

---

## Abstract

---

Classic relighting applications are striving to unite the virtual world and the real world by applying computer graphics algorithms to pixel and image-based descriptions. This has allowed them to apply new virtual lighting conditions on real images as well as inserting virtual objects in real environments under credible lighting conditions. However, state-of-the-art representations for geometry, materials and lighting often limit the capabilities and quality of the simulation of light in relighting applications. Spherical harmonics allow for a fast simulation of light, but can only handle low-frequency lighting effects efficiently. In addition, other relighting applications rely on Haar wavelets; which are capable of representing high-frequency lighting information as well as having great compression performance. In theory, Haar wavelets have an efficient forward rendering evaluation method. However, in practice, they need a complex rotation operator and the three factors of the rendering equation can not be constructed dynamically. In addition, they lack smoothness, which is essential for relighting applications. To overcome most of these constraints, this dissertation researched other, possibly better, representations. This dissertation introduces two new underlying basis representations designed to improve cutting edge relighting algorithms.

First, we will introduce an efficient algorithm to calculate the triple product integral binding coefficients for a heterogeneous mix of wavelet bases. As mentioned above, Haar wavelets excel at encoding piecewise constant signals, but are inadequate for compactly representing smooth signals for which high-order wavelets are ideal. Our algorithm provides an efficient way to calculate the tensor of these binding coefficients, which is essential for the correct evaluation of the light transport integral. The algorithm exploits both the hierarchical nature and vanishing moments of the wavelet bases, as well as the sparsity and symmetry of the tensor. The effectiveness of high-order wavelets will be demonstrated in a rendering application. The smoother wavelets represent the signals more effectively and with less blockiness than their Haar wavelet counterpart.

Using a heterogeneous mix of wavelets allows us to overcome the smoothness problem. However, wavelets still constrain one or several factors of the rendering equation, keeping them

inadequate for more interactive rendering applications. For example, visibility is often pre-calculated and animations are not allowed; and changes in lighting are limited to a simple rotation and are not very detailed. Other techniques compromise on quality and often coarsely tabulate BRDF functions. In the second part of this dissertation, we research how spherical radial basis functions (SRBFs) can be used to overcome most of these problems. SRBFs have already been used in forward rendering, but they still do not guarantee full interactivity of the underlying factors of geometry, materials and lighting. We argue that an interactive representation of the factors is crucial and will greatly improve the flexibility and efficiency of a relighting algorithm. In order to dynamically change lighting conditions or alter scene geometry and materials, these three factors need to be converted to the SRBF representation in a fast manner. This dissertation presents a method to perform the SRBF data construction and rendering in real-time. To support dynamic high-frequency lighting, a multiscale residual transformation algorithm is applied. Area lights are detected through a peak detection algorithm. By using voxel cone tracing and a subsampling scheme, animated geometry casts soft shadows dynamically.

At this point, we have two new approaches for evaluating triple product rendering integrals with fewer coefficients and an advantageous smoothness behavior. But how will they perform in actual relighting applications? We tried to answer this question in the final part of this dissertation by conducting experiments in two distinct use cases.

A first use case focuses on the relighting of virtual objects with real lighting information of existing scenes. To demonstrate this, we have developed an augmented reality application. The ambition is to augment omnidirectional video, also called 360° video, with natural lit virtual objects and to make the experience more realistic for users. Recent years have known a proliferation of real-time capturing and rendering methods for omnidirectional video. Together with these technologies, rendering devices such as virtual reality glasses have tried to increase the immersive experience of users. Structure-from-motion is applied to the omnidirectional video to reconstruct the trajectory of the camera. Then, the position of an inserted virtual object is linked to the appropriate 360° environment map. State-of-the-art augmented reality applications have often lacked realistic appearance and lighting, but our spherical radial basis rendering framework is capable of evaluating the rendering equation in real-time with fully dynamic factors. The captured omnidirectional video can be directly used as lighting information by feeding it to our renderer, where it is instantly transformed to the proper SRBF basis. We demonstrate an application in which a computer generated vehicle can be controlled through an urban environment.

The second use case addresses the relighting of real objects. It will show more practical examples of how an improved representation will influence the quality and time performance of existing inverse rendering and intrinsic image decomposition applications. Such relighting techniques try to extract geometry, material and lighting information of real scenes out of one

---

or multiple input images. First, we show how an inverse rendering technique, as introduced by Haber et al. [Haber et al., 2009], would benefit from the smooth behavior of our high-order wavelet or SRBF representation. To allow for a hierarchical optimization algorithm, where the lower level coefficients are estimated first and then more detailed coefficients are inserted based on the well-posedness of the system, it is essential that the lower level coefficients are good approximates of the signal to estimate and thus have a smooth behavior. Besides a better refinement method, we also show how an integration with our SRBF triple product renderer will reduce the execution time of the optimization process from hours to minutes. Then, in a second application, we conduct experiments on the existing intrinsic image decomposition problem of Barron and Malik [Barron and Malik, 2015], where we used our SRBF renderer in combination with a prior based optimization method. We achieve this by adapting the SRBF rendering framework to export the proper gradients for the L-BFGS minimization step.



---

## Acknowledgments

---

When I first started my PhD, some people said to me: “*it’s gonna be a bumpy road*”. And indeed, a bumpy road it was. However, even if it felt sometimes like mountaineering and otherwise more like a free fall, it was definitely an experience of a lifetime and a road worth taking. My road was directly or indirectly shaped by numerous people who deserve a special acknowledgment.

I would like to express my very great appreciation to Prof. Dr. Philippe Bekaert, who has been my biggest source of inspiration. Ever since I attended his first courses on image processing and computer graphics I was intrigued by his passion for research. I’m particular grateful that he gave me the possibilities to achieve my goals and provided me with an inspiring research environment.

I wish to thank Prof. Dr. Eddy Flerackers and Prof. Dr. Frank Van Reeth, managing directors of the Expertise Center for Digital Media (EDM) for providing me with an excellent working environment. Furthermore, I want to thank the other professors of the faculty Prof. Dr. Karin Coninx, Prof. Dr. Wim Lamotte, Prof. Dr. Kris Luyten, Prof. Dr. Fabian Di Fiore, Prof. Dr. Peter Quax, Prof. Dr. Mieke Haesen, Prof. Dr. Johannes Schöning and—for a brief but fun time—Prof. Dr. ir. Gauthier Lafruit for all their support during my work. My special thanks are extended to Ingrid Konings, Roger Claes, Peter Vandoren and Edith Cloes for managerial and administrative support and to Prof. Dr. Peter Quax, Tom De Weyer, Raf Menten and Donald Degaen for technical support.

I would like to express my gratitude to my PhD committee and jury: Prof. Dr. Philippe Bekaert, Prof. Dr. Frank Van Reeth, Prof. Dr. Ir. Gauthier Lafruit, Prof. Dr. Ir. Phillip Dutré, Prof. Dr. Hendrik P.A. Lensch and Prof. Dr. Jean-Yves Guillemaut. Thank you for taking effort in reading, giving feedback and evaluating my dissertation.

My time at the visual computing group of the EDM allowed me to work with inspiring researchers. In particularly I am very grateful for working together with 1.05a-roommate Jeroen

Put. He significantly contributed to my research. I loved our “*constant, consistent and consequent*” discussions about the magical world of research and life in general. We’ve had a great time together during our pair programming and pair paper writing sessions. Without our way of brainstorming, my dissertation would be virtually impossible.

Furthermore, I would like to thank my other colleagues and ex-colleagues Lode Jorissen, Steven Maesen, Dr. Patrik Goorts, Dr. Tom Haber, Thomas Kovac, Dr. Ing. Sammy Rogmans, Dr. Maarten Dumont, Dr. Cosmin Ancuti, Dr. Codruta Ancuti, Dr. Bert De Decker, Dr. Tom Cuypers, Dr. Cedric Vanaken, Dr. Ashish Doshi, Jun Liu, Rajesh Chenchu and Johannes Taelman for giving me an awesome time at the visual computing research group. By extension, I want to thank all other colleagues of the EDM.

I want to express my thanks to my bachelor and master thesis supervisors Dr. Tom Cuypers, Dr. Bert de Decker and Dr. Tom Haber who gave me the inspiration for a thesis in computer vision and who convinced me to go into research. Furthermore, Dr. Tom Haber introduced me to the field of relighting and his prior research and experience have helped me a great deal in completing my dissertation.

I met some amazing people during my studies computer science. I would like to thank my *fellow rhinos* Robin Marx, Kevin Boutsen, Jimmy Cleuren, Sören Cuypers, Wouter Nivelle, Steven Reekmans, Jens Bruggemans, Kenneth Devloo and Pallieter Verlinden who helped me in developing the managing, technical and social skills necessary to complete my dissertation. Furthermore, I would like to thank all the students of *Kot Koningsoord* for all the necessary distractions during my studies.

Thanks *ma* and *pa* for giving me all the great opportunities in life. I’ll only succeed in life because I was raised by two great parents who were always there for me with all their support. Also a special thanks to my brother Stijn and sister Evi, Claudia, Jens, Francis, Louise, Noélie, my parents-in-law and the rest of the family.

Last but not least, I want to thank my wife Dr. Sanne Nelissen. I met her during my studies computer science while she was studying medicine. She has been the most fundamental and crucial part in the whole process. She has the ability of making me laugh, supporting me, encouraging me, entertaining me, inspiring me, calming me and most of all loving me.

Thank you,

Nick Michiels

---

## Contents

---

<b>Abstract</b>	<b>i</b>
<b>Acknowledgments</b>	<b>v</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	6
1.2 Contributions . . . . .	9
1.3 Dissertation Overview . . . . .	10
<b>2 Background on Relighting</b>	<b>11</b>
2.1 Simulation of Light . . . . .	12
2.1.1 Global Illumination . . . . .	12
2.1.2 The Rendering Equation . . . . .	12
2.1.3 Evaluating the Rendering Equation . . . . .	13
2.2 Precomputed Radiance Transfer . . . . .	15
2.2.1 Spherical Harmonics . . . . .	17
2.2.2 Wavelets . . . . .	17
2.2.3 Spherical Radial Basis Functions . . . . .	19
2.3 Relighting Applications . . . . .	20
<b>3 General Triple Product Theorem for Wavelets</b>	<b>25</b>
3.1 Related Work . . . . .	27
3.2 Choosing a Suitable Wavelet Basis . . . . .	30
3.2.1 Spherical Wavelets versus 2D Wavelets . . . . .	30

3.2.2	Wavelets Tailored to the Signal . . . . .	33
3.3	Wavelet Product Integral . . . . .	34
3.3.1	The Haar Tripling Coefficient Theorem . . . . .	35
3.3.2	General Tripling Coefficient Theorem . . . . .	36
3.4	Tensor of Triple Product Binding Coefficients . . . . .	40
3.4.1	Naive Approach . . . . .	40
3.4.2	Hierarchical Approach . . . . .	40
3.4.3	Tensor Mirroring . . . . .	40
3.4.4	Wavelet Sliding . . . . .	41
3.4.5	Vanishing Moments . . . . .	42
3.4.6	Tensor Sparseness . . . . .	42
3.5	Results and Applications . . . . .	44
3.6	Discussion . . . . .	46
3.7	Conclusions . . . . .	48
<b>4</b>	<b>Spherical Radial Basis Functions for Interactive Triple Product Rendering</b>	<b>51</b>
4.1	Spherical Radial Basis Functions Product Integral . . . . .	54
4.1.1	Types of Spherical Radial Basis Functions . . . . .	54
4.1.2	Rendering with Spherical Radial Basis Functions . . . . .	54
4.1.3	Spherical Radial Basis Functions versus Wavelets . . . . .	55
4.1.3.1	Analytic Evaluation of the Product and Convolution of Two or More SRBFs . . . . .	55
4.1.3.2	Fast BRDF Evaluation . . . . .	55
4.1.3.3	Rotation to the Local Frame . . . . .	56
4.1.3.4	High-frequency Information . . . . .	57
4.1.3.5	Interactivity . . . . .	57
4.1.4	Related Work . . . . .	57
4.2	Dynamic Materials . . . . .	60
4.3	Dynamic Visibility . . . . .	61
4.3.1	Voxelization . . . . .	62
4.3.2	Cone Tracing . . . . .	63
4.3.3	Mapping Cones to SRBFs . . . . .	64
4.4	Dynamic Lighting . . . . .	66
4.4.1	SRBF Approximation . . . . .	66
4.4.1.1	Hierarchical Grid of SRBFs using the Healpix Distribution . . . . .	67
4.4.1.2	Residual Transform to Multiscale SRBFs . . . . .	68
4.4.2	Calculating Overlapping SRBFs . . . . .	71
4.4.3	GPU Implementation . . . . .	72
4.4.3.1	GPU Execution and Architectural Model . . . . .	72
4.4.3.2	Real-time Multiscale SRBFs . . . . .	74

