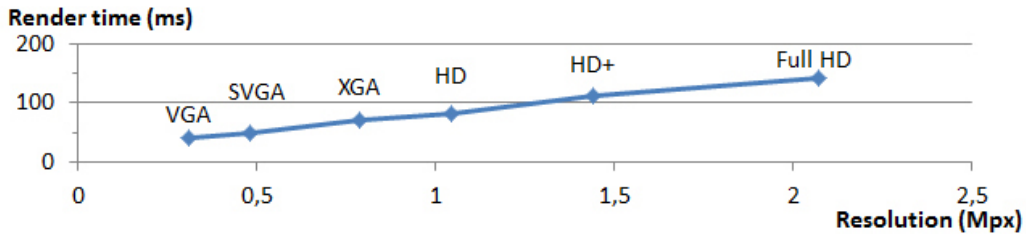

Figure 5.5: Dependence of rendering time on resolution with differential progressive path tracing. A 3D model, consisting of 20K, triangles was used. 1 ray per pixel was sampled.

5.2 Quality Comparison

The visual quality of our rendering results with varying parameters is discussed in this section. The figures, which show the variable quality levels of the developed methods with specific parameters, are displayed in the sections where these methods are presented. Additionally, a comparison to state of the art methods in rendering for AR is shown.

Depth of Field. The comparison of rendering with and without DoF effect is shown in Figure 3.7. This image highlights the importance of DoF for visual consistency of the composited image. Moreover, the results of DoF with variable sampling rates is displayed in Figure 5.1.

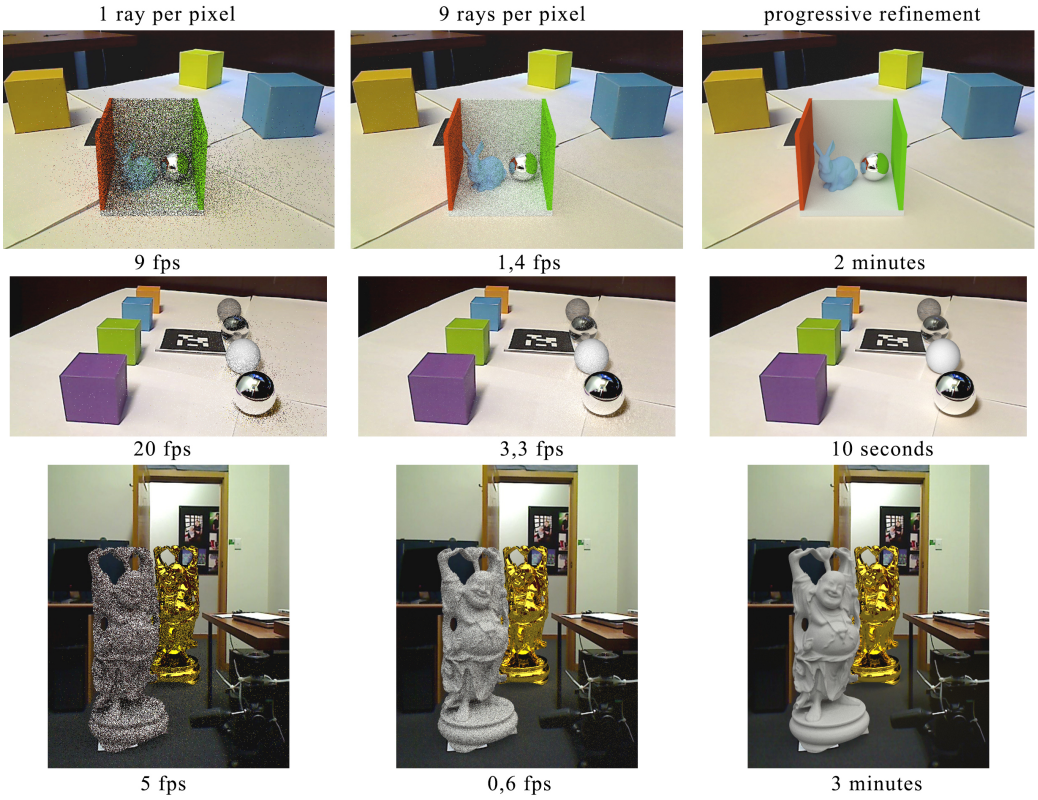

Figure 5.6: Rendering results with differential progressive path tracing. Variable sampling rates are used to asses the quality and speed of the rendering.

Global Illumination. The quality of the rendering with our global illumination methods can be seen in the following figures. Figure 4.5 shows the comparison of different rendering features, including antialiasing and specular GI. Figure 4.12 underlines the importance of the reprojection technique for accessing the splat buffer in irradiance cache splatting. This figure compares the rendering with and without reprojection. The rendering without reprojection suffers from the ghosting edges artifacts while the rendering with reprojection solves this problem. A comparison of the synthetic caustic and diffuse illumination to the real caustic and diffuse illumination can be seen in Figure 4.13. This figure shows consistent caustics, shadows and indirect illumination in our global illumination system.

Differential Progressive Path Tracing. The rendering with differential progressive path tracing with different sampling rates is displayed in Figure 4.15. This figure demonstrates the noise reduction when the sampling rate is increased. Additionally, the comparison of a rendering by differential progressive path tracing, with different sampling rates, can be seen in Figure 5.6. Furthermore, this figure shows the comparison of the preview mode to the converged solution in progressive refinement mode.

Comparison to Other Methods. Our real-time global illumination rendering system for AR was compared to two other rendering methods for AR. First, the comparison to differential instant radiosity [82] is shown. Our rendering system uses ray tracing and differential irradiance caching. Additionally, the differential progressive path tracing is included in this comparison. Second, the results of our rendering are compared to the delta voxel cone tracing [39].

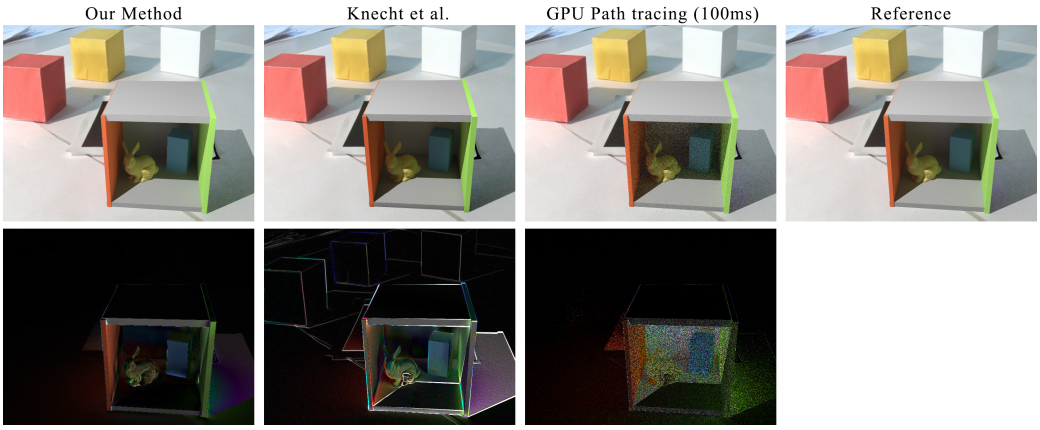

Figure 5.7: The comparison of rendering methods for AR. First row from left: Our method (10fps), differential instant radiosity (28fps), progressive path tracing (10fps), reference solution calculated by differential path tracing in 2 minutes. 4000 samples were used per cache record to evaluate differential irradiance in our method. 256 VPLs were used in differential instant radiosity (DIR) and antialiasing was enabled. The scene contains a virtual box including a bunny and a small box placed on a real table together with real cubes (16K triangles). Indirect illumination is visible especially in shadow areas inside the box. The second row shows the differences of results to the reference. All differences are multiplied by the factor of 5. The image rendered by differential instant radiosity is courtesy of Martin Knecht et al. [82].

In order to assess the quality and speed of our global illumination rendering with differential irradiance caching, we compare it to real-time progressive path tracing and to a differential instant radiosity solution [86]. The results of each method are compared to the reference solution by calculating per-pixel differences. The reference solution was calculated offline by differential path tracing on the GPU. The comparison of all three methods can be seen in Figures 5.7, and 5.8. Measurements were taken at PAL resolution of 720x576 on one Nvidia GeForce GTX 690.

We compare two different scenes. The same time was given to our method and progressive path tracing to render a frame (100ms in Figure 5.7 and 200ms in Figure 5.8). Differential instant radiosity (DIR) computes a solution with 256 virtual point lights (VPLs) in 35-40ms. We did not use more VPLs for DIR because the solution is already practically converged. The remaining error in DIR is due to a bias caused by the visibility approximations and clamping which is inherent in imperfect shadow maps [148], and a biased light-path distribution. The first scene (Figure 5.7) consists of real paper cubes on a table and a virtual box with a bunny and a small box inside. Our method introduces a small error due to interpolation between irradiance cache records. The error is rather visible on curved surfaces, like the bunny, where the

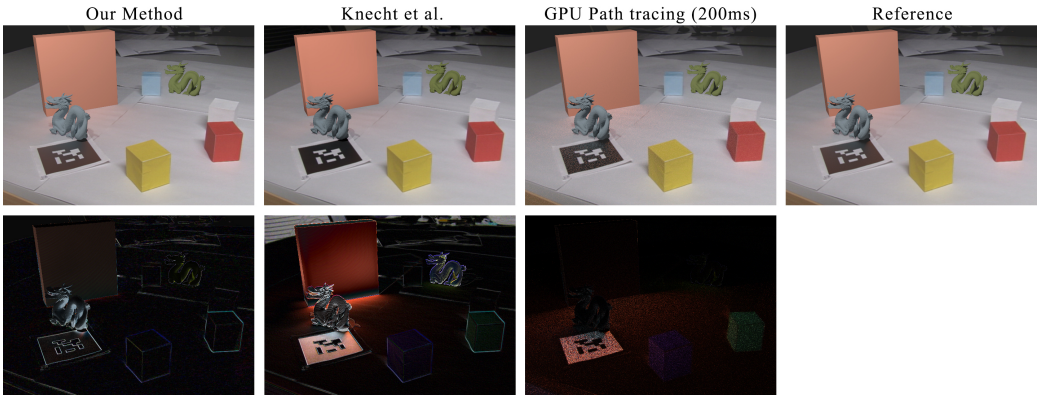

Figure 5.8: The rendering of a scene (402K triangles) with virtual dragons and a big box by different methods. In the first row from left: Our method (5fps), DIR (25fps), progressive path tracing (5fps), reference solution calculated by differential path tracing in 2 minutes. 1000 samples were used per cache record to evaluate differential irradiance in our method. 256 VPLs were used in DIR. The second row shows the differences of the results to the reference. All differences are multiplied by the factor of 5. Sharp edges in the background of the difference images are caused by a specific filtering in the reference solution and are not caused by the evaluated methods. The image, rendered by DIR, is courtesy of Martin Knecht et al. [82].

incorrect illumination is interpolated. The indirect shadow on the left wall of the box exhibits an error caused by missing geometric details in the visibility evaluation. This error is smooth and therefore not visually disturbing. In contrast to that, real-time progressive GPU path tracing introduces perceptually visible noise. DIR introduces a large error in shadow areas, where only indirect light is present, due to its light transport approximation. Moreover, the bright corners are caused by the visibility approximation in DIR. The second scene (Figure 5.8) contains real cubes on the table and two virtual dragons with a big virtual box. Color bleeding from virtual objects onto real geometry can be seen. The results show that our method is close to the reference solution. Our solution introduces a smooth error in shadowed areas on the virtual dragon, where only indirect illumination is present. This error is caused by interpolation between irradiance cache records on the curved surfaces of the model. Differential instant radiosity suffers from numerical errors especially near the corners of the box. The results show that our method can calculate high-quality global light transport between virtual and real objects at interactive frame rates. Our results outperform fast methods in terms of quality and predictive solutions in terms of speed, while achieving quality close to the reference. The limitation of differential irradiance caching is bias in form of interpolation between cache records.