4.4.4	Peak Detection for High-Frequency Lighting . . . . .	77
4.5	Results . . . . .	78
4.6	Discussion . . . . .	81
4.7	Conclusion and Future Work . . . . .	85
<b>5</b>	<b>Use Case: Relighting of Virtual Objects</b>	<b>87</b>
5.1	Augmented Reality Application . . . . .	89
5.2	Related Work . . . . .	91
5.3	Relighting in Augmented Reality . . . . .	92
5.3.1	Omnidirectional Video Capture and Rendering . . . . .	92
5.3.2	Camera Tracking using Structure-from-Motion . . . . .	93
5.3.3	Real-Time Augmented Rendering . . . . .	96
5.4	Results . . . . .	97
5.5	Conclusion . . . . .	99
<b>6</b>	<b>Use Case: Relighting of Real Objects</b>	<b>103</b>
6.1	Related Work . . . . .	104
6.2	Inverse Rendering Application . . . . .	106
6.2.1	Background . . . . .	107
6.2.2	Hierarchical Refinement and High-order Wavelets . . . . .	108
6.2.3	SRBF Triple Product in Inverse Rendering . . . . .	110
6.2.4	Near-Field Lighting . . . . .	113
6.2.5	Results . . . . .	117
6.3	Intrinsic Image Decomposition Application . . . . .	121
6.3.1	Background . . . . .	121
6.3.2	SRBF Triple Product Gradients Rendering . . . . .	122
6.3.3	Results . . . . .	124
6.4	Conclusions . . . . .	125
<b>7</b>	<b>Conclusions and Future Work</b>	<b>129</b>
7.1	General Wavelet Triple Product Rendering . . . . .	130
7.2	Spherical Radial Basis Triple Product Rendering . . . . .	131
7.3	Relighting Use Cases . . . . .	132
7.4	Future Work . . . . .	133
<b>A</b>	<b>Nederlandse Samenvatting (Dutch Summary)</b>	<b>135</b>
<b>B</b>	<b>Scientific Contributions and Publications</b>	<b>139</b>
	<b>Bibliography</b>	<b>156</b>



---

## List of Figures

---

1.1	Example of computer generated imagery in Jurassic World . . . . .	3
1.2	Appearance extraction on The Statue of Liberty . . . . .	4
1.3	Relighting of a virtual object . . . . .	5
1.4	Interactivity in relighting applications . . . . .	5
1.5	State-of-the-art intrinsic image decomposition . . . . .	7
2.1	Bidirectional Reflectance Distribution Function (BRDF) . . . . .	14
2.2	Parameters of the rendering equation . . . . .	15
2.3	Precomputed radiance transfer in the pixel domain . . . . .	17
2.4	Octahedron parametrization . . . . .	17
2.5	The 2D Haar basis functions . . . . .	18
2.6	Examples of triples of 2D Haar basis functions. . . . .	19
3.1	Area distribution error of the octahedron hemisphere . . . . .	31
3.2	Environment map reconstructed with Haar wavelets compared to the smoother Daubechies wavelet . . . . .	33
3.3	Complexity of double and triple products of Haar wavelets . . . . .	36
3.4	Complexity of double and triple products of Daubechies wavelets . . . . .	37
3.5	Tensor of tripling binding coefficients for Haar and high-order wavelets . . . . .	38
3.6	Hierarchical overlapping of wavelet basis functions . . . . .	39
3.7	Tensor mirroring . . . . .	41
3.8	Wavelet sliding . . . . .	42
3.9	Quality comparison of the Elch dataset compressed in the Haar wavelet basis	44
3.10	Quality comparison of the Elch dataset compressed in the Daubechies-6 wavelet basis . . . . .	45
3.11	Quality comparison of the Lucy dataset compressed in the Haar wavelet basis	46
3.12	Quality comparison of the Lucy dataset compressed in the Daubechies-6 wavelet basis . . . . .	47

3.13 Product integral rendering with Haar, Coiflet-5, Symmlet-5 and Daubechies-8 wavelets . . . . .	49
4.1 Rotation operator of SRBFs . . . . .	56
4.2 Dynamic materials . . . . .	60
4.3 Prerendering of a voxel volume . . . . .	62
4.4 Cone tracing . . . . .	64
4.5 Mapping of a visibility cone to a SRBF . . . . .	65
4.6 Adaptive subsampling of visibility cones . . . . .	66
4.7 Accuracy of the visibility function using cone tracing . . . . .	67
4.8 Healpix distribution scheme . . . . .	69
4.9 Residual transform of environment map to a SRBF representation . . . . .	70
4.10 Preprocessing of overlapping lighting SRBFs . . . . .	73
4.11 GPU memory layout . . . . .	74
4.12 Example of residual transform from environment map to SRBFs . . . . .	78
4.13 Peak detection . . . . .	79
4.14 Dynamic visibility . . . . .	80
4.15 Comparison with and without peak detection in a rendered result . . . . .	80
4.16 Dynamic lighting . . . . .	82
4.17 Rendering with Lambertian and Phong BRDFs . . . . .	83
4.18 Rendering with Cook-Torrance BRDF . . . . .	84
4.19 Rotating the environment lighting during rendering . . . . .	86
5.1 Real-time augmented rendering of virtual objects . . . . .	88
5.2 Illustration of real-time augmented rendering of virtual objects in an omnidirectional environment . . . . .	90
5.3 Capture of omnidirectional video . . . . .	93
5.4 Rendering of omnidirectional content . . . . .	94
5.5 Reconstruction of the poses . . . . .	95
5.6 Reconstruction of the trajectory with structure-from-motion . . . . .	96
5.7 Example renderings with different material properties . . . . .	98
5.8 Augmented reality of a sphere dataset with realistic lighting effects . . . . .	99
5.9 Casting shadows on the real world . . . . .	100
5.10 Augmented reality of a user-controlled vehicle in an urban environment with realistic lighting effects . . . . .	101
6.1 Inverse rendering with hierarchical refinement and high-order wavelets . . . . .	109
6.2 Inverse rendering and relighting of a temporal face dataset . . . . .	110
6.3 Overview of the inverse rendering framework using SRBFs . . . . .	113
6.4 Inverse rendering with near-field lighting . . . . .	116
6.5 Inverse rendering of the SurreyFace dataset using SRBFs . . . . .	118

---

6.6 Relighting of the SurreyFace dataset using SRBFs . . . . .	119
6.7 Intrinsic image decomposition on different input images . . . . .	126
6.8 Relighting on the Paper dataset . . . . .	127
6.9 Relighting on the Shorts dataset . . . . .	128
B.1 Relighting Demonstrator at SCENE 2014 . . . . .	141
B.2 Demonstrator at AIVIE 2014 . . . . .	141
B.3 Live demonstrator at HiViz 2016 Lab opening . . . . .	142



---

## List of Tables

---

3.1	Tensor sparseness for different mixtures of wavelet bases at different resolutions	43
4.1	Comparison of state-of-the-art SRBF rendering	59
4.2	Overlapping lighting SRBFs	72
4.3	Influence of scene complexity on performance of SRBF triple product rendering	81
6.1	Comparison: Haar wavelets (Haber et al.) versus SRBFs (ours) in inverse rendering	120



# **Chapter 1**

---

## **Introduction**

<b>1.1</b>	<b>Problem Statement .....</b>	<b>6</b>
<b>1.2</b>	<b>Contributions.....</b>	<b>9</b>
<b>1.3</b>	<b>Dissertation Overview.....</b>	<b>10</b>

In the last decades, the gap between computer graphics representations and image-based representations is narrowing down and both fields are more and more converging to each other. Computer generated imagery is ever-present in the movie industry. Hyper-realistic renderings of virtual objects in real sequences are common. The physical simulation of light is a well-studied domain and their algorithms for photorealistic rendering are effective in fields like engineering, architectural design, movie industry, game industry, etc. On the other hand, captured images are also used in computer graphics applications as a representation to render real objects in 3D. Examples are image-based rendering and light field rendering [Levoy and Hanrahan, 1996; Buehler et al., 2001], where multiple images of a real object in combination with a proxy geometry are used to realistically render novel viewpoints of the object.

The movie industry, game industry and possibly multiple other application domains would benefit a lot from the gap narrowing down even further. If computer graphics concepts like the realistic simulation and propagation of light can be applied to objects in real images or the environment lighting of real scenes can be applied to computer generated imagery, the amount of post-production time will decrease drastically. Now, unnecessary amounts of man-months are spent to change lighting and material properties of captured sequences. Besides the change of light in a captured sequence, it would also be beneficial if the extracted objects can be put into different environments with new lighting conditions. For example, a photographed building merged into a live action movie or the possibility to extract an object (e.g. The Statue of Liberty) out of multiple images and reuse it in any game or augmented reality application. In addition, the same technology can be used to photorealistically insert virtual objects in real captured environments. If it is possible to capture the lighting conditions of a real scene, then, in principle, the extracted lighting can be applied to any computer generated object, preferably in real-time. Such novel technologies will open the door to many new immersive applications.

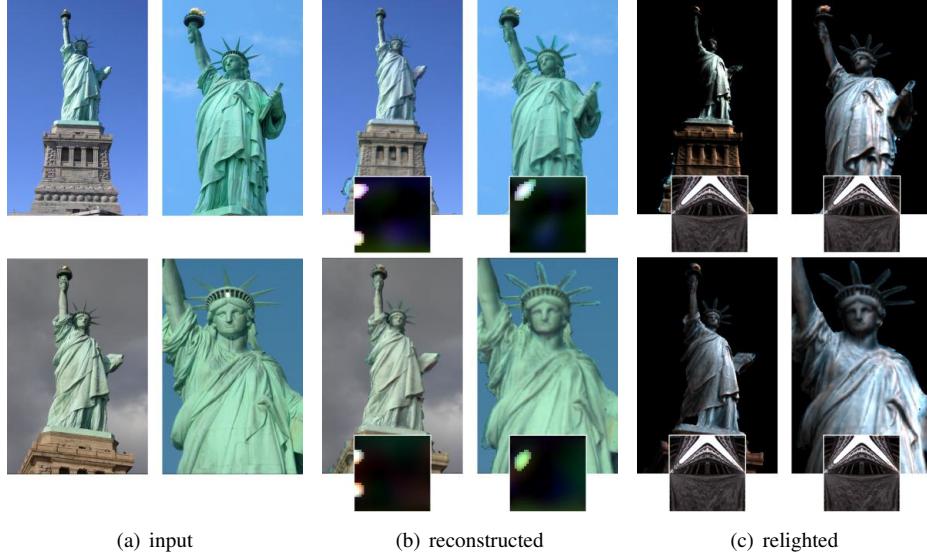
In computer generated imagery, the world is meticulously designed in 3D for all its characteristics. For example, the shape is modeled in a triangle mesh representation and each surface point is assigned a material property in the form of a *bidirectional reflectance distribution function* (BRDF). Any required light source is manually positioned and its direction is set. The computer generated imagery is obtained by applying a state-of-the-art photorealistic renderer on the modeled scene (Figure 1.1). All this modeled information of shape, materials and lighting is not available when working directly on photographs. For example, inserting a virtual object in a photograph would require the actual lighting conditions of the scene where the photograph was taken. In addition, changing the outlook of a real object in a photograph, not only requires knowledge of the lighting conditions, but also requires an algorithm to extract the shape and material characteristics of the object. This field of research is called *appearance extraction* or *inverse rendering*. Once all the required parameters are extracted, the existing lighting conditions of the extracted object can be reversed and then



**Figure 1.1:** Example of computer generated imagery in *Jurassic World*. The sequence is meticulously modeled up until the degree of muscle structures as you can see on the left. The modeled scene is rendered and composited into the photorealistic end result on the right. Images courtesy of Imagine Engine [Imagine Engine, 2016].

rendered under different lighting conditions. This action is called *relighting*. An example of relighting of The Statue of Liberty is shown in Figure 1.2.

Besides relighting of real objects, it also applies to computer generated scenes. Imagine an augmented reality application that merges the virtual world with the real world. The idea is to insert new virtual objects on top of the real world where the real world is perfectly superimposed by a virtual layer. Most augmented reality applications are able to interactively insert new objects, however, they often lack a photorealistic simulation of light. This is mainly due to the fact that the render has no idea of what lighting information to use. As a result, the computer generated objects are rendered using flat shading or using tedious offline rendering where the lighting conditions are manually set by the user. Furthermore, augmented reality applications are becoming more and more important with upcoming future technologies like the Oculus Rift [VR, 2012], Microsoft Hololens [Microsoft, 2016], the Samsung Gear VR [Samsung, 2016] and the HoloVizio displays [Holografika, 2016]. They are all designed to increase the immersive experience of users and achieve this by adding view-dependent information. The Oculus Rift and the Samsung gear are head mounted virtual reality displays that change the content based on the direction the user is looking at. The Microsoft Hololens is an augmented reality head mounted display that automatically adds an extra virtual layer that perfectly coincides with the real world and the HoloVizio is an autostereoscopic display that sends out hundreds of images in different directions, simulating a holographic experi-



**Figure 1.2: Appearance extraction on The Statue of Liberty.** (a) A set of input images are used to estimate information on shape, materials and illumination. The estimates can be used to reconstruct The Statue of Liberty under (b) the same—estimated—lighting conditions or (c) under entire different Uffizi environment lighting conditions. The inset images depict respectively the reconstructed environment lighting map and the Uffizi lighting map. Courtesy of Haber et al. [Haber et al., 2009; Haber, 2015].

ence. Since the light propagation in a scene is calculated based on the material and lighting properties as well as the view direction of the user, it is also an important view-dependent effect. If we want to increase the immersive experience of users even more, accurate and photorealistic lighting is essential. For the augmented reality application, optimally we want to use the lighting conditions of the real scene and directly apply them on the virtual objects. We call this the *relighting of virtual objects*. An example of how relighting of the virtual world can be applied in the movie industry is shown in Figure 1.3. Another interesting application domain is real estate. Nowadays, real estate agents make it possible already for customers to have a virtual walkthrough of a house on a website or in a head mounted display, by scanning in the entire house. Relighting algorithms will make it possible to layer the scanned house with, for example, new furniture and an entire new interior design of the house can be previewed by the customer.

A key challenge that both the relighting of real objects, as well as the relighting of virtual objects share, is interactivity. The immersive experience of users will only be satisfied if real-time interaction with the rendered scene is possible. The user should be able to change



**Figure 1.3:** Example of relighting of a virtual object. The environment lighting of a movie set can be captured using an omnidirectional camera. The environment lighting can then be used in a photorealistic renderer to apply the same lighting conditions to the computer generated object. In addition, real-time preview renderings can be shown to the director on-set.



**Figure 1.4:** Interactivity in relighting applications is key to increase the immersive experience of users. Time consuming offline algorithms will allow for high quality and photorealistic renderings, but they lack dynamic interaction with the user. In such applications, the user will not be able to change the lighting conditions or influence the reflections on the object. A real-time and fully dynamic photorealistic renderer, on the other hand, will allow for an entire new immersive experience where the user can cast light directly onto the virtual object or where the user sees his or her reflection in a rendered scene.

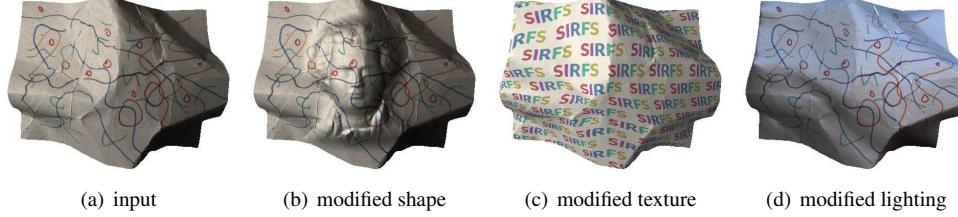
lighting conditions and material properties dynamically and instantly see the impact on the result in real-time. Furthermore extra interaction should be possible by capturing live environment lighting information with an omnidirectional camera, for example the Point Grey Ladybug [PointGrey, 2014]. Imagine an application where the user can see himself or herself reflected in the rendered scene, or the user can cast the light of a real light source onto the scene where the reflections and shadows are changed accordingly. A mockup of such application is shown in Figure 1.4. In the case of an appearance extraction algorithm, real-time previews of the current estimates of materials properties, shape and lighting can be used by the user as feedback to guide the optimization process to the correct outcome.

## 1.1 Problem Statement

Relighting is a fundamental research problem that has taken some major steps forward in the last couple of years. It tries to solve some important ambiguous problems. Basically all the information of the scene like lighting, shape and materials are integrated into a collection of image pixels. Each pixel shares information of shape, reflectiveness, diffuse inter-reflections, environment lighting, etc. Out of the single pixels alone, an appearance extraction technique desires to reconstruct information of shape, materials and lighting. It is apparent that this introduces some big challenges.

One known challenge that all appearance extraction techniques share, is the texture-illumination ambiguity. To illustrate this, imagine a photograph of a red object. Is this object perceived as red because it is a white colored object under red lighting conditions or a red colored object under white lighting conditions? Often multiple outcomes are possible and the appearance extraction technique needs to find the most plausible solution. This is generally due to under-sampling of the problem. A solution would be to add more information to the system by including more input images with different lighting conditions. Other state-of-the-art techniques have tackled this problem by adding more regularization on the optimization problem [Haber, 2015]. A related technique to inverse rendering is the *intrinsic images* problem. Here, single images are decomposed into shape, albedo, reflectance and a basic representation of lighting. State-of-the-art techniques in intrinsic images are able to reconstruct all factors quite well. This is illustrated in Figure 1.5. Barron and Malik [Barron and Malik, 2011, 2012, 2015] try to tackle the texture-illumination problem by defining priors for each of the factors. For example, they constrain the estimation of the albedo to a piecewise constant smooth function. Most of the priors are built using basic learning of a multi-variate Gaussian distribution to learn the characteristics of a specific factor. They achieve a relatively good separation of shape, albedo and reflectance, but the texture-illumination problem is far from completely solved. This is partly due to the fact that they only support low-frequency lighting using a spherical harmonics representation and no BRDF models.

Besides high level ambiguity problems, a second challenge is the correct simulation of light propagation, which is also crucial in relighting techniques. This is the part of relighting where current techniques are often overconstrained. It is not feasible and even intractable to simulate all global illumination effects during the appearance extraction phase. It takes hundreds of iterations and each iteration needs to evaluate with a rendering step. That is why more efficient methods for rendering were introduced in the field of relighting using *precomputed radiance transfer* [Nimeroff et al., 1995; Sloan et al., 2002, 2003]. Yu et al. [Yu et al., 2006a,b] used spherical harmonics rendering and precomputed radiance transfer to extract low-frequency distant lighting and a basic Gaussian filtered mirrored BRDF model. Later, Haber et al. [Haber et al., 2009] introduced a very similar technique to extract distant illumination and BRDF, but now allows for high-frequency lighting. This is achieved by



**Figure 1.5: State-of-the-art intrinsic image decomposition. Out of (a) a single input image, the intrinsic shape, albedo and illumination are estimated. The estimates are used to reconstruct the image or modify (b) the shape, (c) the albedo or (d) the lighting. Images courtesy of Barron and Malik [Barron and Malik, 2015].**

modeling the data in the Haar wavelet domain. Ng. et al. [Ng et al., 2004] researched how the product integral in precomputed radiance transfer can be evaluated in a Haar wavelet basis. Haar wavelets have a good compression performance and allow for a fast product integral evaluation. However, in most appearance extraction applications, the rendering phase still remains a bottleneck and can take up until 90% of the execution time. This is due to time consuming preprocessing of the estimates, as well as a slow rotation operator. A new efficient representation should be able to remove this bottleneck and thus to drastically decrease the execution time of an appearance extraction algorithm. Furthermore, it will allow for interactive preview rendering and will help to increase the immersive experience of real-time relighting applications.

This dissertation will mainly focus on the second challenge. We will attempt to find a new and possibly better representation to simulate the light transport in relighting applications. We argue that a better underlying representation will allow for a better simulation of the light transport, with improved global illumination effects. In addition, a representation that is able to efficiently evaluate the rendering equation will also allow for more performant and interactive relighting algorithms. A first requirement for the representation is to keep the advantages of the previous Haar wavelet basis. For example, all-frequency lighting is essential for high-quality reflections. In addition, the basis representation should be tailored to the signals. A smooth signal is better represented with a smooth basis representation. Furthermore, an efficient transform to the basis representation as well as an efficient product integral evaluation are beneficial if we want to keep the time complexity of the appearance extraction algorithms low. A real-time rendering phase is also favorable for other relighting applications. Augmented reality algorithms will only be interactive if the rendering of the inserted object can be done in real-time.

The first improvement in this dissertation is the creation of a general triple product theorem for all wavelets. The use of wavelets in triple product rendering has greatly influenced the

quality of appearance extraction techniques since it allowed for all-frequency lighting. Unfortunately, the current techniques are limited to the piecewise constant Haar wavelet basis. The visibility factor is a piecewise constant function, for which Haar wavelets are ideally suited. However, we argue that the lighting environment map and certainly the BRDF (bidirectional reflectance distribution function) exhibit, profoundly more smoothness. These factors are better represented with a smoother wavelet, for example the Daubechies wavelet. In contrast to previous methods, our proposed algorithm is able to calculate the triple product integral with a heterogeneous mix of wavelet bases, where each factor is expressed in a basis specifically tailored to the signal characteristics.

Then we will focus on the interactivity and time complexity of current relighting algorithms. Triple product rendering using wavelets is ideal for applications that require good compression performance. They have proven their worth in appearance extraction algorithms, as shown by Haber et al. [Haber et al., 2009]. However, the evaluation of the triple product rendering integral using wavelets is far from real-time and lacks flexibility. We propose the use of spherical radial basis functions (SRBFs). They eliminate most of the constraints of the wavelet approach, but can still represent all-frequency data. We mainly focus on the interactivity without putting any constraints on the factors. We want the shape, materials and illumination to be fully dynamic. As a result, the time complexity of existing appearance extraction approaches will drop from multiple hours to only a couple of minutes and even more importantly, the relighting and change of the different parameters can now be done in real-time.

Last, we will show how our improved basis representation for light transport behaves in different relighting use cases. For example, we will demonstrate an augmented reality application that uses our spherical radial basis triple product renderer to apply relighting of virtual objects which are inserted into an omnidirectional video. We will explain how the frames of an omnidirectional video can be used as environment lighting for rendering. Besides relighting of virtual objects, we will also integrate our improved representation in different appearance extraction algorithms. First, we will test on the state-of-the-art inverse rendering technique of Haber et al. [Haber et al., 2009]. Here we show how our smooth high-order wavelets triple product rendering can help to design a better regularization method for a hierarchical optimization algorithm. We also show the necessary algorithms to utilize our spherical radial basis rendering framework instead of the original Haar wavelets and how it will speed up the estimation of the unknowns. In a second appearance extraction use case, we will show how our improved triple product renderer can be used in an intrinsic image decomposition problem. For this, we implement the state-of-the-art intrinsic image decomposition technique of Barron and Malik [Barron and Malik, 2015]. We show results of how our all-frequency SRBF triple product renderer performs, compared to their low-frequency spherical harmonics implementation.

## 1.2 Contributions

The main contributions are:

1. The development of a general triple product theorem, which is not only limited to Haar wavelets, but also supports high-order wavelets. We developed a novel and efficient algorithm to calculate the tensor of binding coefficients of n-product integrals of a wide range of wavelet basis functions. Mixing of various basis types is supported. This allows for an efficient forward renderer where the underlying basis representations for visibility, materials and lighting are best tailored to the signal, with optimal compression. This dissertation focuses on double and triple coefficients, but the approach can easily be extended to quadruple or higher product integrals. We show that it is important to use a basis representation for visibility, materials and lighting that is best tailored to the signal. This will result in an optimal compression, but also allows for a better smoothness constraint in an inverse rendering framework. Furthermore we show that the signals are best represented in the original domain. Signal data expressed in the spherical domain should be represented with spherical basis functions.
2. We introduce an all-frequency precomputed radiance transfer relighting framework based on spherical radial basis functions. The main contribution is that our framework is able to change any of the three factors dynamically, resulting in a relighting approach that supports accurate and complex all-frequency lighting and where either illumination, materials and geometry can be altered in real-time.
3. We implement relighting for virtual objects. We show this in an augmented reality application where virtual objects are merged into an omnidirectional environment, using the lighting conditions of the real environment.
4. We show how better and more efficient underlying representations will influence the state-of-the-art relighting of real objects. First, we introduce our dynamic SRBF triple product rendering technique in the inverse rendering approach of Haber et al. [Haber et al., 2009]. It allows for per pixel based rendering and reduces the time complexity from multiple hours to only a couple of minutes. Secondly, we also used our SRBF representation in the intrinsic image decomposition of Barron and Malik [Barron and Malik, 2015]. Lastly, we experimented with near-field lighting effects by splitting the rendering integral into near-field and distant lighting, in contrast to current state-of-the-art techniques that only support distant lighting.