In the second comparison, our global illumination rendering is compared to the delta voxel cone tracing [39]. The results of this comparison can be seen in Figure 5.9. As the rendering times in this figure suggest, delta voxel cone tracing is faster than our system. However, our rendering outperforms the voxel cone tracing in terms of quality. The visual artifacts caused by voxel approximation of the geometry, in reflections calculation, can be seen in Figure 5.9 top left. Opposite to that, our method calculates high-quality reflections by the ray tracing.

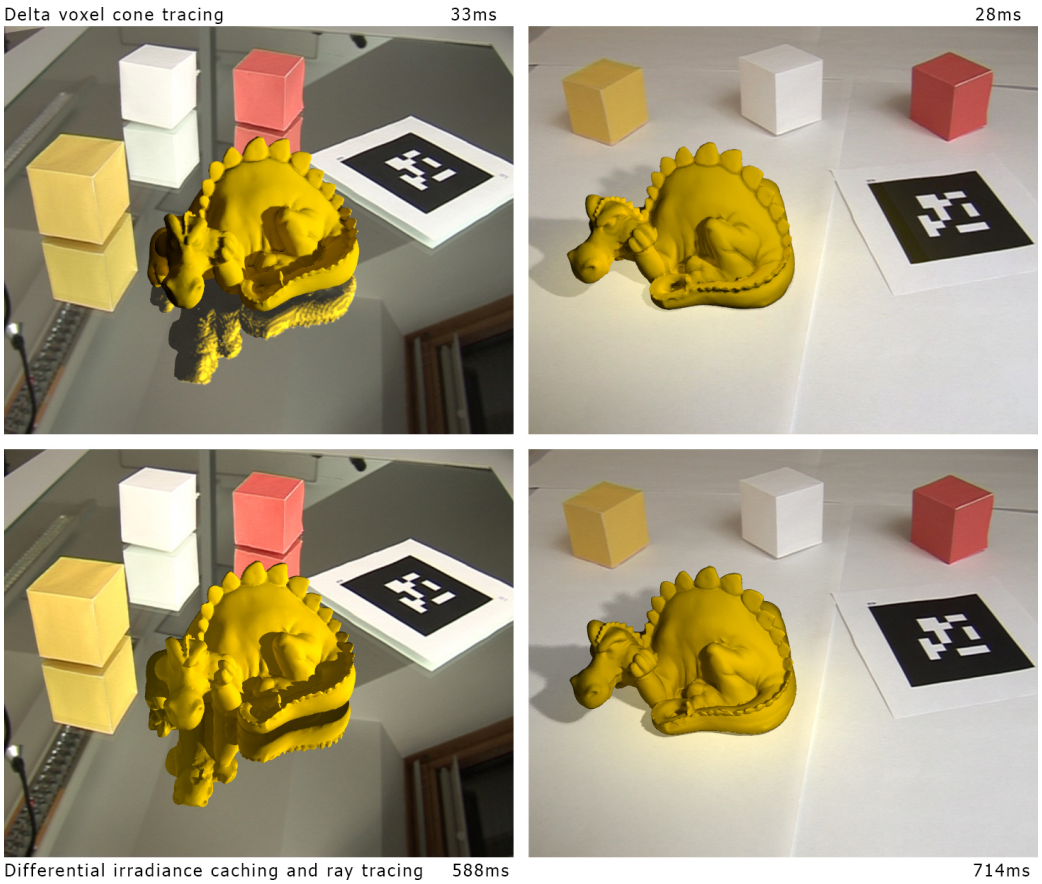

Figure 5.9: The comparison of differential irradiance caching (bottom row) with delta voxel cone tracing (top row). The virtual dragon is rendered on two types of real surfaces: a mirror and a diffuse table. Our method provides higher quality of the final result, especially in the scene with the mirror. The voxelization artifacts are visible in the reflections rendered by delta voxel cone tracing. Our system calculates the result in lower speed. This slow down is caused by the high sampling rate in differential irradiance evaluation. The rendering times in milliseconds are indicated near the images. The images rendered by delta voxel cone tracing are courtesy of Tobias Franke [39].

5.3 Study of Visual Realism

Visual realism and coherence are important aspects of AR. The difficulty of measuring visual realism, perceived by the user, is that it is subjective. In order to study visual realism in AR with our global illumination methods, we designed and conducted several user studies. The following sections describe the studies of visual realism with the depth of field effect and specular global illumination in AR. The additional study of visual realism, related to the sense of presence in AR, is described in Section 5.4.

5.3.1 Depth of Field

In order to study the visual realism in AR with and without DoF, we conducted a preliminary user study. We evaluated the users' perception when watching an AR video, rendered with DoF, in comparison to a video without DoF. Our hypothesis was that visual realism in AR is higher in a rendering with depth of field than without it. Two recorded videos were shown to each user: one video with, the other without a DoF effect. Users were asked to tell which video looks more realistic. Thirty people (computer scientists as well as people from other professions) participated in this study. Fifteen of them were men and fifteen were women. 90% of participants considered the video with DoF more realistic. The result of this preliminary study suggest that DoF is an important feature for visual realism in AR. Additionally, we studied the visual realism with the depth of field effect and global illumination in a more complex user study, described in the next section.

5.3.2 Specular Global Illumination

In order to study the impact of our global illumination rendering on visual realism in AR, we evaluated users' perception by quantitative methods. Visual realism was evaluated in two stages. Additionally, we asked participants to identify which objects in the shown video are real and which are virtual. The intention of this question was to investigate if the visual quality and coherence is good enough to make people believe that AR content is real. This study contains the following hypotheses:

H1: Realistic features of ray tracing based rendering have a positive impact on the user's perception of visual realism in augmented reality.

H2: Every realistic rendering effect, used in our system, positively influences the realistic appearance of the composited video.

Study Design. The conducted user study consisted of the two main stages. In the first stage, a recorded AR video, rendered by our system, was shown to the user. This video contains diffuse real cubes, diffuse virtual cubes, a refractive virtual glass sphere and a reflective virtual ring. The following effects were enabled in this video: Refraction/reflection, antialiasing, caustics, and DoF. After watching the video, users were asked to rate how realistic the video was on a linear visual analogue scale from -3 to 3. The value of -3 means that the video is not realistic at all and the value of 3 means that the video looks completely realistic. Additionally, users were asked to indicate which objects in the video are real and which are virtual. The first stage of the user study was used to assess hypothesis H1. In the second stage of the user study, we used sequences consisting of five videos each. The same scene was shown in each clip within the sequence. In each successive video in a sequence a new rendering feature was added. Figure 4.5 demonstrates an example of enabling different effects one after another. In order to make the study independent of the scene layout and scene content, two different scenes were used. In the first scene, a virtual refractive glass sphere on a real table with real diffuse cubes was rendered. In the second scene, a virtual metal ring and diffuse real cubes were shown. Each video, including the first general video, was played to the users only once. Two sequences of videos were shown

to every user to avoid the dependence of answers on the order of effects and rendered scenes. Each sequence introduced the effects in a different order. Therefore, the order was randomized for all participants in a standardized way.

In total, four different sequences were created and two of them were selected for every user in the second stage of the user study. This selection randomizes sequences to avoid a bias in results. In the first sequence of videos the realistic effects were added in the following order: no effect, refraction/reflection, antialiasing, caustics, and depth of field. The virtual refractive sphere was rendered in this sequence. In the second sequence the effects were added in a different order: no effect, caustics, refraction/reflection, antialiasing, and depth of field. The virtual metal ring was rendered in this sequence. In a third sequence the effects were added in the same order as in sequence 1, however the metal ring scene was used. Finally, in the fourth sequence the effects were added in the same order as in sequence 2 with a refractive glass sphere used.

Users were asked to compare each pair of successive videos in the sequence. They answered the question: Which video in the pair (previous, current) looks more realistic? Users evaluated the increase of realism in successive videos by using a linear visual analogue scale in range from -3 to 3. The value -3 means that the previous video was much more realistic. 0 has the meaning of a similar, comparable amount of realism in both videos. And 3 means that the current video is much more realistic. The difference between two successive clips was evaluated because we wanted users to concentrate on a desired rendering effect and to judge the increase of realism when this effect is enabled. The second stage of the user study was used to evaluate hypothesis H2.

Results. 43 people participated in our study. 12 of them were men and 31 were women. 26% of the participants had previous knowledge in computer graphics and 74% did not. The general video, with all effects, was shown to all participants. The first and second video sequences were shown to 21 users; the third and fourth sequences were shown to the remaining 22 users.

In evaluation of first hypothesis, we investigated perception of realism in the overall video. The results can be seen in Figure 5.10. The average value of 0.8 suggests that the scene was perceived as realistic and this result supports our first hypothesis. Additionally, we analyzed the perception of realism separately for selected groups within our participants. We analyzed the results depending on gender and computer graphics experience. The results are shown in Figure 5.10. Men with CG knowledge report higher values of realism.

In addition, the participants were asked to judge which objects in the overall video are real and which are virtual. In average 40.1% of the virtual objects were mistakenly marked as real ones. The virtual metal ring which reflects the real environment and casts caustics has the most realistic appearance according to this evaluation. 53.5% of participants marked this virtual object as a real one. It is important to note that in 36% of all cases the users mistakenly thought that real objects were virtual ones.