### 1.3 Dissertation Overview

The dissertation is organized as followed:

Chapter 2 gives a background on the basic concepts of relighting. First, we go more into detail on light transport simulation. Then, we explain why a correct simulation of light is essential in a relighting framework. We give an overview of current state-of-the-art representations within relighting and how they are used to extract the three factors in appearance extraction algorithms.

In Chapter 3 we present our general triple product theorem and how to use high-order wavelets in triple product integral rendering. We show how the binding coefficients are calculated and applied in the rendering. Then, we show how important it is to tailor the representation to the signal. Furthermore, we show our findings on the fact that it is desirable to keep the factors in the spherical domain.

Chapter 4 introduces another representation for triple product rendering. It shows how a spherical radial basis representation outperforms a wavelet representation. We show how we designed our framework to make materials, lighting and geometry fully dynamic.

Chapter 5 and Chapter 6 discuss the advantages of using this new way of rendering in different applications and use cases for respectively the relighting of virtual objects and the relighting of real objects. We show the importance of an efficient, interactive and high-quality relighting of virtual objects in the context of an augmented reality application. In addition, we show different use cases for our novel representation in the context of relighting of real objects. Current existing techniques for inverse rendering and intrinsic images are adapted to work with our representations and we discuss how such appearance extraction techniques are improved when the forward rendering itself is improved.

Conclusions of the dissertation are given in Chapter 7. We finish with a Dutch summary and an overview of my publications and other dissemination.

# Chapter 2

---

## Background on Relighting

<b>2.1</b>	<b>Simulation of Light</b>	<b>12</b>
2.1.1	Global Illumination	12
2.1.2	The Rendering Equation	12
2.1.3	Evaluating the Rendering Equation	13
<b>2.2</b>	<b>Precomputed Radiance Transfer</b>	<b>15</b>
2.2.1	Spherical Harmonics	17
2.2.2	Wavelets	17
2.2.3	Spherical Radial Basis Functions	19
<b>2.3</b>	<b>Relighting Applications</b>	<b>20</b>

This chapter discusses in more detail how current relighting algorithms work. A key part of a relighting application is the simulation of light propagation in a scene. The first section gives more background information on how light is simulated in general. Then, we explain the different underlying representations that can be used for an efficient simulation. Once the different concepts of forward rendering are clear, we provide more detail on how they are used in relighting algorithms.

## 2.1 Simulation of Light

### 2.1.1 Global Illumination

*Global illumination* [Dutre et al., 2006] is a term used to describe all the algorithms that add more photorealistic lighting effects to computer generated imagery. The goal is to create realistic images with plausible lighting effects. A global illumination algorithm does not only address direct lighting effects where light is directly coming from a light source, but also indirect lighting effects, including specular inter-reflections, as well as diffuse inter-reflections (color bleeding). These algorithms have drastically increased the visual quality of photorealistic rendering. To achieve such quality, the rendering requires a perfect simulation of light transport at each point in the scene by taking into account all the incident light over the entire hemisphere and the view-dependent reflection of this light. The simulation of light at each point  $x$  in the scene over the hemisphere  $\Omega$  is mathematically formulated by the *rendering equation*.

### 2.1.2 The Rendering Equation

The rendering equation was first introduced by Kajiya [Kajiya, 1986] and Immel et al. [Immel et al., 1986]. It defines the equilibrium radiance  $L$  exiting a surface point  $x$  of the scene in the direction of  $\omega_o$  as the sum of the emitted radiance  $L_e$  and reflected radiance  $L_r$ :

$$L(x \rightarrow \omega_o) = L_e(x \rightarrow \omega_o) + L_r(x \rightarrow \omega_o) \quad (2.1)$$

This formulation assumes the absence of participating media. The emitted radiance  $L_e$  is the amount of light that is exiting directly from the surface itself. The reflected radiance  $L_r(x \rightarrow \omega_o)$  at a surface point  $x$  in the direction  $\omega_o$  is the integral for directions of the incoming light  $L_r(x \leftarrow \omega_i)$  multiplied by the surface reflectance properties and a cosine term of the incident angle and normal  $n$ :

$$L_r(x \rightarrow \omega_o) = \int_{\Omega} L_r(x \leftarrow \omega_i) \rho(x, \omega_r, \omega_o) (\omega_i \cdot n(x)) d\omega_i \quad (2.2)$$

The reflectance property of the surface is modeled using a *bidirectional reflectance distribution function (BRDF)*, denoted as  $\rho(x, \omega_i, \omega_o)$ . The BRDF is a function that relates how all

incident irradiance values from direction  $\omega_i$  are altered and reflected into all outgoing directions  $\omega_o$  for each surface point  $x$ . It defines which part and how light of the hemisphere is reflected towards the outgoing direction  $\omega_o$ . The relation of the parameters are illustrated in Figure 2.1(a). Different types of BRDFs are possible. If an incident illumination ray can be scattered over the entire hemisphere, we refer to a pure diffuse BRDF. On the other hand, if the incident illumination ray is always reflected into its ideal reflection direction, we speak of a purely specular BRDF. Most surfaces are glossy and have a specific reflectance behavior. Different BRDF types are depicted in Figure 2.1(b) to 2.1(d).

The integral of the reflected radiance  $L_r$  is often split into a direct and an indirect component. The direct component evaluates the incident radiance directly from the light sources  $L_e(x \leftarrow \omega_i)$  and the indirect component describes the light  $L_i(x \leftarrow \omega_i)$  that arrives after bouncing off other surfaces in the scene [Dutre et al., 2006].

$$L_r(x \rightarrow \omega_o) = L_{direct} + L_{indirect} \quad (2.3a)$$

$$L_{direct} = \int_{\Omega} L_e(x \leftarrow \omega_i) \rho(x, \omega_i, \omega_o) (\omega_i \cdot n(x)) d\omega_i \quad (2.3b)$$

$$L_{indirect} = \int_{\Omega} L_i(x \leftarrow \omega_i) \rho(x, \omega_i, \omega_o) (\omega_i \cdot n(x)) d\omega_i \quad (2.3c)$$

The reflected light can be further simplified by incorporating the cosine term for incident illumination from  $\omega_i$  and surface normal  $n(x)$  in the BRDF, now denoted by  $\tilde{\rho}$  and by separating the incident illumination into an illumination without occlusions  $\tilde{L}$  and a visibility term  $V$ .

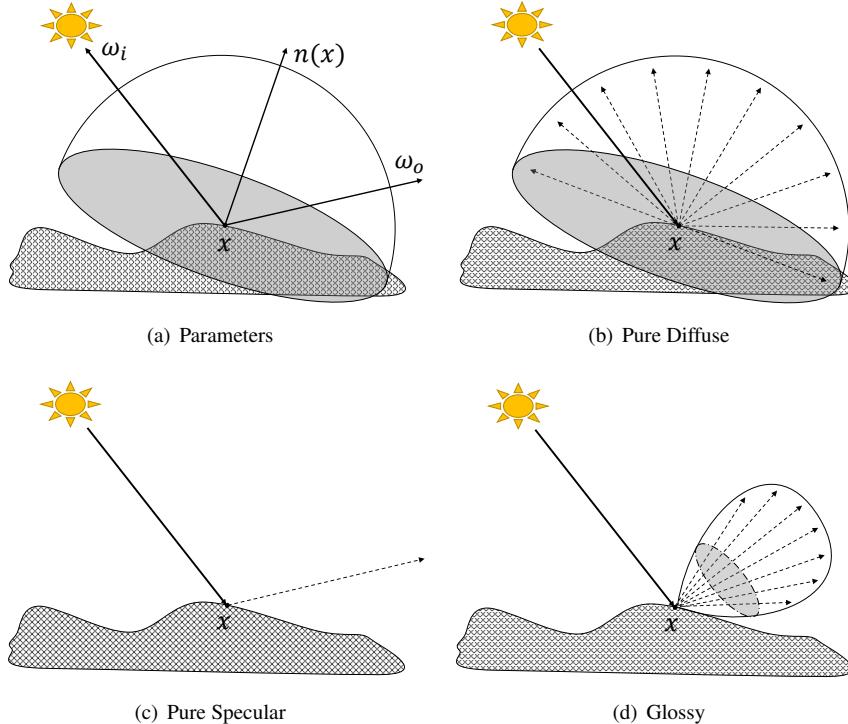
$$L_r(x \rightarrow \omega_o) = \int_{\Omega} V(x, \omega_i) \tilde{L}(x, \omega_i) \tilde{\rho}(x, \omega_i, \omega_o) d\omega_i \quad (2.4)$$

$V$  is the hemispherical visibility term evaluated around  $x$  and  $\tilde{L}$  is the incident illumination rotated in the local frame of  $x$ , defined by the normal  $n$ . The components are illustrated in Figure 2.2, where each of the factors affect the appearance of the rendered result.

### 2.1.3 Evaluating the Rendering Equation

Realistic rendering and thus integrating the rendering equation has been broadly researched in the past centuries. A lot of different approaches have been proposed. There are two popular groups of algorithms that try to evaluate the rendering equation: *ray tracing* and *radiosity*.

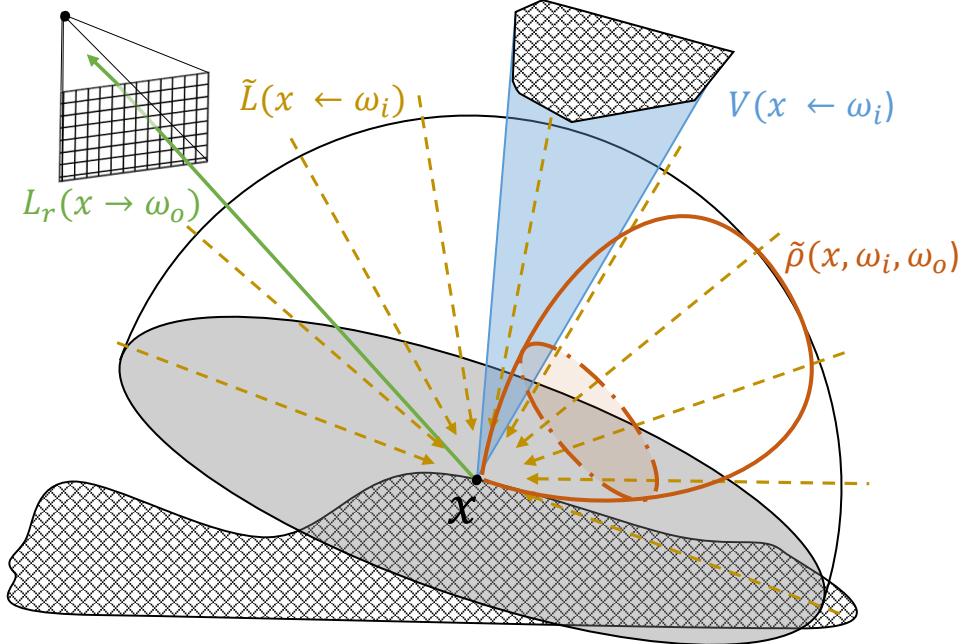
Ray tracing algorithms calculate a single color for each pixel of the rendered image. This is achieved by shooting a ray from the camera through the pixel into the scene. The path of the ray is traced throughout the scene and for each hit and bounce the influence of the lighting, materials and geometry is calculated. Originally, global illumination effects were not included in the calculations. Later, the development of stochastic ray tracing or Monte Carlo ray tracing had a great impact, since it became possible to compute high quality and photo realistic images including all global illumination effects.



**Figure 2.1: Bidirectional Reflectance Distribution Function (BRDF).** (a) The parameters of the BRDF where for each surface point  $x$ , the incident direction  $\omega_i$  is related to any outgoing direction  $\omega_o$ . (b) Example of a diffuse BRDF where the incident light is scattered over the entire hemisphere. (c) Example of an entirely specular BRDF where the incident light is reflected to its ideal reflection direction. (d) Example of a glossy BRDF where the light is scattered in a small lobe of directions centered around the ideal reflection.

A radiosity algorithm, on the other hand, does not calculate the light transfer in image space, but is a scene based method. The scene is subdivided in small patches. For each patch, the radiometric value is calculated and stored in the scene. Once all the patches are precalculated, the scene can be rendered from a chosen viewpoint by rendering out the patches. A radiosity method is relatively easy to implement and allows for global illumination effects, since for each patch not only illumination arriving from the light sources is included, but also reflected light from other objects in the scene.

In addition, there exist some hybrid multi-pass methods that try to exploit the advantages of both ray tracing and radiosity algorithms. For example, inspired by radiosity, some techniques first construct a photon map containing all the irradiance values for each point in



**Figure 2.2: Parameters of the rendering equation.** For each pixel in the image plane, a radiance value is calculated at a point  $x$  in the direction of the viewer  $\omega_o$  through the pixel. The lighting  $\tilde{L}$  describes the incident irradiance values over the entire hemisphere. The visibility factor  $V$  describes what part of the hemisphere is occluded by other objects in the scene. The BRDF  $\tilde{\rho}$  describes what part of the hemisphere is actually reflected in the outgoing direction. The integration of these three factors over the entire hemisphere results in a photorealistic radiance value.

the scene. Then, in a second pass, the photon map is integrated in a stochastic ray tracing algorithm where it serves as an efficient alternative for diffuse inter-reflection calculations.

State-of-the-art ray tracing, radiosity or hybrid multi-pass methods allow for photorealistic renderings with a spectacular visual realism. However, one big disadvantage they share is time performance. In general, such algorithms take a lot of time to generate a rendered result. For more interactive applications, we need a more efficient algorithm to evaluate the rendering equation.

## 2.2 Precomputed Radiance Transfer

A more efficient way to render a scene with realistic lighting effects is to preprocess parts of the rendering equation. Precomputed radiance transfer [Nimeroff et al., 1995; Sloan et al.,

2002, 2003; Ng et al., 2004] is an algorithm to efficiently render scenes with complex illumination. The idea is to preprocess and compress the incident illumination, 6D BRDF and visibility data for each surface position  $x$ . Naive sampling of the hemispherical data of the rendering equation for each position is impractical. To make the calculations more manageable, the light transport is now precalculated in a factored form. For each point  $x$ , viewed from a direction  $\omega_o$ , the three factors are represented in a 2D parametrization of the spherical domain.  $\tilde{\rho}$  is denoted as a 2D slice out of the 6D BRDF. The data of the three factors are represented with different basis functions.

$$\begin{aligned} V(\omega) &= \sum_i V_i \Psi_i(\omega) \\ \tilde{L}(\omega) &= \sum_j \tilde{L}_j \Psi_j(\omega) \\ \tilde{\rho}(\omega) &= \sum_k \tilde{\rho}_k \Psi_k(\omega) \end{aligned} \quad (2.5)$$

where  $\Psi$  is an appropriate basis on the hemisphere. We can now rewrite the rendering equation as shown in Equation 2.4 in function of the new basis  $\Psi$ .

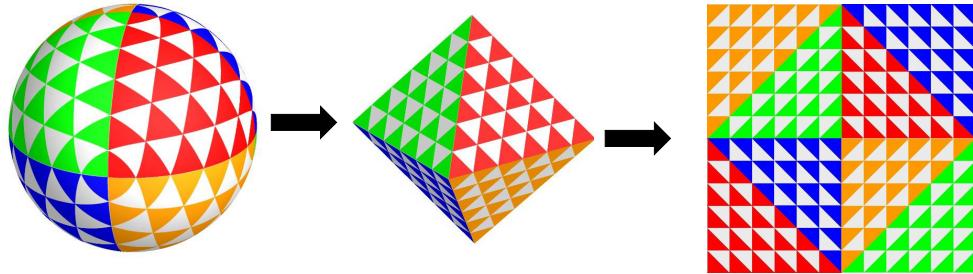
$$\begin{aligned} L_r(x \rightarrow \omega_o) &= \int_{\Omega} V(x, \omega_i) \tilde{L}(x, \omega_i) \tilde{\rho}(x, \omega_i, \omega_o) d\omega_i \\ &= \int_{\Omega} \sum_i V_i \Psi_i(\omega) \sum_j \tilde{L}_j \Psi_j(\omega) \sum_k \tilde{\rho}_k(x) \Psi_k d\omega_i \\ &= \sum_i \sum_j \sum_k V_i \tilde{L}_j \tilde{\rho}_k(\omega_o) \int_{\Omega} \Psi_i(\omega) \Psi_j(\omega) \Psi_k(\omega) d\omega \\ &= \sum_i \sum_j \sum_k V_i \tilde{L}_j \tilde{\rho}_k(\omega_o) C_{ijk} \end{aligned} \quad (2.6)$$

where  $C_{ijk}$  is the triple product integral of the three basis functions. The values of  $C_{ijk}$  for different permutations of  $i, j$  and  $k$  are called triple product binding coefficients.

The efficiency of this calculation is dependent on the choice of basis to represent the factors. Calculations in the pixel domain will be inefficient, as pixels cannot sparsely represent the information of the factors. Figure 2.3 shows an example of representing the factors in the 2D pixel domain. Here, the spherical data of the factors are transformed to the 2D image domain using the Praun and Hoppe parameterization method [Praun and Hoppe, 2003]. Furthermore, as we explain later, the less information necessary to store the underlying data, the less information an appearance extraction technique needs to recover. Generally, the data is projected into the spherical harmonics basis [Sloan et al., 2002], into the Haar wavelet basis [Ng et al., 2004] or more recently, into spherical radial basis functions [Tsai and Shih, 2006].



**Figure 2.3:** Precomputed radiance transfer in the pixel domain. The three factors are represented in the 2D pixel domain using 3D to 2D parametrization [Praun and Hoppe, 2003].



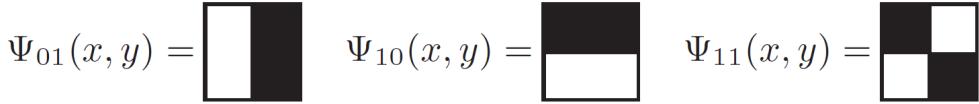
**Figure 2.4:** Octahedron parametrization to represent 3D spherical data in the 2D image domain. Courtesy of Praun and Hoppe [Praun and Hoppe, 2003].

### 2.2.1 Spherical Harmonics

Spherical harmonic basis functions were one of the first functions proposed to approximate the factors in the triple product integral [Sloan et al., 2002; Kautz et al., 2002; Sloan et al., 2003]. Spherical harmonics are an extension of the Fourier transformation to the spherical domain. They have the advantage of being a well-studied mathematical tool that can succinctly approximate low-frequency signals. Expressing detailed high-frequency effects with spherical harmonics, however, requires exponentially more coefficients. Another disadvantage is that no efficient analogue of the Fast Fourier Transform [Cooley and Tukey, 1965] exists for harmonic analysis in the spherical domain. A relatively fast alternative algorithm is known, but its time complexity is still super quadratic [Mohlenkamp, 1997]. A triple product integral theorem can also be formulated for Legendre polynomials, but due to the global support of this basis, it suffers from the same shortcomings as spherical harmonics [Gupta and Narasimhan, 2007].

### 2.2.2 Wavelets

To accurately represent high-frequency lighting details, wavelet bases can be utilized [Daubechies, 1992]. Spherical wavelets are an alternative representation for spherical



**Figure 2.5:** The 2D Haar wavelet transform consist of horizontal, vertical and diagonal basis functions.

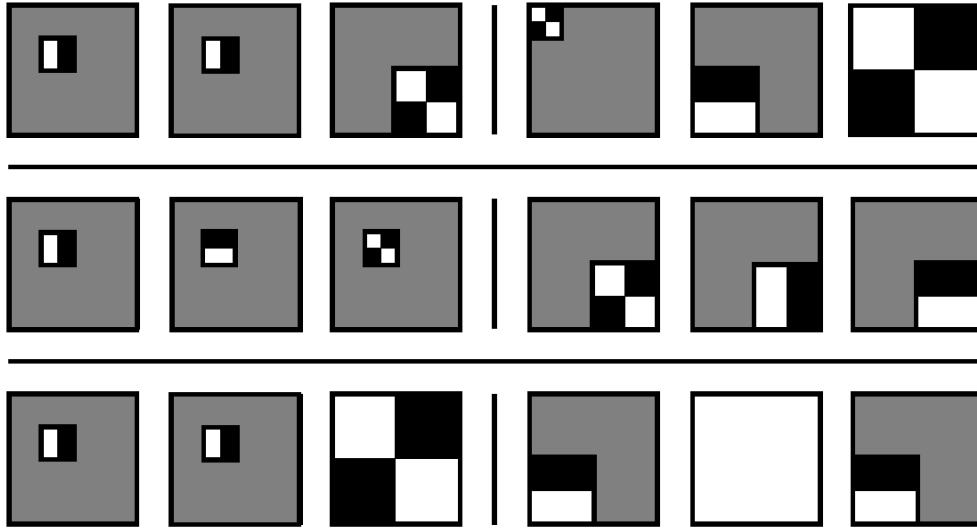
harmonics in the wavelet domain, which have demonstrated to be a compact and efficient representation [Schröder and Sweldens, 1995]. While the possibility to construct an orthogonal spherical wavelet basis with compact support and symmetry has been demonstrated [Lessig and Fiume, 2008], these wavelets are considerably more difficult to construct than their 2D counterparts. To leverage conventional 2D wavelet analysis, spherical functions are often parametrized with an area-preserving octahedron parametrization [Praun and Hoppe, 2003]. The octahedron parametrization is illustrated in Figure 2.4.

Ng. et al. [Ng et al., 2004] were the first to solve a triple product integral of three factors approximated in the Haar wavelet basis. The 2D Haar wavelet basis functions are shown in Figure 2.5. They noticed that the product integral tensor of wavelet functions is very sparse and the calculations can be categorized in a small number of cases, which they exhaustively listed in their Haar tripling coefficient theorem. They manually studied the different possible wavelet combinations and their outcome. Only a fraction of the wavelet combinations on different levels resulted in a non-zero integral, and only these particular cases are required to correctly evaluate the triple product integral. They used an analytical approach to formulate the very limited set of cases that arise when calculating the triple product integral of simple Haar basis functions. Ng et al. [Ng et al., 2004] realized that the different binding coefficients are very redundant when using a simple Haar wavelet basis. Their final tripling coefficient theorem has three cases, with variations for scaling and wavelet combinations.

1. All three are the scaling functions.
2. All three occupy the same wavelet square and all are different wavelet types.
3. Two are identical wavelets; the third is either the scaling function or a wavelet that overlaps at a strictly coarser level.

Examples of the different cases are shown in Figure 2.6. A recursive algorithm is given which is able to calculate only the relevant triple coefficients. Furthermore, Haar wavelets with smaller dilation factors are always fully contained in a subset of the parents support with constant value. These characteristics allow for a fast sublinear time algorithm that iterates over all non-zero coefficients by passing the constant parent sum to its children.

Haar wavelets or eigenbases are often able to represent the content more compactly. These bases, however, place their own restrictions on the data or the operations that can be per-



**Figure 2.6:** Examples of triple products of 2D Haar basis functions. White is positive one, black is negative one and gray is zero. The first row contains two examples where the triple product results in zero because not all basis functions overlap. The second row contains two examples that result in a non-zero triple product because of the second case of the theorem where all three wavelets perfectly overlap and have a different wavelet type. The last row contains two examples of a non-zero triple product because of the third case of the theorem where two are identical wavelets and the third is either the scaling function or a wavelet that overlaps at a coarser level. Courtesy of Ng et al. [Ng et al., 2004].

formed on it. Wavelets are very good at compression but lack flexibility. Wavelets are constrained to static scenes, since the visibility needs to be precomputed. Moreover, rotation is a complex operation in the wavelet domain [Wang et al., 2006]. Furthermore, their manual approach does not scale well when using smoother wavelets. In this dissertation, we therefore take a computational approach to the problem and calculate the tripling coefficients for a wide range of wavelets automatically.