In the second hypothesis, we evaluated the increase of realism caused by enabling specific effects. Figure 5.11 shows the results of comparison of all effects. The addition of each effect contributed positively to the perception of realism in AR. The results indicate that antialiasing has the highest contribution to visual realism. It reduces disturbing alias on the edges of virtual objects, making them appear more natural. Realistic refraction and reflection is also an impor-

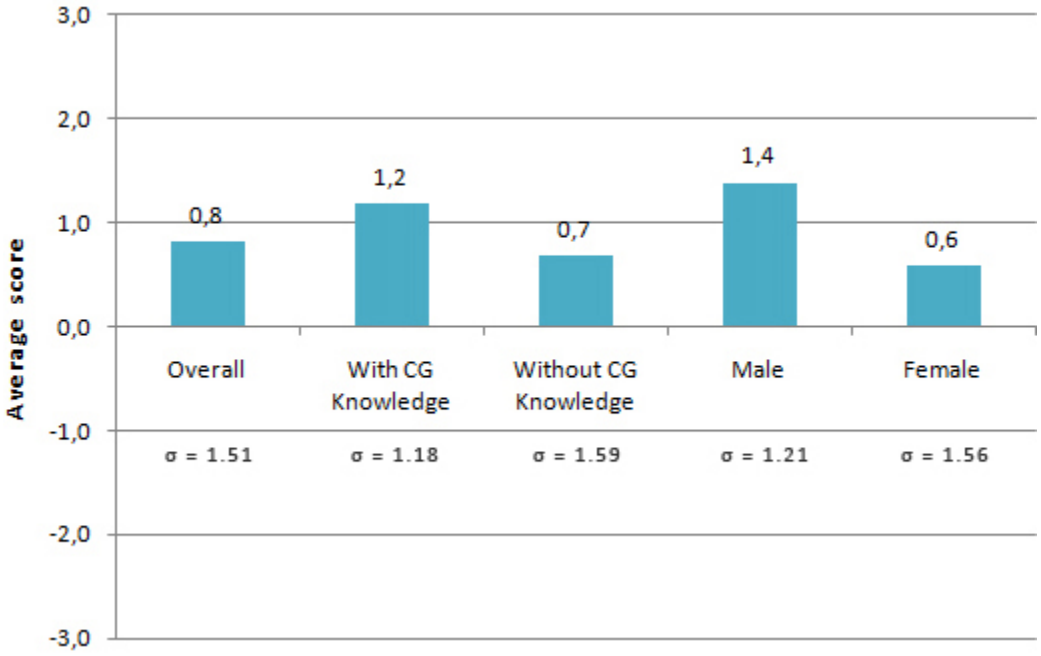

Figure 5.10: Average results of perceived realism of the overall AR video. σ = std. dev.

tant feature which enhances realism. The caustics were less important for users, but they also increased the amount of realism in AR. However, the result with caustics could be influenced by the fact that in the second sequence caustics were added to an object before reflection/refraction. This made caustics look unnatural because the respective objects were rendered just semitransparently. In summary, our second hypothesis was supported because the results indicate that all added rendering features increase the amount of realism in AR. Influence of virtual content on realism, perceived by the users in video sequences 1 and 3, can be seen in Figure 5.12.

In summary, our results demonstrate that each developed rendering effect has a positive impact on visual realism and coherence in AR. The rendering results of different effects can be seen in Figure 4.5. Finally, the user study suggests that the ray tracing based rendering has a positive effect on visual coherence in AR.

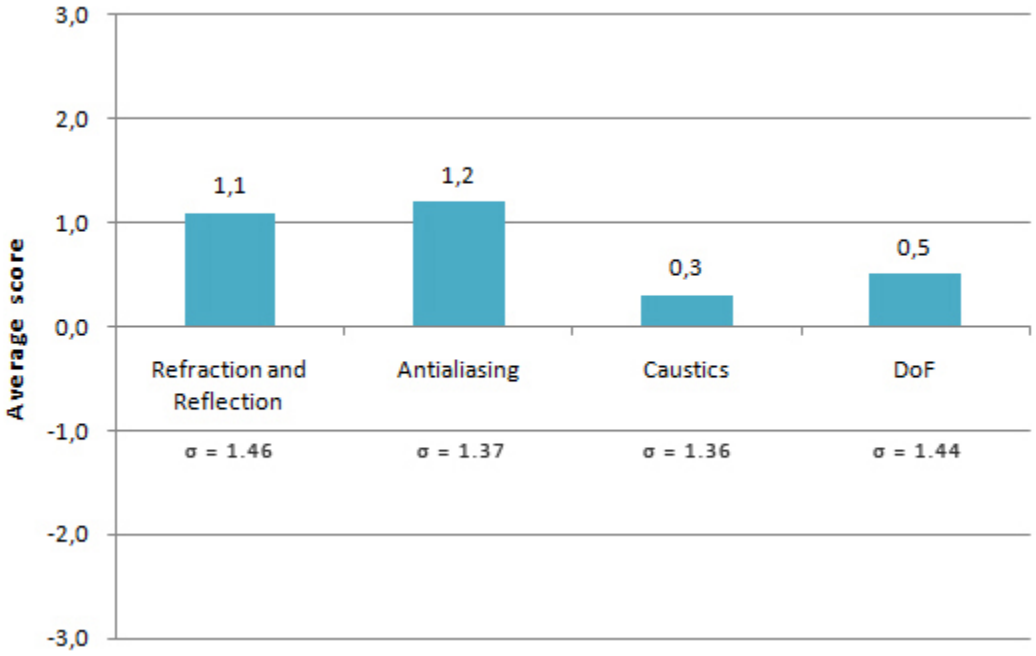

Figure 5.11: The influence of rendering effects on perceived realism in AR. σ = std. dev.

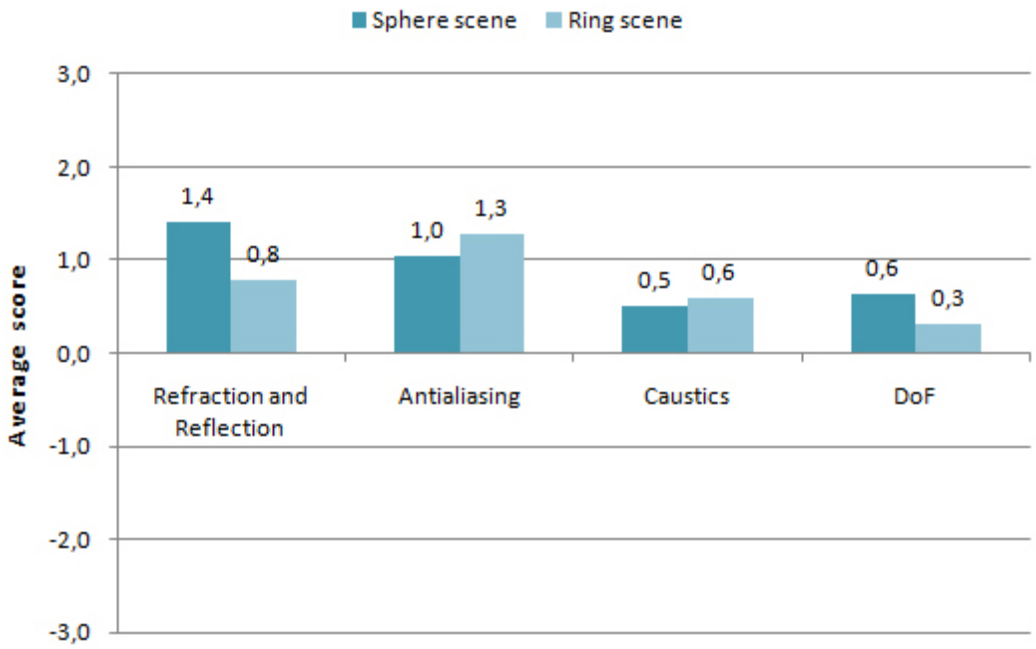

Figure 5.12: Difference of perceived realism with different virtual content.

5.4 Presence Study

Presence has been studied in previous research mainly in virtual reality but less so in AR. Studying presence in AR can help to better understand the perception of virtual information and to increase the immersive user experience with applications. Although there is previous research of presence in VR, there are still open questions about the sense of presence and the factors which influence it especially in AR. The dependence between the illumination model, used in rendering, and the perceived presence is important for the future development of AR applications.

In order to study the effect of lighting on the sense of presence in AR we designed and conducted a user study. We compared presence ratings between global illumination rendering and direct illumination and asked study participants to judge which of the shown objects are virtual and which ones are real. Thirty people participated in a within-group experiment. A set of questionnaires was used to measure the sense of presence, perception of realism, and the rate of interpretation of virtual objects as real ones, with both global and direct illumination conditions. This section describes the design, methods, and the results of the study. We discuss the major differences between real and virtual objects, as observed by the users, and the categories of important features for visual realism in AR.

5.4.1 Illumination Models

Two illumination models were examined in our experiment, direct illumination and global illumination in interactive AR. Our one-pass compositing algorithm was used to composite virtual objects into the real-time video.

Global Illumination. Our global illumination system for AR was used to render the objects in this study. We simulate direct and indirect illumination separately. Finally, we combine them to calculate the final radiance value for each pixel and display it. Direct illumination is calculated by real-time ray tracing on the GPU and indirect illumination is calculated by differential irradiance caching presented in Section 4.2.1.

Direct Illumination. Direct illumination accounts for the light that is emitted from the light source and reflected from the surfaces directly towards the observer (camera). Shadows are included in the result of the direct illumination. However, no indirect light reflections are included in this solution. We used GPU ray tracing to calculate the direct illumination in an interactive AR setup. The difference between global and direct illumination, used in our experiment, can be seen in Figure 5.13.

5.4.2 Methods

The main aim of our presence study was to explore the relation between the illumination models and the sense of presence in AR. The study was designed as a within-group experiment which uses questionnaires to measure the sense of presence and realism perceived by the user.

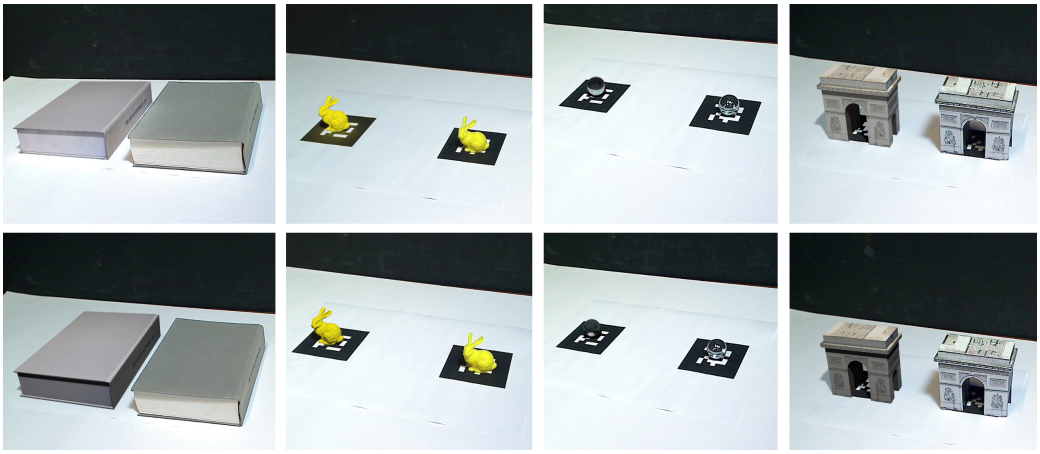

Figure 5.13: (Top row) Global illumination rendering in AR. (Bottom row) Direct illumination rendering in AR. Objects from left: Book, bunny, glass sphere, Arc de Triomphe. The left object is virtual and the right one is real in all images.

Hypotheses. We posed three hypotheses to examine the effect of lighting in AR on presence:

H1: The sense of presence is higher with global illumination rendering than with direct illumination.

H2: The number of virtual objects mistakenly judged as real ones is higher with global illumination rendering than with direct illumination.

H3: There is a positive correlation between the sense of presence and the perceived realism of virtual objects.