### 2.2.3 Spherical Radial Basis Functions

A more recent basis to represent the three factors of the rendering—that at the same time allows for all-frequency lighting—is the spherical radial basis [Tsai and Shih, 2006; Wang et al., 2009]. One important advantage of spherical radial basis functions (SRBFs), compared to wavelets, is an efficient rotation operator. A popular spherical radial basis function is the spherical Gaussian. A spherical Gaussian  $G$  is defined as

$$G(v \cdot p, \lambda, \mu) = \mu e^{\lambda(v \cdot p - 1)} \quad (2.7)$$

where  $v \cdot p$  is the dot product between a vector  $v$  on the sphere and the center of the Gaussian,  $\lambda$  the Gaussian bandwidth and  $\mu$  the amplitude. Spherical Gaussians have several advantages:

- A multiscale hierarchy of spherical Gaussians allows for an efficient representation of both low- and high-frequency information.
- Spherical Gaussians are rotationally symmetric and therefore allow the efficient rotation of the environment map to the local vertex coordinate system by rotating the center of the Gaussian.
- BRDF functions can be approximated well with spherical Gaussians, allowing run-time evaluation of reflectance.
- The mixing of Gaussians is additive and allows for a less complex implementation, compared to spherical harmonics or wavelet theory.

Nevertheless, current state-of-the-art techniques often put constraints on the different factors. They require lots of precomputation on one or more of the factors before they are able to represent them with spherical radial basis functions. For example, the fitting process of Gaussians in the lighting environment map is often done using an optimization process which can take up to several minutes to fit the full lighting factor [Tsai and Shih, 2006; Wang et al., 2009; Iwasaki et al., 2012]. Furthermore, most of the techniques precalculate visibility and thus lack the possibility to render dynamic scenes [Wang et al., 2009].

In Chapter 4 we will explain our novel way of using spherical radial basis functions. In contrast to previous techniques, we will be able to construct and update all three factors in real-time. We will show how a residual transformation technique can be used to efficiently transform the lighting information to a spherical Gaussian representation. In addition, cone tracing together with peak-detection of powerful and high-frequency light sources will allow for interactive rendering of soft shadows.

## 2.3 Relighting Applications

In forward rendering applications, the scene is entirely modeled in 3D. All the required information on shape, materials and lighting is set by the designer. A forward rendering algorithm will use this information to render the scene as photorealistically as possible. The goal of *relighting* is to bridge the gap between the virtual and the real world. For example, a relighting algorithm wants to insert virtual objects in real environments with consistent lighting conditions of the environment itself as if the object is physically present. Another possibility is to do the exact same thing for real objects, where they are first extracted out of photographs and then placed in a different environment under different lighting conditions. In a sense, relighting is the opposite of the forward rendering. In research, *inverse rendering* and *intrinsic image decomposition* are two common approaches to achieve this.

The general idea of inverse rendering is to start from one or multiple input images and extract the parameters for shape, material and lighting properties out of the scene. More specifically, an inverse rendering technique tries to estimate the coefficients of the underlying basis of the three factors. Once the parameters are estimated, existing lighting effects can be removed and the object can be rerendered under different lighting conditions and possibly with different material properties. Solving the inverse rendering problem has many interesting application domains. For example in the movie industry, parts of the scenes can be rerendered without the need of recapturing the scene on set. Other target domains are the game industry or augmented reality applications, where real objects can be efficiently inserted, without the need of a time consuming modeling phase.

There are different kinds of techniques proposed in the research domain of inverse rendering. Most of them vary as a function of the number of unknown factors. Early techniques only estimated either for shape, lighting or materials. For example, shape-from-shading methods [Horn, 1970, 1989; Zhang et al., 1999; Durou et al., 2008] extract only geometry by making assumptions about shading with known illumination. Structure-from-motion [Triggs et al., 2000; Hartley and Zisserman, 2003] techniques, on the other hand, extract geometry by ignoring shading and illumination. Other techniques have tried to extract shape by actively adding known and coded illumination in the form of structured light. An overview of active scanning techniques is given by Salvi et al. [Salvi et al., 2004]. Reflectance information is often extracted in a controlled setup called a light-stage [Debevec et al., 2000], by densely sampling the BRDF. Other approaches have worked on sparse sets of images under known point light illumination [Marschner, 1998; Marschner et al., 1999; Lensch et al., 2003; Zickler et al., 2006] or under more complex outdoor illumination [Yu and Malik, 1998; Debevec et al., 2004; Romeiro et al., 2008]. Early attempts at illumination estimation made use of a light probe to capture environment lighting [Miller and C., 1984; Debevec, 1998]. The captured lighting was used to add synthetic objects in a scene. Schoeneman et al. [Schoeneman et al., 1993] proposed a method to identify intensities of a set of fixed light sources in a synthetic scene with known geometry and reflectance properties. An alternative approach estimated the directional distribution of the incident light from a single photograph [Marschner and Greenberg, 1997]. Here, a linear least squares system was used to estimate lighting effects and to perform relighting. Yu et al. [Yu et al., 1999] constructed an inverse global illumination model for recovering reflectance models of real scenes from photographs. Sato et al. [Sato et al., 2003] have recovered the illumination distribution of a scene from image brightness, inside shadows, cast by an object of known shape in the scene. Finally, Mei et al. [Mei et al., 2009] have developed a representation for environment lighting that combines low-frequency spherical harmonics with sparse directional light sources. The factorization does not take any local emitters into account and is therefore limited to distant lighting.

There are only a few techniques that have tried to extract shape, illumination and materials at the same time. The methods described above focused mainly on recovering either the shape,

illumination or the reflectance factor. We cannot, however, assume knowledge of either in the scene, so it is essential that a simultaneous estimation of BRDFs together with the illumination is made. Ramamoorthi et al. [Ramamoorthi and Hanrahan, 2001] provided a theoretical framework based on the factorization of the factors. It predicts under which conditions good results can be achieved and studies the well-posedness of the combined estimation problem. More recent work focused on the simultaneous estimation of these factors [Yu et al., 2006a,b; Laffont et al., 2012]. These techniques were relying on spherical harmonics rendering [Sloan et al., 2002; Kautz et al., 2002; Sloan et al., 2003], which lack the ability to store high-frequency data without exponentially increasing the number of coefficients. Haber et al. [Haber et al., 2009] are able to recover high frequencies, building further on the development of efficient triple product wavelet integrals by Ng et al. [Ng et al., 2004]. Using wavelets, it is possible to represent BRDFs and illumination data with a small number of coefficients.

A related technique to inverse rendering is the *intrinsic image decomposition* problem [Land et al., 1971; Barrow, 1978]. Here, instead of a multi-view input, only single images are decomposed into shape, albedo, reflectance and a basic representation of lighting. State-of-the-art techniques in intrinsic images are able to reconstruct all factors quite well. For example Barron and Malik [Barron and Malik, 2011, 2012, 2015] tried to tackle the texture-illumination problem by defining priors for each of the factors. For example, they constrained the estimation of the albedo to a piecewise constant smooth function. Most of the priors use basic learning of multi-variate Gaussian distributions to learn the characteristics of the factors. They managed to obtain a relatively good separation of shape, albedo and reflectance. This is also true for Chen and Koltun [Chen and Koltun, 2013]. They used a different model for representing the albedo and shading. Hachama et al. [Hachama et al., 2015] tried to improve the definition of a prior by adding color-independent information. They reduced the amount of texture-illumination ambiguity in the result. However, they assume that a good reconstruction of depth and normals are given. The aforementioned techniques are able to separate shading and illumination quite well, but the texture-illumination problem is far from solved completely. This is partly due to the fact that they only support low-frequency lighting with a spherical harmonics representation and use no BRDF models. The problem of intrinsic image estimation is made difficult because multiple optical phenomena often give rise to the same pattern of pixel values in a picture. These ambiguities can arise during the estimation of geometry [Belhumeur et al., 1999], but reflectance and lighting are also inextricably linked [Pont and te Pas, 2006]. In the absence of additional information, this problem cannot be solved [Ramamoorthi and Hanrahan, 2001]. Therefore, other constraints are usually added in the form of multi-view information [Haber et al., 2009; Lee et al., 2012; Laffont et al., 2013; Duchêne et al., 2015]. In addition to the static image estimation, some methods have been devised to work on videos [Ye et al., 2014; Imber et al., 2014; Meka et al., 2016].

More recent techniques tried to use more sophisticated machine learning techniques to estimate properties of the scene. For example, Kulkarni et al. [Kulkarni et al., 2015] used deep learning with convolutional neural networks for inverse graphics. In addition, convolutional neural networks are learned to estimate basic lighting direction and color [Augusto Dorta Marques and Walter Gonzalez Clua, 2016]. Another example is the work of Narihira et al. [Narihira et al., 2015]. Instead of relying on the physically-motivated priors or graph-based algorithms, they want to estimate the so called *direct intrinsics* by learning a convolutional neural network that predicts output albedo and shading channels. They outperform current state-of-the-art intrinsic image decomposition techniques.



# Chapter 3

---

## General Triple Product Theorem for Wavelets

<b>3.1</b>	<b>Related Work</b>	<b>27</b>
<b>3.2</b>	<b>Choosing a Suitable Wavelet Basis</b>	<b>30</b>
3.2.1	Spherical Wavelets versus 2D Wavelets	30
3.2.2	Wavelets Tailored to the Signal	33
<b>3.3</b>	<b>Wavelet Product Integral</b>	<b>34</b>
3.3.1	The Haar Tripling Coefficient Theorem	35
3.3.2	General Tripling Coefficient Theorem	36
<b>3.4</b>	<b>Tensor of Triple Product Binding Coefficients</b>	<b>40</b>
3.4.1	Naive Approach	40
3.4.2	Hierarchical Approach	40
3.4.3	Tensor Mirroring	40
3.4.4	Wavelet Sliding	41
3.4.5	Vanishing Moments	42
3.4.6	Tensor Sparseness	42
<b>3.5</b>	<b>Results and Applications</b>	<b>44</b>
<b>3.6</b>	<b>Discussion</b>	<b>46</b>
<b>3.7</b>	<b>Conclusions</b>	<b>48</b>

The introduction of Haar wavelets as an underlying basis for triple product integrals has been an important advancement since its introduction by Ng et al. [Ng et al., 2004]. The efficient evaluation of triple product integrals is ubiquitous in multiple applications, such as signal processing and rendering algorithms. Relighting applications, for example, require efficient methods for evaluating the light propagation in a scene, including realistic view-dependent materials and soft shadows under full hemispherical lighting. By using a precomputed radiance transfer approach, which breaks down the rendering equation in precomputed factors for lighting, materials and visibility, they are able to achieve efficient and high quality rendering. Evaluating the rendering equation is now reduced to solving a triple product rendering integral of the three factors. Evaluating the triple product rendering integral with Haar wavelets is an important improvement since they are able to represent all-frequency lighting effects, in contrast to the original approach as introduced by Sloan et al. [Sloan et al., 2002], where the underlying spherical harmonics basis was only able to simulate low-frequency lighting effects.

Using Haar wavelets has multiple advantages comparing to previous bases. First, Ng. et al [Ng et al., 2004] were the first to devise a solution method for evaluating the triple integral using Haar wavelets by manually identifying cases where overlapping Haar wavelets return in non-zero binding coefficients. Furthermore, they devised a fast sublinear-time Haar algorithm for evaluating the integral. A second advantage, compared to previous methods, is the ability to represent high-frequency information and still have a very good compression performance. In this dissertation, we use the term compression, as a reduction of non-zero or near-zero elements. The goal of a wavelet transformation is to represent the underlying data with only a few important coefficients. Most of the other coefficients will be zero or near-zero. By pruning the zero elements (lossless compression) or the near-zero elements (lossy compression), the data is still well preserved. A good compression performance allows for efficient rendering, because only few coefficients need to be evaluated.

However, Haar wavelets also have some important limitations. Haar wavelets excel at encoding piecewise constant signals like the visibility function, but are inadequate for compactly representing smooth signals like environment lighting or BRDF. Redundant Haar wavelet coefficients are required to allow for smooth signals. Better would be the use of high-order wavelets, which are optimal for representing and compressing smooth signals. Haar wavelet triple products are efficient, but are not really interactive. If triple product integrals can be efficiently calculated for higher-order wavelets, the reduction in coefficients will reduce the number of calculations, therefore improving performance and memory usage. Some BRDFs can be stored 5 times more compactly. Furthermore, inverse rendering algorithms would benefit from the smooth nature of high-order wavelets. Current inverse rendering algorithms rely on solving large systems of bilinear equations. Later in Chapter 6, we will show a hierarchical refinement algorithm that exploits the tree structure of the wavelet basis. By only splitting at interesting nodes in the hierarchy, large portions of less important coefficients can

be skipped. Haar wavelets can only insert piecewise constant basis functions, causing blocky artifacts in the lower levels and as a result, the lesser coefficients can be skipped. High-order wavelet inherently have a smoothness characteristic, allowing the inverse rendering technique to converge much faster to an optimal solution.

The goal of this chapter is to devise an efficient algorithm to calculate product integral binding coefficients for a heterogeneous mix of wavelet bases. Previous work (Section 3.1) has focused on simple Haar wavelets. Haar wavelets excel at encoding piecewise constant signals, but are inadequate for compactly representing smooth signals for which high-order wavelets are ideal. First, we will briefly demonstrate in Section 3.2 why it is advantageous to tailor the choice of the underlying wavelet basis to the characteristics of the signal. Then, Section 3.3 will explain in more detail why high-order smooth wavelets will improve current relighting techniques and what challenges and difficulties occur in triple product rendering when using high-order wavelets instead of the simple Haar wavelets. Our algorithm provides an efficient way to calculate the tensor of these binding coefficients, explained in Section 3.4. The algorithm exploits both the hierarchical nature and vanishing moments of the wavelet bases, as well as the sparsity and symmetry of the tensor. To conclude the chapter, we will demonstrate the effectiveness of high-order wavelets with a rendering application in Section 3.5. The smooth wavelets represent the signals more effectively and with less blockiness than Haar wavelets.

### 3.1 Related Work

Spherical harmonic basis functions were one of the first functions proposed to approximate the factors in the triple product integral [Sloan et al., 2002]. Spherical harmonics are an extension of the Fourier transformation to the spherical domain. They have the advantage of being a well-studied mathematical tool that can succinctly approximate low-frequency signals. Expressing detailed high-frequency effects with spherical harmonics, however, requires exponentially more coefficients. Another disadvantage is that no efficient analogue of the Fast Fourier Transform [Cooley and Tukey, 1965] exists for harmonic analysis in the spherical domain. A relatively fast alternative algorithm is known, but its time complexity is still superquadratic [Mohlenkamp, 1997]. A triple product integral theorem can also be formulated for Legendre polynomials, but due to the global support of this basis, it suffers from the same shortcomings as spherical harmonics [Gupta and Narasimhan, 2007]. Their graphical representation of tensor slices inspired the computational approach taken in this dissertation. Since most signals in triple product rendering integrals consist of high-frequency information, there is a fundamental need for an underlying basis representation that allows such effects.

Wavelets [Daubechies, 1988; Chui, 1992] are known to be an efficient basis representation in signal processing applications that incorporates high-frequency detail and has at the same

time an excellent compression ratio. Wavelets have already been extensively used in image processing and computer graphics applications [Meyer, 1993; Stollnitz et al., 1995; Kingsbury and Magarey, 1998]. Examples are image editing [Berman et al., 1994], compression [DeVore et al., 1992] and global illumination rendering [Christensen et al., 1995; Gortler et al., 1993]. Wavelets possess some interesting properties. A first property is that a wavelet has zero mean. More specific, for a certain wavelet  $\psi$  the following is true:

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0 \quad (3.1)$$

Furthermore, most wavelets have compact support, i.e. non-zero for only a small subset of the entire function range. Lastly, most wavelets are orthogonal, meaning that the product of two distinct wavelets is zero:

$$\int_{-\infty}^{+\infty} \psi_i(t) \psi_j(t) dt = \|\psi_i\|^2 \delta_{i,j} = \|\psi_j\|^2 \delta_{i,j} \quad (3.2)$$

where  $\delta_{i,j}$  is also known as the Kronecker delta.

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

The characteristics of all orthogonal wavelets with compact support are given by Daubechies [Daubechies, 1988]. There exist a lot of different wavelet functions, each having their own characteristics. A very popular wavelet basis is the piecewise constant Haar wavelet basis [Haar, 1910]. Its 2D generalization is often used in image processing [Beylkin et al., 1991]. Haar wavelets are excellent for encoding piecewise constant functions. More complex functions require more specialized high-order wavelets [Daubechies, 1992]. Examples are the high-order Coiflet, Daubechies, Symlet and Legendre wavelets. One advantage of a spherical harmonics representation is that the basis is defined in the spherical domain. This is not the case for most wavelets. Some experiments for building a wavelet basis in the spherical domain are conducted by Schröder and Sweldens [Schröder and Sweldens, 1995]. Ma et al. [Ma et al., 2006] used such spherical wavelets for real-time triple product rendering. While the possibility to construct an orthogonal spherical wavelet basis with compact support and symmetry has been demonstrated [Lessig and Fiume, 2008], these wavelets are considerably more difficult to construct than their 2D counterparts. Therefore, functions are often parametrized with an area-preserving parametrization [Praun and Hoppe, 2003], so that conventional 2D wavelet analysis can be leveraged.

Using wavelets as an underlying representation for triple product rendering is a challenging task. Ng. et al. [Ng et al., 2004] were the first to solve a triple product integral of three factors approximated in the Haar wavelet basis. They noticed that the product integral tensor of wavelet functions is very sparse and the calculations can be categorized in a small number of

cases, which they exhaustively list in their Haar tripling coefficient theorem. They manually studied the different possible wavelet combinations and their outcomes. Only a fraction of the wavelet combinations on different levels resulted in a non-zero integral, which is why only these particular cases need to be treated to evaluate the triple product integral.

Previous work has also developed a general technique for importance sampling products of complex functions using wavelets [Clarberg et al., 2005]. They performed on-the-fly stochastic sampling of the wavelet scaling coefficients to evaluate a double product integral of the BRDF and an environment map. They have based their sampling scheme on the characteristics of the Haar wavelet basis and only double product integrals are demonstrated. In addition, they had been preprocessing the 4D BRDFs, allowing that only sample points need to be evaluated at runtime. However, this means that the sample pattern does not adapt when either the environment map or the BRDF is dynamic. Also, the random sampling patterns they used, introduced considerable noise in the resulting images.

A generalized Haar integral coefficient theorem has been proposed for evaluating arbitrary dimensional Haar product integral coefficients [Sun and Mukherjee, 2006]. They extended the approach of Ng. et al. and created an efficient sublinear algorithm to evaluate these N-product integrals. However, similar to other previous work, they are limited to simple Haar wavelet bases.

There also exists a geometry-dependent basis for diffuse precomputed radiance transfer [Nowrouzezahrai et al., 2007]. Their basis is derived from Principal Component Analysis of the sampled transport functions at each vertex. They demonstrate double product integral capabilities and the rendering results are limited to diffuse only. Interpolation artifacts also arise due to the dependency of the basis on the geometric representation of the scene.

Later, an affine double and triple product integral theory was developed, enabling one of the product functions to be scaled and translated [Sun and Ramamoorthi, 2009]. They have demonstrated that these operations are very sparse and scale with linear complexity. This sparsity enabled them to add some of the first near-field lighting effects. In their disposition, they gave specific attention to the common Haar wavelets and relied on its non-overlapping property. They stated that an implementation for non-Haar wavelets is more expensive but that their general approach can be similarly applied to general wavelets. They do not, however, provide a solution to this problem.

Previous work has tried to exploit the fact that in areas where the visibility factor is constant, the triple product integral reduces to a double product integral [Inger et al., 2013]. Nevertheless, in the areas where a full triple product evaluation is needed, a fall back to an expensive pixel domain integral is still required. In addition, mixing of arbitrary and high-order wavelet bases is not supported.

Our work differs from previous work in that a computational approach is taken to calculate the tensor coefficients for triple product integrals. This allows the use and mixing of a wide range of high-order wavelets, as opposed to simple Haar wavelets. High-order wavelets can compactly approximate the smooth factors in the product. In the next section, the use of these high-order wavelets will be motivated.

## 3.2 Choosing a Suitable Wavelet Basis

There is no optimal choice of wavelet function for approximating each signal with an optimal quality and compression performance. Each signal has its own characteristics and based on these characteristics, one should determine which wavelet is ideal. A good choice of wavelet representation will result in an improved approximation of the signal, expressed with fewer coefficients. A good compression performance is not only beneficial for memory consumption, but it also has an influence on time performance, especially for triple product rendering applications. We have observed two main limitations, while implementing and using a triple product renderer. First, current rendering applications introduce distortion artifacts while representing the spherical signals of the rendering equation in a 2D Haar wavelet basis. Second, current rendering applications are limited to the Haar wavelet transform and lack a good wavelet transform that is tailored to the signals. Both limitations will have an influence on how we represent each signal in the rendering equation and thus also limits the compression performance and the quality of rendering.

### 3.2.1 Spherical Wavelets versus 2D Wavelets

Our first observation is that the signals of the three factors in the rendering equation are all defined in the spherical domain. However, current applications rely on a 2D wavelet representation for triple product wavelet rendering. There is no uniform one-to-one mapping from the spherical to the planar domain and using a 3D to 2D transformation will introduce distortion artifacts. To minimize the distortion, an octahedron parametrization [Praun and Hoppe, 2003] is often used (for more information on the octahedron parametrization, we refer to Section 2.2.2). However, while the distortion is minimized, not all spherical triangles on the octahedron parametrization have exactly the same surface area. An overview of the distortion error distributed over the sphere for an octahedron parametrization is given in Figure 3.1. The surface area variation can be up to a difference of 65% between the largest and smallest spherical triangle. Applying such parametrization in a rendering application will not have a great impact on the visual quality performance, but we have observed that it does have an impact on compression performance of the underlying wavelet basis.

In collaboration with Put et al. [Put et al., 2014], we investigated the effect of the spherical to planar parametrization distortion on the compression performance. We compared Haar

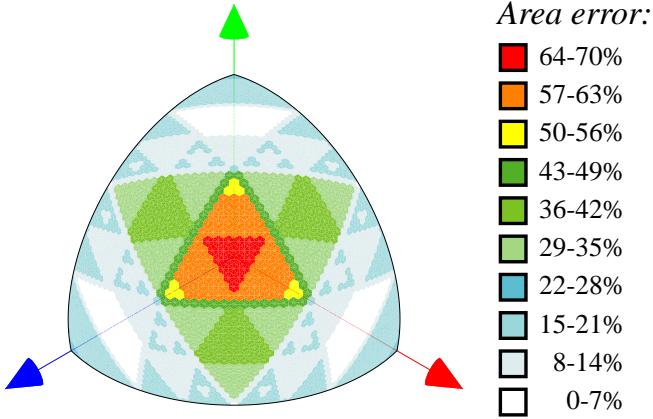


Figure 3.1: The area error distribution on a part of the octahedron hemisphere. Image courtesy of Mortensen [Mortensen, 2011]

wavelets in its 2D form as well as their spherical counterparts. For the 2D case, a discrete Haar wavelet transform is applied using the lifting scheme [Sweldens, 1996, 1998]. In general, for a 2D signal, the lifting scheme requires a multiresolution analysis stage for a certain level  $k$ , where color values from deeper levels are transformed to the coefficients of a scaling function  $\Phi^k$  and three wavelet functions  $\Psi_{hor}^k$  (horizontal),  $\Psi_{vert}^k$  (vertical) and  $\Psi_{diag}^k$  (diagonal). A quadtree subdivision scheme in the pixel domain is used for multiresolution analysis. A similar lifting scheme also exists for the spherical domain and has been introduced by Schröder and Sweldens [Schröder and Sweldens, 1995]. A spherical lifting scheme also requires a subdivision scheme, but instead of defining a quadtree subdivision in the 2D image domain, it needs to be defined in the spherical domain. An octahedron subdivision is used where each of the octahedron octants are represented by a quadtree of wavelet coefficients. Each triangle of the octahedron is subdivided in four children by adding vertices along each edge (geodesic bisector). For each triangle in the subdivision scheme, the underlying piecewise constant function is approximated using a linear combination of the scaling function and the three wavelets. This is called the synthesis operation and is implemented as a simple matrix multiplication:

$$\begin{bmatrix} x_0^{k+1} \\ x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Psi_{hor}^k(x_0^{k+1}) & \Psi_{vert}^k(x_0^{k+1}) & \Psi_{diag}^k(x_0^{k+1}) \\ 1 & \Psi_{hor}^k(x_1^{k+1}) & \Psi_{vert}^k(x_1^{k+1}) & \Psi_{diag}^k(x_1^{k+1}) \\ 1 & \Psi_{hor}^k(x_2^{k+1}) & \Psi_{vert}^k(x_2^{k+1}) & \Psi_{diag}^k(x_2^{k+1}) \\ 1 & \Psi_{hor}^k(x_3^{k+1}) & \Psi_{vert}^k(x_3^{k+1}) & \Psi_{diag}^k(x_3^{k+1}) \end{bmatrix} \begin{bmatrix} x^k \\ y_{hor}^k \\ y_{vert}^k \\ y_{diag}^k \end{bmatrix} \quad (3.3)$$