Experimental Design. Two different experimental setups were used to verify the hypotheses:

1. Observation of videos recorded from AR.
2. A real-time AR setup.

The first setup in our study is based on videos, recorded from interactive AR. This setup was designed to test the hypothesis H2. Two AR scenes were created. Both scenes contain 4 virtual and 3 real objects. Both scenes were recorded with global illumination as well as with direct illumination. All four videos were shown to the participants in a random order. They could watch each video repeatedly and were allowed to pause the video. After each observation, the participants were asked to identify which objects were real and which were virtual. The recorded videos were used in this task instead of interactive AR to minimize the bias, caused by recognition of virtual objects due to tracking imperfections.

The second setup was the real-time AR scenario. In this setup, the users could freely observe the virtual and real objects in a video-see-through system. The real scene was captured

by a single camera and displayed on a head mounted display (HMD) (Sony HMZ-T1 HMD; resolution 800x600 per eye). The camera was mounted on the HMD. Either direct or global illumination was calculated to render the virtual objects in the AR scenes. In global illumination, the interreflections between virtual and real objects were included.

An optical marker-based ARToolKitPlus [185] tracking system was used to calculate the position and orientation of the real camera. The scene consisted of two markers with a virtual object positioned on one of them. A real object, similar to the virtual one, was positioned onto the second marker. Users were informed which object was real and which was virtual. Participants could freely walk and observe the objects from different viewpoints as long as they wished. A participant and the corresponding AR view, during the experiment, can be seen in Figure 5.14.

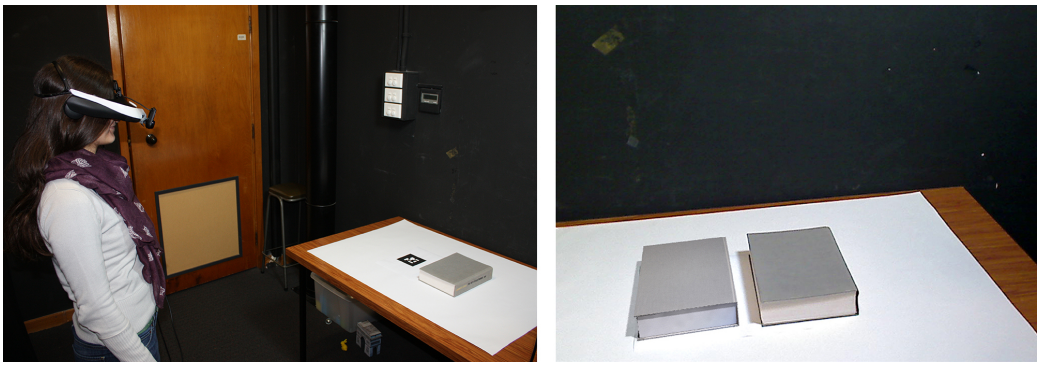

Figure 5.14: (Left) The participant, wearing an HMD, is observing the AR scene. (Right) The view of the participant shows a virtual book (left) and a real book (right).

Each participant observed eight trials. After each observation, the users filled in a presence questionnaire (Table 5.4) which consists of specific AR presence questions designed by Regenbrecht et. al [143, 144]. Moreover, we added a question for comparing virtual and real objects (Question 7). All questions were answered on 7 point Likert type items (1 strongly disagree, 7 strongly agree).

1.	Watching the virtual object was just as natural as watching the real world.
2.	The virtual object and I were in the same environment. (I felt I could have touched the virtual object)
3.	Virtual and real environments formed one, common space.
4.	I perceived virtual element as being only a computerized image, not as a real object.
5.	The virtual element seemed real for me.
6.	I could not distinguish between real object and virtual object.
7.	The virtual object looked visually the same as its real counterpart.

Table 5.4: Presence questionnaire. Questions 1-6 were developed by Regenbrecht et al. [143].

The participants observed four virtual objects with each illumination condition. The order of the conditions and objects displayed was randomized to minimize a positioning effect. The following virtual objects (Figure 5.13) were observed: Book (1278 triangles), Stanford bunny (69K triangles), glass sphere (962 triangles), and textured building Arc de Triomphe (346 triangles). After the participants had observed the objects with a respective illumination model, they completed an additional questionnaire measuring the overall presence and perceived realism with this illumination model (Table 5.5) [143]. The Likert scale ranging from 1 to 7 was used in this questionnaire (1 - very low, 7 - very high).

1.	Overall, how would you rate the sense of presence generated by the virtual elements; to what extent "they were here"?
2.	Overall, how would you rate the degree of realness achieved by the virtual elements; to what extent "they seemed real to you"?

Table 5.5: Overall questionnaire, related to sense of presence and perception of realism. Questions were created by Regenbrecht et al. [143].

At the end of the experiment we asked users to tell what they perceived as the major differences between virtual and real objects in order to get a better understanding of the importance of different visual features for coherence in AR.

Participants. Thirty people, aged between 19 and 42 years, participated in our experiment. All had normal or corrected-to-normal vision. The participants provided informed consent and ethical approval was obtained from the University of Canterbury Human Ethics Committee. The group of participants consisted of 14 women and 16 men. Nine had previous experience with AR and eight had knowledge in computer graphics.

Data Analysis. After recording answers to the presence questionnaire, Cronbach's alpha was calculated to analyze internal consistency of this questionnaire. Values 1-7 from question 4 were recoded to 7-1 because of the negative nature of this question. Then, the questions were merged together and a mean was calculated. A Wilcoxon signed-rank test was used to test the difference in answers between global and direct illumination conditions.

For the trials in which the participants were asked to judge which objects were real and which were virtual, the number of virtual objects, mistakenly marked as real ones, was counted for both global and direct illumination. In order to test for difference between the two illumination models we again used a Wilcoxon signed-rank test. The decision of using a non-parametric test was made because the data did not follow the normal distribution. Hypothesis 3 was tested by calculating Pearson's correlation for responses in the overall questionnaire.

5.4.3 Results

Cronbach's alpha showed high internal consistency of the presence questionnaire ($\alpha = .938$). We merged the questions using the mean for analyzing the questionnaire results. The difference was calculated by subtracting the mean presence questionnaire result for direct illumination

from the mean result for global illumination. For the majority of observations, the difference is positive which indicates that the sense of presence was rated higher for global illumination than for direct illumination.

Taken all scenes together the Wilcoxon signed-rank test showed that the sense of presence is significantly higher with global illumination than with direct illumination ($Z = -4.043, p < .001$)[1]. However, a more detailed analysis showed that this result is scene or object dependent. We found significant differences in presence for the book scene and the Arc de Triomphe scene. The remaining two scenes showed no significant difference (see Table 5.6). Figure 5.15 shows the means of presence for global and direct illumination.

	Book	**Bunny**	**Arc de Triomphe**	**Glass sphere**	**All scenes**
Z	-2.883	-1.574	-2.888	-0.649	-4.043
p	**.004**	**.116**	**.004**	**.516**	**< .001**

Table 5.6: Results of Wilcoxon signed-rank test on data from presence questionnaire.

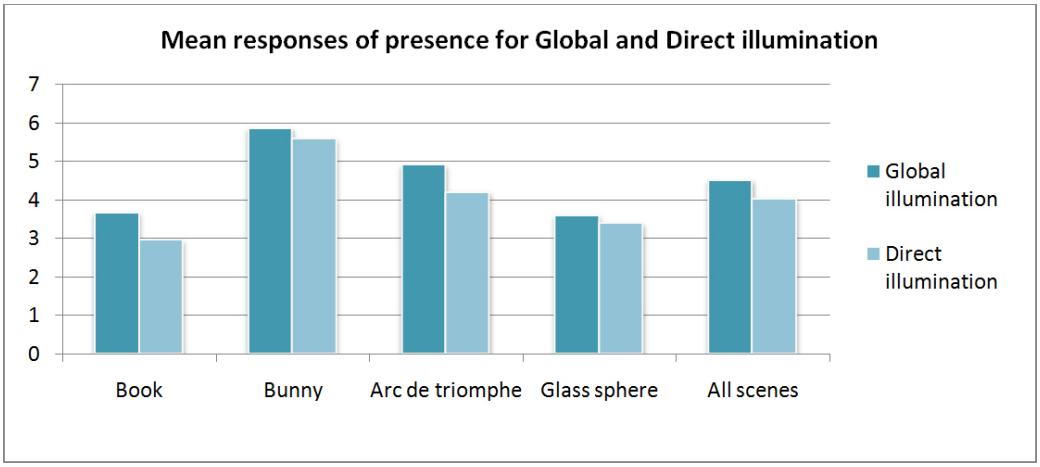

Figure 5.15: Mean of presence questionnaire responses for global and direct illumination.

The number of virtual objects, mistakenly judged as real ones, was counted in each video, watched by the participants, in the video observation task and summed for the two global illumination videos and the two direct illumination videos. Our assumption in H2 was that the number of virtual objects mistakenly marked as real ones will be higher with global illumination rendering. We tested the hypothesis H2 by the Wilcoxon signed-rank test. Results showed that the number of virtual objects mistakenly marked as real ones is significantly higher with global illumination than with direct illumination ($Z = -2.207, p = .027$) (Figure 5.16).

The overall questionnaire (Table 5.5) was completed after observations of each global and direct illumination condition. This questionnaire was used to verify H3. In H3, we assume a positive correlation between the sense of presence and the realism of virtual objects perceived by the

[1]To minimize the risk of type I error we use a Bonferroni adjusted p-value for interpreting significance of p =.01

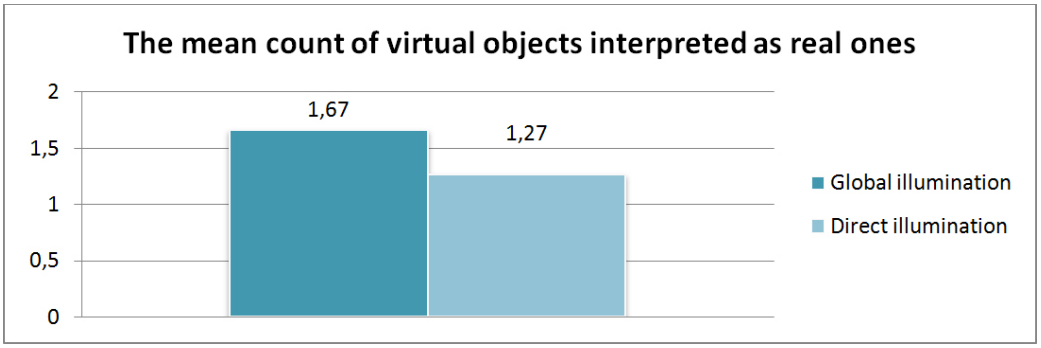

Figure 5.16: The number of virtual objects interpreted as real ones for global and direct illumination conditions.

users. A Pearson product-moment correlation showed a significant, strong, positive correlation between the sense of presence and realism perceived by the users ($r = .717, n = 60, p < .001$).

In addition to the questionnaires, we asked the participants what they perceived as the major differences between the appearance of virtual and real objects. The answers were processed by axial coding. We focused on finding the most important features of rendering which can contribute to visual realism. The following codes were extracted from the text ordered by the frequency of occurrence starting by the most frequent:

1. Shadows
2. Colors
3. Textures
4. HDR reflections on glass
5. Tracking precision
6. Sharpness of edges
7. Camera distortions

The four categories in which the mentioned items fit are:

- Lighting
- Materials
- Camera
- Tracking

According to the frequency of occurrence the most frequent aspects are shadows. Inconsistency in shadows is obvious especially in the case when real shadows are present for comparison. In addition, material properties and proper simulation of camera distortion turned out to play a significant role. Precise camera pose tracking is important for the spatial coherence between virtual and real scenes.

5.5 Discussion

The results of the evaluations show that the methods, presented in this thesis, are suitable for high-quality AR. The presented rendering methods can render the AR scene in interactive to real-time frame rates while achieving a high quality of the composited image. Additionally, the results of our user studies show a significant positive effect of our rendering algorithms to the sense of presence and perception of realism in AR.

Rendering Speed. Section 5.1 provides a detailed overview of the rendering times for particular algorithms. The interactivity, required by the nature of AR, is achieved in our results. Moreover, different steps of rendering with differential irradiance caching are analyzed. This analysis gives an overview of the computational load of particular rendering steps. Generally, a tradeoff between quality and speed can be achieved with the presented rendering algorithms, leading to the adaptability to different scenarios including fast interactive applications, production rendering, and unbiased previews.

Rendering Quality. The rendering quality of the final image, composited by our system, is evaluated with different sampling rates. The visual comparison of rendering with and without our methods are presented in the particular sections which describe these methods. Additionally, the results of differential irradiance caching are compared to the other GI methods for AR. This comparison shows that our method outperforms the other methods in terms of quality. Therefore, the algorithms presented in this thesis can be used in AR applications to produce a high-quality final result and to increase visual coherence between virtual and real objects.

Visual Realism and the Sense of Presence. Several user studies were conducted during this PhD research. We studied visual realism and the sense of presence, perceived by users, to provide better insight into human perception of AR. First, we focused on visual realism of different visual features in AR rendering (Section 5.3). The increase of visual realism, caused by adding the following effects, was studied: depth of field, reflection and refraction, antialiasing, and caustics. Our results suggest that all studied effects have a positive impact on visual realism in AR. This observation leads to a guideline which suggests to use these effects for AR applications to increase visual coherence. Second, we focused on the study of the sense of presence in AR (Section 5.4). Our main goal was to investigate if global illumination has a positive impact on the sense of presence. Additionally, the correlation between the sense of presence and visual realism was evaluated. We also studied if the ability of making people believe that virtual objects are the real ones is higher with global illumination than with direct illumination.

The results of our presence study suggest that overall the sense of presence is higher with global illumination in comparison to direct illumination. However, a separate analysis of different AR scenes shows that only in two out of the four scenes this effect is significant. Therefore, hypothesis H1 of our presence study was only partially supported. We assume that the difference between rendering of global and direct illumination was small in the bunny and glass sphere scenes, causing low significance of difference in presence in these scenes. Furthermore, we show that people perceive virtual objects as real objects to a higher degree with global illumination than with direct illumination. Hypothesis H2 was supported by the analysis of video observation data. The number of virtual objects marked as real ones was significantly higher with global illumination than with direct illumination. With Hypothesis H3 we assumed that the realism of virtual objects perceived by users correlates with the sense of presence. This hypothesis was confirmed and we found evidence of a significant, strong, positive correlation between the sense of presence and realism perceived by the users.

The technical setup used in this experiment poses some possible limitations. There were several technical issues that could affect the quality of the gathered data such as the precision of camera tracking. This can be improved in future work to ensure correct spatial registration. Also, the single camera setup can be extended to stereo cameras to enable a 3D AR scenario. Eight participants had previous knowledge in computer graphics. These people could possibly tend to prefer global illumination based on their knowledge. The results of our experiment suggest that the illumination model used in AR rendering has a significant effect on presence and realism in some scenes. This finding poses an important guideline for future research and development in the field of AR which recommends to use global illumination in AR applications.

5.6 Application Areas

The algorithms, presented in this thesis, can be used in many application areas which benefit from AR to increase visual coherence. Computer graphics and particularly augmented reality is helpful for solving different kinds of problems. The possible application areas which can profit from our results include the following:

- **Automotive design** - The high-quality rendering plays an important role in automotive design. It allows designers to see the visual aspects of the designed model before the production. Augmented reality can help to improve the process of design and visualization in the automotive industry. The natural user interaction, available in AR, can help to better create, modify, or view the designed vehicle model. Furthermore, the designed car can be visualized in the natural environment with consistent lighting. Thus, our rendering algorithms for AR may enhance the productivity, creativity, and efficiency in the design process.

- **Architectural design** - Architectural visualizations, created by high-quality rendering, are often used to show the designed buildings before construction. Additionally, interactive high-quality rendering can be useful in the design process. AR can help to position the designed virtual buildings directly on the construction site. Moreover, the real-time global illumination calculation accounts for correct lighting of buildings and the surrounding

environment. This feature can be used to visualize the buildings at different day times or to calculate the light intensity at certain points of the model.

- **Medical therapy and rehabilitation** - Previous research showed that augmented and virtual reality can be efficiently used in therapy and rehabilitation. An interactive AR exposure treatment system for the fear of spiders was developed by Davies et al. [23]. In their system, a virtual spider is inserted into the real world through AR. A patient can interact with the spider. The treatment is done by exposing the patient to the spider. This exposure is controlled by the therapist. The high-quality rendering algorithms developed in this PhD thesis can help to improve the efficiency of treatment, because virtual objects will become more believable. Another research work, presented by Hoffman et al. [66], shows the possibility of using virtual reality in pain reduction. Their results show that burn patients report 35-50% reductions in procedural pain while in a distracting immersive virtual reality. Additionally, fMRI brain scans showed associated reductions in pain related brain activity during VR. We believe that similar results can be achieved in AR and that the visual realism, provided by our high-quality rendering algorithms, can increase the amount of pain reduction.

- **Education and training** - AR applications for education and training can enhance the imagination of students, the attractivity of educative information, and the understandability of the explanations. Real-time high-quality rendering in AR can help to boost these advantages and create visually pleasing educational materials.

- **Entertainment** - Augmented reality games become popular due to their interaction of real and virtual worlds. Visual coherence between virtual and real objects in these games can improve their attractiveness and game experience. The majority of AR games run on mobile devices, while these do not provide the required computational power to run the ray tracing based algorithms in real time. This limitation can be overcome by performing the rendering on the server and streaming the data back to the mobile device. We believe that the future development of mobile GPUs will allow ray tracing based rendering in real time on mobile devices.

- **Movie production** - Movie production is the area which highly benefits from high-quality rendering, indistinguishable from reality. AR can be used on set while shooting the film to see a real-time preview of the final compositing of virtual and real worlds. Moreover, our differential progressive path tracing allows for a progressive convergence to an unbiased rendering solution in AR, which can be used to see the compositing result in the final quality.

CHAPTER 6

Conclusion

This thesis has presented novel methods for high-quality rendering in augmented reality. The problems of visual coherence, interactivity, light source estimation, and user perception were approached by this PhD research. We developed algorithms for high-quality rendering of numerous visual features in AR. The results show that our algorithms are suitable for AR and that they outperform the state of the art methods in terms of quality. Furthermore, the results of our studies suggest that our global illumination rendering has a positive effect on perception of realism and the sense of presence in AR. The following section summarizes the particular contributions and findings.

6.1 Summary

One-Pass Differential Rendering in Ray Tracing. The novel one-pass differential rendering algorithm in ray tracing, presented in this thesis, helps to improve the efficiency of rendering in AR. The previously used differential rendering algorithm requires two solutions of light transport for correct compositing. Our algorithm calculates both of these light transport solutions together. Therefore, the efficiency of the rendering in AR by ray tracing is increased.

Physically Based Depth of Field in Augmented Reality. An accurate camera simulation, including a depth of field effect, is an important feature in rendering and particularly in AR. This thesis proposed the first algorithm for physically based DoF calculation in interactive AR. We use the camera model with finite sized aperture and sample the aperture area in the ray tracing pipeline. Our results show that a physically accurate depth of field can increase the visual coherence between virtual and real objects.

Specular Global Illumination in Augmented Reality. We presented the method for interactive rendering in AR which can simulate specular global illumination effects including refraction, reflection, and caustics. Our method is based on GPU ray tracing and photon mapping;

and it is capable of producing believable interactive augmentations. The correct material of refractive objects is simulated by employing the Fresnel reflection. Additionally, we developed an interactive implementation of photon mapping which calculates caustics in high quality. Finally, our user study shows that the presented algorithms have a positive impact on visual realism perceived by the users.

Diffuse Global Illumination in Augmented Reality. A novel method for high-quality diffuse indirect illumination calculation in AR scenes, running at interactive frame rates, was presented. Our method outperforms state of the art real-time methods in terms of quality and physically based methods in terms of speed. We introduced a parallel differential irradiance calculation at irradiance cache records which runs on the GPU and uses Monte Carlo integration. Additionally, differential irradiance splatting with data addressed by a reprojection technique was presented. This new way of accessing data in splatting approaches is useful for visually plausible depth of field, refraction, and reflection of splatted data. In conclusion, the differential irradiance caching turned out to be an efficient algorithm for accurate global illumination calculation in augmented reality.

Global Illumination System for Augmented Reality. The presented algorithms were incorporated into the complex system for high-quality rendering and compositing in AR. This system is capable of simulating various realistic effects including specular and diffuse global illumination; reflection, refraction; and depth of field. Additionally, the real light is estimated and used in rendering to coherently light the virtual objects. The results of the evaluation show that our system outperforms other rendering systems for AR in terms of visual quality.

Unbiased Rendering in Augmented Reality. We proposed a novel progressive rendering algorithm for augmented reality based on path tracing and differential rendering. A framework, capable of simulating complex global light transport between virtual and real worlds, was presented. Two interaction modes are available in this framework. The first one is the interactive preview mode which allows users to see the immediate view of an AR scene and interact with it. The low sampling rate is used which leads to a noisy preview. The second mode enables a progressive refinement of quality which converges to the accurate unbiased rendering solution. Our framework can possibly be used in movie previsualization, cinematic relighting or other fields.

Perception Studies. Several user studies were conducted in this PhD research to investigate the human perception of AR with different rendering effects. Mainly, two important factors were studied: Perception of realism and the sense of presence. The first two studies were focused on the perception of realism in AR with different effects. The following effects were evaluated: Depth of field, reflection/refraction, antialiasing, and caustics. The results of these studies suggest that each evaluated rendering effect increases the realism perceived by the user. Additionally, we showed that some people believed that virtual objects are real ones. In 40.1% cases, the virtual objects were marked as the real ones.

In the third study, we evaluated the effect of global and direct illumination to the sense of presence in AR. We found that global illumination leads to a higher sense of presence than direct illumination for certain scenes. Moreover, we found a significant correlation between perception of realism and the sense of presence. We showed that our participants more likely judged virtual objects as real ones with global illumination compared to direct illumination. Finally, we analyzed the differences between real and virtual objects observed by users and identified the most important features for visual realism in AR.

Our results indicate that high-quality global illumination is important for believable AR. The evaluation suggests that the rendering algorithms, presented in this thesis, are suitable for increasing perception of realism and the sense of presence; thus making AR applications more immersive, efficient, and attractive.

6.2 Future Research

Despite the high quality provided by the presented rendering algorithms, there are still several open problems which need to be solved to achieve ultimate quality of AR. This section discusses these problems and suggestions for future research.

Real-Time 3D Reconstruction of the Real Scene. People are used to interact with real objects in terms of physical and visual interaction with their own body or with other objects. Therefore, it is important to enable the same interaction with virtual objects in AR. In order to achieve this goal, the geometry, materials, and lights of the real scene have to be reconstructed in real time. Moreover, accurate geometry reconstruction can allow the correct occlusion of virtual objects by the real ones. In the future, the methods for 3D scene reconstruction (Section 2.8) can be extended and incorporated into our AR system to provide dynamic visual and physical interaction between real and virtual objects.

Glossy Reflections. Glossy reflections appear on many materials in the real world. Therefore, the simulation of glossy reflectance is important for AR. A natural future extension of our differential irradiance caching algorithm is differential radiance caching which stores the directional radiance in spherical harmonics [91]. This extension can simulate light transport on glossy and diffuse surfaces. Together with specular light transport by ray tracing and photon mapping it will cover the full range of global light transport between surfaces in AR scenes. Recent research demonstrates novel ways of storing and handling directional radiance information [154]. This method can also be used to extend our rendering. Another problem are the aliasing artifacts caused by rasterization in irradiance splatting. These artifacts can be reduced, in future work, by supersampling of the irradiance splat buffer. Finally, future research can investigate using real-time ray tracing to create high-quality AR scenes that are indistinguishable from reality.

Tracking. Accurate camera tracking is an essential part of an AR system, because even little jitter in tracking data can be immediately observed by users as unnatural and unstable behavior of virtual objects. Our system can be extended with a more stable tracking solution, for example the ioTracker [135]. Additionally, any other tracking solution from Section 2.7 can be included.

Mobile Implementation. Mobile devices become widely used in everyday life. Moreover, they can serve as a good platform for AR applications, because they include additional sensors and they provide a display coupled with a camera. Therefore, the mobile implementation of high-quality rendering in AR can be beneficial. The disadvantage of mobile devices is their limited computational power. Current mobile devices do not provide enough performance to run ray tracing in real time. One solution of this problem is to stream the tracking data to a server and render the image there. Then, the rendered image can be sent to the mobile device and composited with the camera image. This approach needs future research in the area of remote rendering, streaming, and computer networks. We believe that in the future mobile devices will provide enough power to run ray tracing based algorithms locally.

Interactivity. Real-time rendering is the area of attractive research in computer graphics. The future performance improvements of ray tracing based algorithms can be used in our system to improve the rendering speed. Additionally, possible performance improvements can be achieved by employing a GPU implementation of the Kd-tree construction [199] for photon mapping.

User Studies. Additional user studies in AR can help to better understand the human perception of this technology. The new knowledge, gained from user studies, can lead to improvements of AR technology to be more useful, natural, and user friendly. The results of our studies indicate the positive effect of global illumination rendering on the sense of presence and perception of realism in AR. It will be important in future research to replicate the results of our studies and confirm our findings. Additionally, new studies can bring interesting findings about the perception of AR. Our studies can be extended, for example, by comparing the sense of presence during rendering with different visual features. Another way of extension is to study the perception of different subparts of AR, including tracking, 3D scene reconstruction, camera parameters, delay, or lighting.

Unbiased Global Illumination in Augmented Reality. In the future, our differential progressive path tracing algorithm can be improved by advanced filtering approaches. These approaches can reduce noise in interactive preview mode caused by insufficient sampling. The examples of such methods can be seen in [42, 58, 102, 151, 170].

Physically based rendering has a high importance for AR and computer graphics. A physically accurate unbiased GI algorithm, running in real time, is the ultimate goal of rendering. The quality, provided by ray tracing based algorithms, together with performance improvement can be a solid basis for reaching this goal. We believe that due to the improvements of graphics hardware, ray tracing based rendering will become the standard rendering method for AR and other real-time applications.

Bibliography

[1] Kusuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. Photorealistic rendering for augmented reality using environment illumination. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 208–218, Washington, DC, USA, 2003. IEEE Computer Society.

[2] Okan Arikan, David A. Forsyth, and James F. O'Brien. Fast and detailed approximate global illumination by irradiance decomposition. In *ACM SIGGRAPH 2005*, pages 1108–1114, New York, NY, USA, 2005. ACM.

[3] Remi Arnaud and Mark C. Barnes. *Collada: Sailing the Gulf of 3D Digital Content Creation*. AK Peters Ltd, 2006.

[4] Michael Ashikhmin. A tone mapping algorithm for high contrast images. In *Proceedings of the 13th Eurographics Workshop on Rendering*, EGRW '02, pages 145–156, Aire-la-Ville, Switzerland, 2002. Eurographics Association.

[5] Michael Ashikhmin and Peter Shirley. An anisotropic phong BRDF model. *Journal of Graphics Tools*, 5:25–32, 2002.

[6] Gary A. Atkinson and Edwin R. Hancock. Two-dimensional BRDF estimation from polarisation. *Computer Vision and Image Understanding*, 111(2):126–141, 2008.

[7] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.

[8] 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*, GRAPHITE '06, pages 349–356, New York, NY, USA, 2006. ACM.

[9] Brian A. Barsky and Todd J. Kosloff. Algorithms for rendering depth of field effects in computer graphics. In *Proceedings of the 12th WSEAS international conference on Computers*, pages 999–1010, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[10] Louis Bavoil, Miguel Sainz, and Rouslan Dimitrov. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 talks*, pages 22:1–22:1, New York, NY, USA, 2008. ACM.

[11] Carsten Benthin, Ingo Wald, and Philipp Slusallek. A scalable approach to interactive global illumination. *Computer Graphics Forum*, 22(3):621–631, 2003.

[12] Paul. J. Besl and Neil. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 –256, February 1992.

[13] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77, pages 192–198, New York, NY, USA, 1977. ACM.

[14] Jonathan Brouillat, Pascal Gautron, and Kadi Bouatouch. Photon-driven irradiance cache. *Computer Graphics Forum*, 27(7):1971–1978, 2008.

[15] Grigore C. Burdea and Philippe Coiffet. *Virtual Reality Technology*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2003.

[16] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3D reconstruction using signed distance functions. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[17] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Dover Publications Inc., 1950.

[18] Jiating Chen, Bin Wang, and Jun-Hai Yong. Improved stochastic progressive photon mapping with metropolis sampling. In *Proceedings of the 22th Eurographics Conference on Rendering*, EGSR'11, pages 1205–1213, Aire-la-Ville, Switzerland, 2011. Eurographics Association.

[19] Youngkwan Cho, Jongweon Lee, and Ulrich Neumann. A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In *IWAR*, pages 147–165, 1998.

[20] Per H. Christensen. Point-based approximate color bleeding. Technical report, Pixar Animation Studios, 2008.

[21] Andrew I. Comport, Éric Marchand, and François Chaumette. A real-time tracker for markerless augmented reality. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 36–45. IEEE Computer Society, 2003.

[22] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM.

[23] Sam Corbett-Davies, Andreas Dünser, and Adrian Clark. An advanced interaction framework for augmented reality based exposure treatment. In *IEEE Virtual Reality (VR)*, pages 19–22, Orlando, Florida, USA, March 2013.

[24] Oliver Cossairt, Shree Nayar, and Ravi Ramamoorthi. Light field transfer: global illumination between real and synthetic objects. In *ACM SIGGRAPH 2008*, pages 57:1–57:6, New York, NY, USA, 2008. ACM.

[25] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2011)*, 30(7), September 2011.

[26] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 203–231, New York, NY, USA, 2005. ACM.

[27] 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 the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 189–198, New York, 1998. ACM.

[28] Paul Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[29] Artur L. Dos Santos, Diego Lemos, Jorge E. F. Lindoso, and Veronica Teichrieb. Real time ray tracing for augmented reality. In *14th Symposium on Virtual and Augmented Reality (SVR)*, pages 131–140, May 2012.

[30] Frédéric Drago, Karol Myszkowski, Thomas Annen, and Norishige Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. *Computer Graphics Forum*, 22(3):419–426, 2003.

[31] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques, SIGGRAPH '02*, pages 257–266, New York, NY, USA, 2002. ACM.

[32] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): Part I the essential algorithms. *Robotics & Automation Magazine*, 13(2):99–110, 2006.

[33] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A K Peters, Ltd., Natick, MA, USA, 2002.

[34] Bartosz Fabianowski and John Dingliana. Interactive global photon mapping. *Computer Graphics Forum*, 28(4):1151–1159, 2009.

[35] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 249–256, New York, NY, USA, 2002. ACM.

[36] Alain Fournier, Atjeng S. Gunawan, and Chris Romanzin. Common illumination between real and computer generated scenes. In *Proceedings of Graphics Interface '93*, pages 254–262, Toronto, Canada, 1993.

[37] Jan-Michael Frahm, Kevin Koeser, Daniel Grest, and Reinhard Koch. Markerless augmented reality with light source estimation for direct illumination. In *The 2nd IEE European Conference on Visual Media Production, CVMP 2005*, pages 211–220, 2005.

[38] Tobias Franke. Delta light propagation volumes for mixed reality. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2013, pages 125–132. IEEE Computer Society, October 2013.

[39] Tobias Franke. Delta voxel cone tracing. In *Proceedings of International Symposium on Mixed and Augmented Reality*, ISMAR 2014. IEEE Computer Society, 2014.

[40] Tobias Franke and Yvonne Jung. Real-time mixed reality with gpu techniques. In *GRAPP*, pages 249–252, 2008.

[41] Nakano Gaku, Kitahara Itaru, and Ohta Yuichi. Generating perceptually-correct shadows for mixed reality. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 173–174, Washington, DC, USA, 2008. IEEE Computer Society.

[42] Eduardo S. L. Gastal and Manuel M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics, SIGGRAPH 2012*, 31(4):1–13, 2012.

[43] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3):335–360, September 2011.

[44] Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Temporal radiance caching. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):891–901, 2007.

[45] Pascal Gautron, Jaroslav Křivánek, Kadi Bouatouch, and Sumanta N. Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques 2005, Eurographics Symposium on Rendering*, pages 55–64. Eurographics Association, 2005.

[46] Abhijeet Ghosh, Shruthi Achutha, Wolfgang Heidrich, and Matthew O'Toole. BRDF acquisition with basis illumination. In *IEEE 11th International Conference on Computer Vision, 2007*, pages 1–8, October 2007.

[47] Dan B. Goldman, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and spatially-varying BRDFs from photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2010.

[48] Costantino Grana, Daniele Borghesani, and Rita Cucchiara. Connected component labeling techniques on modern architectures. In *Proceedings of the 15th International Conference on Image Analysis and Processing*, ICIAP '09, pages 816–824, Berlin, Heidelberg, 2009. Springer-Verlag.

[49] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, March 1998.

[50] Thorsten Grosch. Differential photon mapping: Consistent augmentation of photographs with correction of all light paths. In *Eurographics 2005 Short Papers*, 2005.

[51] 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*, VRST '07, pages 125–132, New York, NY, USA, 2007. ACM.

[52] J. P. Grossman and William J. Dally. Point sample rendering. In *Rendering Techniques '98*, pages 181–192. Springer, 1998.

[53] Lukas Gruber, Steffen Gauglitz, Jonathan Ventura, Stefanie Zollmann, Manuel Huber, Michael Schlegel, Gudrun Klinker, Dieter Schmalstieg, and Tobias Höllerer. The city of sights: Design, construction, and measurement of an augmented reality stage set. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR'10, pages 157–163, Seoul, Korea, October 2010.

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

[55] Johannes Günther, Ingo Wald, and Philipp Slusallek. Realtime caustics using distributed photon mapping. In *Proceedings of the 15th Eurographics Symposium on Rendering*, pages 111–121, June 2004.

[56] Sanket Gupte. Real-time photon mapping on gpu. Technical report, University of Maryland, Baltimore County, 2011.

[57] Toshiya Hachisuka. *GPU Gems 2: Programming Techniques for High Performance Graphics and General-Purpose Computation*, chapter 38. 2. Addison-Wesley, 2005.

[58] 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. *ACM Transactions on Graphics, SIGGRAPH 2008*, 27(3):33:1–33:10, August 2008.

[59] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *ACM SIGGRAPH Asia 2008*, pages 130:1–130:8, New York, NY, USA, 2008. ACM.

[60] Kenji Hara, Ko Nishino, and Katsushi Ikeuchi. Light source position and reflectance estimation from a single view without the distant illumination assumption. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):493–505, April 2005.

[61] Vlastimil Havran. *Heuristic Ray Shooting Algorithms*. PhD thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

[62] Jan Herling and Wolfgang Broll. Markerless tracking for augmented reality. In *Handbook of Augmented Reality*, pages 255–272. Springer New York, 2011.

[63] Robert Herzog, Vlastimil Havran, Shinichi Kinuwaki, Karol Myszkowski, and Hans-Peter Seidel. Global illumination using photon ray splatting. In *Computer Graphics Forum*, volume 26(3), pages 503–513, Prague, Czech Republic, 2007. Blackwell.

[64] Sébastien Hillaire, Anatole Lécuyer, Rémi Cozot, and Géry Casiez. Depth-of-field blur effects for first-person navigation in virtual environments. *IEEE Computer Graphics and Applications*, 28:47–55, November 2008.

[65] Ernest W. Hobson. *The Theory of Spherical and Ellipsoidal Harmonics*. Chelsea Pub. Co., 1955.

[66] Hunter G. Hoffman, Gloria T. Chambers, Walter J. Meyer III, Lisa L. Arceneaux, William J. Russell, Eric J. Seibel, Todd L. Richards, Sam R. Sharar, and David R. Patterson. Virtual reality as an adjunctive non-pharmacologic analgesic for acute burn pain during medical procedures. *Annals of Behavioral Medicine*, 41(2):183–191, 2011.

[67] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 133–142, New York, NY, USA, 1986. ACM.

[68] Katrien Jacobs and Celine Loscos. Classification of Illumination Methods for Mixed Reality. *Computer Graphics Forum*, 25(1):29–51, March 2006.

[69] Wojciech Jarosz, Volker Schönefeld, Leif Kobbelt, and Henrik Wann Jensen. Theory, analysis and applications of 2D global illumination. *ACM Transactions on Graphics*, 31(5):1–21, 2012.

[70] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, Eurographics, pages 21–30. Springer, Vienna, 1996.

[71] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Limited, 2009.

[72] Henrik Wann Jensen, Frank Suykens, Per Christensen, and Toshi Kato. A practical guide to global illumination using photon mapping. In *SIGGRAPH Course Notes*, 2002.

[73] James T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[74] Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pages 99–107, New York, NY, USA, 2010. ACM.

[75] Anton Kaplanyan and Carsten Dachsbacher. Path space regularization for holistic and robust light transport. *Computer Graphics Forum*, 32(2), 2013.

[76] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[77] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *ACM SIGGRAPH Asia*, pages 157:1–157:12, New York, NY, USA, 2011. ACM.

[78] Hirokazu Kato and Mark Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, Washington, DC, USA, 1999. IEEE Computer Society.

[79] Alexander Keller. Quasi-monte carlo radiosity. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 101–110, London, UK, 1996. Springer-Verlag.

[80] Alexander Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[81] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality*, ISMAR'07, Nara, Japan, November 2007.

[82] Martin Knecht. *Reciprocal Shading for Mixed Reality*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria, December 2013.

[83] Martin Knecht, Andreas Dünser, Christoph Traxler, Michael Wimmer, and Raphael Grasset. A Framework For Perceptual Studies In Photorealistic Augmented Reality. In *Proceedings of the 3rd IEEE VR 2011 Workshop on Perceptual Illusions in Virtual Environments*, pages 27–32, 2011.

[84] Martin Knecht, Georg Tanzmeister, Christoph Traxler, and Michael Wimmer. Interactive BRDF estimation for mixed-reality applications. *Journal of WSCG*, 20(1):47–56, June 2012.

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

[86] Martin Knecht, Christoph Traxler, Oliver Mattausch, and Michael Wimmer. Reciprocal shading for mixed reality. *Computers & Graphics*, 36(7):846–856, November 2012.

[87] Martin Knecht, Christoph Traxler, Christoph Winklhofer, and Michael Wimmer. Reflective and refractive objects for mixed reality. *IEEE Transactions on Visualization and Computer Graphics*, 19(4):576–582, April 2013.

[88] Kalin Kolev and Daniel Cremers. Integration of multiview stereo and silhouettes via convex functionals on convex domains. In *Computer Vision – ECCV 2008*, volume 5302 of *Lecture Notes in Computer Science*, pages 752–765. Springer Berlin Heidelberg, 2008.

[89] Jaroslav Křivánek, Kadi Bouatouch, Sumanta N. Pattanaik, and Jiří Žára. Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques, Eurographics Symposium on Rendering*, pages 127–138, Nicosia, Cyprus, June 2006. Eurographics Association.

[90] Jaroslav Křivánek. *Radiance Caching for Global Illumination Computation on Glossy Surfaces*. PhD thesis, Université de Rennes 1 and Czech Technical University in Prague, December 2005.

[91] Jaroslav Křivánek and Pascal Gautron. *Practical Global Illumination with Irradiance Caching*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2009.

[92] Jaroslav Křivánek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Improved radiance gradient computation. In *Proceedings of the 21st spring conference on Computer graphics*, SCCG '05, pages 155–159, New York, NY, USA, 2005. ACM.

[93] Eric P. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Katholieke Universiteit Leuven, 1996.

[94] Eric P. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[95] Eric P. Lafortune and Yves D. Willems. Bi-directional path Tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, 1993.

[96] Eric P. Lafortune and Yves D. Willems. Using the modified phong reflectance model for physically based rendering. Technical report, Department of Computing Science, K.U. Leuven, 1994.

[97] Bent Dalgaard Larsen and Niels Jørgen Christensen. Simulating photon mapping for real-time applications. In *Eurographics Symposium on Rendering*, pages 123–132. Eurographics Association, Jun 2004.

[98] Christian Lauterbach, Sung eui Yoon, David Tuft, and Dinesh Manocha. RT-DEFORM: Interactive ray tracing of dynamic scenes using BVHs. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing*, pages 39–45, 2006.

[99] Fabien Lavignotte and Mathias Paulin. Scalable photon splatting for global illumination. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 203–210, New York, 2003. ACM.

[100] Kwan Min Lee. Presence, explicated. *Communication Theory*, 14(1):27–50, 2004.

[101] Taehee Lee and Tobias Höllerer. Multithreaded hybrid feature tracking for markerless augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):355–368, May 2009.

[102] Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. Reconstructing the indirect light field for global illumination. *ACM Transactions on Graphics, SIGGRAPH 2012*, 31(4):51:1–51:10, July 2012.

[103] Eric Lengyel. *Mathematics for 3D game programming and computer graphics, third edition*. Game development series. Course Technology/Cengage Learning, 2012.

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

[105] Robert R. Lewis. Making shaders more physically plausible. *Computer Graphics Forum*, 13(2):109–120, 1994.

[106] Matthew Lombard and Theresa Ditton. At the heart of it all: The concept of presence. *Journal of Computer-Mediated Communication*, 3(2), 1997.

[107] Sebastian Magda, David J. Kriegman, Todd Zickler, and Peter N. Belhumeur. Beyond lambert: Reconstructing surfaces with arbitrary BRDFs. In *ICCV*, pages 391–399, 2001.

[108] Katerina Mania and Andrew Robinson. The effect of quality of rendering on user lighting impressions and presence in virtual environments. In *International conference on Virtual Reality continuum and its applications in industry*, VRCAI '04, pages 200–205, NY, USA, 2004. ACM.

[109] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. In *ACM SIGGRAPH 2003*, pages 759–769, New York, NY, USA, 2003. ACM.

[110] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic BRDF measurement. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, pages 241–247, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[111] Morgan McGuire. Computer graphics archive, 2011. http://graphics.cs.williams.edu/data.

[112] Morgan McGuire and David Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conference on High Performance Graphics*, pages 77–89, New York, NY, USA, August 2009. ACM.

[113] Maxime Meilland, Christian Barat, and Andrew Comport. 3D high dynamic range dense visual slam and its application to real-time object re-lighting. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2013, pages 143–152, October 2013.

[114] Don P. Mitchell. Consequences of stratified sampling in graphics. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 277–280, New York, NY, USA, 1996. ACM.

[115] Martin Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, pages 97–121, New York, NY, USA, 2007. ACM.

[116] Benjamin Mora. Naive ray-tracing: A divide-and-conquer approach. *ACM Transactions on Graphics*, 30(5):117:1–117:12, October 2011.

[117] Gero Müller, Jan Meseth, Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Acquisition, synthesis and rendering of bidirectional texture functions. In *Eurographics 2004, State of the Art Reports*, pages 69–94. INRIA and Eurographics Association, September 2004.

[118] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2011, pages 127–136. IEEE Computer Society, October 2011.

[119] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, 2011.

[120] Peter Nillius and Jan-Olof Eklundh. Automatic estimation of the projected light source direction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1 of *CVPR 2001*, pages 1076–1083. IEEE Computer Society, 2001.

[121] Jan Novák, Vlastimil Havran, and Carsten Daschbacher. Path regeneration for interactive path tracing. In *EUROGRAPHICS 2007, Short papers*, pages 61–64. The European Association for Computer Graphics, 2010.

[122] Bunyo Okumura, Masayuki Kanbara, and Naokazu Yokoya. Augmented reality based on estimation of defocusing and motion blurring from captured images. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '06, pages 219–225, Washington, DC, USA, 2006. IEEE Computer Society.

[123] Michael Oren and Shree K. Nayar. Generalization of lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 239–246, New York, NY, USA, 1994. ACM.

[124] Geoffrey Oxholm and Ko Nishino. Shape and reflectance from natural illumination. In *Computer Vision - ECCV 2012*, volume 7572 of *Lecture Notes in Computer Science*, pages 528–541. Springer Berlin Heidelberg, 2012.

[125] Anthony Pajot, Loïc Barthe, Mathias Paulin, and Pierre Poulin. Combinatorial bidirectional path-tracing for efficient hybrid cpu/gpu rendering. *Computer Graphics Forum*, 30(2):315–324, 2011.

[126] Jun Park, Suya You, and Ulrich Neumann. Natural feature tracking for extendible robust augmented realities. In *International Workshop on Augmented Reality (IWAR)'98*, 1998.

[127] Youngmin Park, Vincent Lepetit, and Woontack Woo. Esm-blur: Handling & rendering blur in 3D tracking and augmentation. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, pages 163–166, Washington, DC, USA, 2009. IEEE Computer Society.

[128] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: a general purpose ray tracing engine. *ACM Transactions on Graphics*, 29, 2010.

[129] Gustavo Patow and Xavier Pueyo. A Survey of Inverse Rendering Problems. *Computer Graphics Forum*, 22(4):663–687, 2003.

[130] Sumanta Pattanaik and Hector Yee. Adaptive gain control for high dynamic range image display. In *Proceedings of the 18th Spring Conference on Computer Graphics*, SCCG '02, pages 83–87, New York, NY, USA, 2002. ACM.

[131] Saulo A. Pessoa, Guilherme de S. Moura, João Paulo S. M. Lima, Veronica Teichrieb, and Judith Kelner. Rpr-sors: Real-time photorealistic rendering of synthetic objects into real scenes. *Computers & Graphics*, 36(2):50–69, 2012.

[132] Saulo A. Pessoa, Guilherme de S. Moura, João Paulo S. M. Lima, Veronica Teichrieb, and Judith Kelner. Photorealistic rendering for augmented reality: A global illumination and

BRDF solution. In *IEEE Virtual Reality Conference (VR)*, pages 3–10. IEEE Computer Society, March 2010.

[133] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Elsevier Science, 2010.

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

[135] Thomas Pintaric and Hannes Kaufmann. Affordable Infrared-Optical Pose Tracking for Virtual and Augmented Reality. In *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, pages 44–51, 2007.

[136] Sören Pirk. *GPU-Based Rendering of Reflective and Refractive Objects in Augmented Reality Environments*. Master's thesis, University of Applied Sciences, Oldenburg, 2007.

[137] Andreas Pomi and Philipp Slusallek. Interactive Ray Tracing for Virtual TV Studio Applications. *Journal of Virtual Reality and Broadcasting*, 2(1), December 2005.

[138] Vivek Pradeep, Christoph Rhemann, Shahram Izadi, Christopher Zach, Michael Bleyer, and Steven Bathiche. Monofusion: Real-time 3D reconstruction of small scenes with a single web camera. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2013, pages 83–88, October 2013.

[139] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics, SIGGRAPH 2002*, 21(3):703–712, July 2002.

[140] Timothy J. Purcell, Craig Donner, Mike Cammarano, Henrik Wann Jensen, and Pat Hanrahan. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '03, pages 41–50, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[141] 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*, SIGGRAPH '01, pages 497–500, New York, 2001. ACM.

[142] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 117–128, New York, NY, USA, 2001. ACM.

[143] Holger Regenbrecht, Cristina Botella, Rosa Banos, and Thomas Schubert. Mixed Reality Experience Questionnaire v1.0., 2013. Unpublished online document. http://www.hci.otago.ac.nz/mreq/MixedRealityExperienceQuestionnaireWeb.htm, last accessed 23/May/2013.

[144] Holger Regenbrecht and Thomas Schubert. Measuring Presence in Augmented Reality Environments: Design and a First Test of a Questionnaire. In *Proceedings of the Fifth Annual International Workshop Presence*, 2002.

[145] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 267–276, New York, NY, USA, 2002. ACM.

[146] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1):160–188, 2012.

[147] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, and Carsten Dachsbacher. Micro-rendering for scalable, parallel final gathering. In *ACM SIGGRAPH Asia*, pages 132:1–132:8, New York, NY, USA, 2009. ACM.

[148] Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans-Peter Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect shadow maps for efficient computation of indirect illumination. In *ACM SIGGRAPH Asia*, pages 129:1–129:8, New York, NY, USA, 2008. ACM.

[149] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 75–82, New York, NY, USA, 2009. ACM.

[150] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive sampling and reconstruction using greedy error minimization. In *SIGGRAPH Asia*, pages 159:1–159:12, New York, NY, USA, 2011. ACM.

[151] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive rendering with non-local means filtering. *ACM Transactions on Graphics*, 31(6):195:1–195:11, November 2012.

[152] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):290–300, March 2003.

[153] Fabian Scheer, Oliver Abert, and Stefan Müller. Towards using realistic ray tracing in augmented reality applications with natural lighting. *GI Workshop ARVR 07*, 2007.

[154] Daniel Scherzer, Chuong H. Nguyen, Tobias Ritschel, and Hans-Peter Seidel. Pre-convolved Radiance Caching. *Computer Graphics Forum*, 4(31), 2012.

[155] Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben, and Jon Sporring. Photon differentials. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 179–186, New York, NY, USA, 2007. ACM.

[156] Christophe Schlick. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum*, 13:233–246, 1994.

[157] Martijn J. Schuemie, Peter van der Straaten, and Merel Krijn. Research on presence in virtual reality: A survey. *CyberPsychology & Behavior*, 4(2):183–201, 2001.

[158] Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. Practical hessian-based error control for irradiance caching. *ACM Transactions on Graphics, SIGGRAPH Asia*, 31(6), November 2012.

[159] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528. IEEE Computer Society, 2006.

[160] Perumaal Shanmugam and Okan Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 73–80, New York, NY, USA, 2007. ACM.

[161] Hyunjung Shim. Estimating all frequency lighting using a color/depth image. In *19th IEEE International Conference on Image Processing (ICIP)*, pages 565–568, September 2012.

[162] Bernard W. Silverman. *Density estimation for statistics and data analysis*. Chapmann and Hall, New York, 1986.

[163] Mel Slater, Pankaj Khanna, Jesper Mortensen, and Insu Yu. Visual Realism Enhances Realistic Response in an Immersive Virtual Environment. *IEEE Computer Graphics and Applications*, 29(3):76–84, 2009.

[164] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 527–536, New York, NY, USA, 2002. ACM.

[165] Philipp Slusallek, Peter Shirley, William Mark, Gordon Stoll, and Ingo Wald. Introduction to real-time ray tracing. In *ACM SIGGRAPH 2005 Courses*, New York, NY, USA, 2005. ACM.

[166] Miłosław Smyk, Shinichi Kinuwaki, Roman Ďurikovič, and Karol Myszkowski. Temporally coherent irradiance caching for high quality animation rendering. *Computer Graphics Forum*, 24(3):401–412, 2005.

[167] Ben Spencer and Mark W. Jones. Hierarchical photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):49–61, 2009.

[168] Ben Spencer and Mark W. Jones. Into the blue: Better caustics through photon relaxation. *Computer Graphics Forum*, 28(2):319–328, 2009.

[169] Natsuki Sugano, Hirokazu Kato, and Keihachiro Tachibana. The effects of shadow representation of virtual objects in augmented reality. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 76–83, Washington, DC, USA, 2003. IEEE Computer Society.

[170] Frank Suykens and Yves D. Willems. Adaptive filtering for progressive monte carlo image rendering. In *The International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, WSCG, 2000.

[171] László Szirmay-Kalos and Werner Purgathofer. Global ray-bundle tracing with hardware acceleration. In *Rendering Techniques*, pages 247–258. Springer, 1998.

[172] László Szirmay-Kalos, László Szécsi, and Mateu Sbert. *GPU-Based Techniques for Global Illumination Effects*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008.

[173] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. In *ACM SIGGRAPH 2004*, pages 469–476, New York, NY, USA, 2004. ACM.

[174] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular slam in dynamic environments. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2013, pages 209–218, October 2013.

[175] Petri Tanskanen, Kalin Kolev, Lorenz Meier, Federico Camposeco, Olivier Saurer, and Marc Pollefeys. Live metric 3D reconstruction on mobile phones. In *ICCV*, 2013.

[176] 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*, I3D '12, pages 183–190, New York, NY, USA, 2012. ACM.

[177] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 537–546, New York, NY, USA, 2002. ACM.

[178] Kenneth E. Torrance and Edward M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57(9):1105–1112, September 1967.

[179] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '04, pages 48–57, Washington, DC, USA, 2004. IEEE Computer Society.

[180] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[181] Sebastian Vogt, Ali Khamene, Frank Sauer, and Heinrich Niemann. Single camera tracking of marker clusters: multiparameter cluster optimization and experimental verification. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2002, pages 127–136, 2002.

[182] Hoang-Hiep Vu, Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, 2012.

[183] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2008, pages 125–134, September 2008.

[184] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368, May 2010.

[185] Daniel Wagner and Dieter Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, February 2007.

[186] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Distributed interactive ray tracing of dynamic scenes. In *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, PVG '03, pages 77–85, Washington, DC, USA, 2003. IEEE Computer Society.

[187] Bruce Walter. Notes on the ward BRDF. Technical report, Cornell Program of Computer Graphics, 2005.

[188] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient gpu-based approach for interactive global illumination. In *ACM SIGGRAPH 2009*, pages 91:1–91:8, New York, NY, USA, 2009. ACM.

[189] Rui Wang, Jiajun Zhu, and Greg Humphreys. Precomputed radiance transfer for real-time indirect lighting using a spectral mesh basis. In *Eurographics Symposium on Rendering*, 2007.

[190] Yang Wang and Dimitris Samaras. Estimation of multiple directional light sources for synthesis of mixed reality images. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, PG '02, pages 185–205, Washington, DC, USA, 2002. IEEE Computer Society.

[191] Greg Ward and Paul Heckbert. Irradiance gradients. In *Eurographics Rendering Workshop*, pages 85–98, May 1992.

[192] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 265–272, New York, NY, USA, 1992. ACM.

[193] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 85–92, New York, 1988. ACM.

[194] Thomas Whelan, Michael Kaess, Maurice F. Fallon, Hordur Johannsson, John J. Leonard, and John B. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[195] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[196] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '78, pages 270–274, New York, NY, USA, 1978. ACM.

[197] Sven Woop, Louis Feng, Ingo Wald, and Carsten Benthin. Embree ray tracing kernels for cpus and the xeon phi architecture. In *ACM SIGGRAPH 2013 Talks*, New York, NY, USA, 2013. ACM.

[198] Zuoyong Zheng, Lizhuang Ma, Zhong Li, and Zhihua Chen. Reconstruction of shape and reflectance properties based on visual hull. In *Proceedings of the 2009 Computer Graphics International Conference*, CGI '09, pages 29–38, New York, NY, USA, 2009. ACM.

[199] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM SIGGRAPH Asia*, pages 126:1–126:11, New York, NY, USA, 2008. ACM.

[200] Paul Zimmons and Abigail Panter. The influence of rendering quality on presence and task performance in a virtual environment. In *IEEE Virtual Reality*, pages 293–294, 2003.