Here,  $\Psi_{hor}^k(x)$  is the value of  $\Psi^k$  at a specific position  $x$  on the subtriangle. The inverse operation, called analysis, is used to extract  $\Phi^k, \Psi_{hor}^k, \Psi_{vert}^k$  and  $\Psi_{diag}^k$  out of color samples of

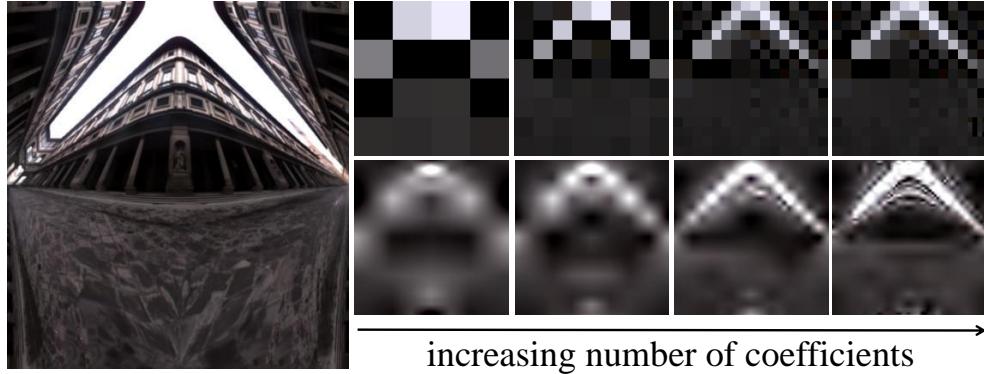
the underlying function and is performed bottom-up by applying the inverse of the foregoing synthesis matrix.

As stated before, we conducted experiments with different Haar wavelet bases. To this end, we implemented different spherical Haar bases, next to the traditional 2D Haar wavelet basis. There does not exist any orthogonal wavelet basis over the spherical domain. We implemented wavelet transforms for Bio-Haar wavelets [Schröder and Sweldens, 1995], Bonneau wavelets [Bonneau, 1999] and Pseudo Haar wavelets [Ma et al., 2006]. Bio-Haar wavelets are defined in the spherical domain and have the property of being bi-orthogonal. It specifies that the double product of the wavelet functions and the scaling function results in zero and is an essential condition for the existence of a wavelet:

$$\int \psi_{hor}^k \phi^k = \int \psi_{vert}^k \phi^k = \int \psi_{diag}^k \phi^k = 0 \quad (3.4)$$

Since bi-orthogonal wavelets are not fully orthogonal, truncating the least significant coefficients does not necessarily result in the best approximation and compression performance. Bonneau wavelets and pseudo Haar wavelets, on the other hand, are nearly-orthogonal, meaning that they become more orthogonal when the subdivision depth increases (i.e. when the spherical triangles become planar) [Nielson et al., 1997]. This is advantageous because they provide better approximation when the number of coefficients increase and thus allow for a better compression ratio.

The four wavelet bases are used to conduct different comparisons. The experiments were mainly focused on signals of the rendering equation, more specific the BRDFs. Furthermore, it is still possible to apply the triple product wavelet integral approach of Ng et al. [Ng et al., 2004], because the four wavelet bases are all devised from the Haar wavelet domain. The first experiment focused on the compression performance of a BRDF signal with respect to the wavelet basis. The MERL BRDF database [Matusik et al., 2003] has been used because it contains a mixture of different BRDF slices. Each slice of the BRDF database represents a spherical function defined at a point  $x$  on an object and observed from a view vector  $\omega_o$  as a function of the incident direction  $\omega_i$  over the hemisphere. Good compression needs to result in less coefficients with a small  $L^1$ -norm compared to the ground truth. Compression is performed with a fixed budget of wavelet coefficients. Only 5% of the most significant coefficients were retained. The main observation is that both Bonneau and pseudo Haar bases perform equally well and have an overall better compression performance compared to the 2D Haar wavelets. A second important observation is that the compression performance behaves differently for diffuse BRDF slices and specular BRDF slices. If we evaluate the specular BRDF slices separately, we observe that for some cases the spherical wavelets perform slightly worse and for most of them equally well as the 2D Haar wavelets. On the other hand, for the diffuse BRDF slices the spherical Haar wavelets outperform the 2D Haar wavelets for all cases with almost 15%. In some cases, the 2D Haar wavelets still outperform



**Figure 3.2:** Reconstruction of an environment map (left) with a small number of coefficients. A Haar wavelet basis is used in the top row on the right. The result is much blockier in comparison to the smoother Daubechies-6 reconstruction in the bottom row.

the spherical Haar wavelets, because of their fully orthogonal characteristics compared to the bi-orthogonality of spherical wavelets.

To have a clearer measurement of the parametrization error, independent from the wavelet representation, we conducted a second experiment. We first sampled the BRDF slices uniformly on the unit sphere with a statistical method [Dutre et al., 2001]. This can be used as the ground truth data of the spherical data. Then, we compared with a spherical octahedron subdivision representation and a 2D parametrization. There is an overall improvement of about 25% for the spherical representation compared to the ground truth, in contrast to a 2D parametrization.

### 3.2.2 Wavelets Tailored to the Signal

The best way to approximate piecewise constant signals, e.g. the visibility function in the rendering equation, is with a Haar wavelet basis. On the other hand, smooth signals, like an environment map or BRDF, are better approximated with high-order wavelet functions. Examples of high-order wavelet functions are Daubechies, Coiflets, Symlets, etc. Overall, the Daubechies, Coiflets and Symlets behave somewhat similar. They mainly differ in terms of symmetry, support—which describes the width of the wavelet—and number of vanishing moments. The choice of wavelet is primarily dependent on the target application. For example, the Coiflets-5 tends to be a little smoother compared to Daubechies-4 wavelets because of the larger support. This will be ideal for applications that require smoother wavelets. However, because of the larger support, they compromise on complexity. For some applications, the properties of the Daubechies wavelet, like an optimal number of vanishing moments for a certain support, can be beneficial. Texture analysis is a popular example. The Symlet and the

Coiflet wavelets have the extra property of being near-symmetric. This can be advantageous for the representation of symmetric functions. To maximize compression performance and minimize artifacts in the representation of different signals, it is beneficial to tailor the underlying basis representation to the signals itself. Figure 3.2 shows a comparison between the Haar wavelet basis and the Daubechies-4 wavelet basis. A reconstruction with few coefficients leads to a noticeable blockier result when using the Haar basis. In this particular example, the original signal requires roughly five times less coefficients in the Daubechies basis. This motivates our approach to perform the product integral calculation on this sparser representation. The development of a general triple product theorem for wavelets has allowed us to evaluate a triple product integral problem where the data can be approximated with any chosen wavelet basis.

### 3.3 Wavelet Product Integral

Many functions are naturally expressed in the spherical domain. Solving the rendering equation (see Equation 2.4) at each point in space can be considered a spherical convolution operation in signal processing. Each term in the convolution, in the rendering case  $V$ ,  $\tilde{\rho}$  and  $\tilde{L}$ , is expanded in an appropriate basis  $\Psi$ :

$$\begin{aligned} V(\omega) &= \sum_i V_i \Psi_i(\omega), \\ \tilde{L}(\omega) &= \sum_j \tilde{L}_j \Psi_j(\omega), \\ \tilde{\rho}(\omega) &= \sum_k \tilde{\rho}_k \Psi_k(\omega) \end{aligned} \tag{3.5}$$

If we substitute this basis representations in the rendering equation, we get a triple product integral

$$L_r(x \rightarrow \omega_o) = \sum_i \sum_j \sum_k V_i \tilde{L}_j \tilde{\rho}_k C_{ijk} \tag{3.6}$$

with triple product basis coefficients for the visibility  $V$ , reflectance  $\rho$ , environment lighting  $L$  and its corresponding binding coefficient  $C_{ijk}$  defined as:

$$C_{ijk} = \int_{\Omega} \Psi_i(\omega) \Psi_j(\omega) \Psi_k(\omega) d\omega \tag{3.7}$$

As described before, previous work either used harmonic analysis to model the functions compactly on the sphere, spherical wavelets or regular 2D Haar wavelets. Harmonic analysis has the advantage of being a natural choice of representation for spherical functions, but requires a high number of coefficients to represent high-frequency detail. Haar wavelets, on the other hand, are better suited to compactly represent these local details, because the basis functions have a small support. Haar functions lack smoothness, however, which makes them

less suitable for the approximation of smooth signals. In contrast, high-order wavelet bases (e.g. Daubechies) are better tailored to represent smooth signals.

In this chapter we focus on calculating the binding coefficients  $C_{ijk}$  for an arbitrary mixture of wavelet basis functions  $\Psi_i$ ,  $\Psi_j$  and  $\Psi_k$ . The position and dilation factor are controlled by the basis function numbers  $i$ ,  $j$  and  $k$ . Choosing an appropriate basis to represent the various factors becomes critical to achieve high quality results with minimal computation time. This dissertation recognizes that each factor in the triple product has different signal characteristics. Therefore, it would be efficient to encode each factor with a basis specifically tailored to it. While the Haar wavelet basis  $\Psi_H$  excels at encoding piecewise constant signals, like the visibility factor  $V$ , it fails at representing smooth signals compactly such as reflectance  $\rho$  and lighting  $L$ . For this task, smooth high-order wavelets, e.g. the Daubechies-4 basis  $\Psi_D$ , are generally considered more appropriate (the number 4 indicates the support size of the basis functions with the smallest dilation). These smooth bases often require an order of magnitude less coefficients for smooth signals. We will devise a novel method to calculate product integrals for arbitrary wavelet bases, focusing specifically on triple product integrals for rendering applications. We will also show the applicative advantages of using smooth wavelet for inverse rendering applications.

Figure 3.3 and Figure 3.4 show two wavelet bases:  $\Psi_H$  and  $\Psi_D$ . Without loss of generality, the illustrations in this dissertation show 1D wavelet functions. Separable wavelets in higher dimensions can easily be obtained by combining two lower dimensional basis functions. These bases are orthonormal:

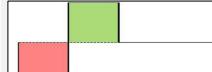
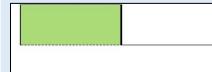
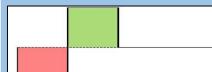
$$\int_{\Omega} \Psi_{Hi} \Psi_{Hj} d\omega = \delta_{ij} \text{ and } \int_{\Omega} \Psi_{Di} \Psi_{Dj} d\omega = \delta_{ij}, \quad (3.8)$$

where  $\delta_{ij}$  is the Kronecker delta [Clapham and Nicholson, 2009]. The double product integral of such orthonormal bases reduces to a convenient dot product. This can be verified in Figure 3.3(f) and Figure 3.4(f). Only basis functions with the same position and dilation result in a non-zero double product integral. Unfortunately, the triple product integral is substantially more difficult to calculate.

### 3.3.1 The Haar Tripling Coefficient Theorem

An analytical approach was developed for the Haar triple product integral, which iterates only over non-zero coefficients [Ng et al., 2004]. This is possible due to the strongly regular and hierarchical structure of Haar wavelets. Children with a smaller dilation factor fall completely under a constant part of their parent. Figure 3.3(g) shows that only specific and predictable combinations result in non-zero product integrals.

The Haar tripling coefficient theorem binds appropriate weighting factors to each of the wavelet coefficients that combine to form the product integral. They use an analytical approach to formulate the very limited set of cases that arise when calculating the triple product

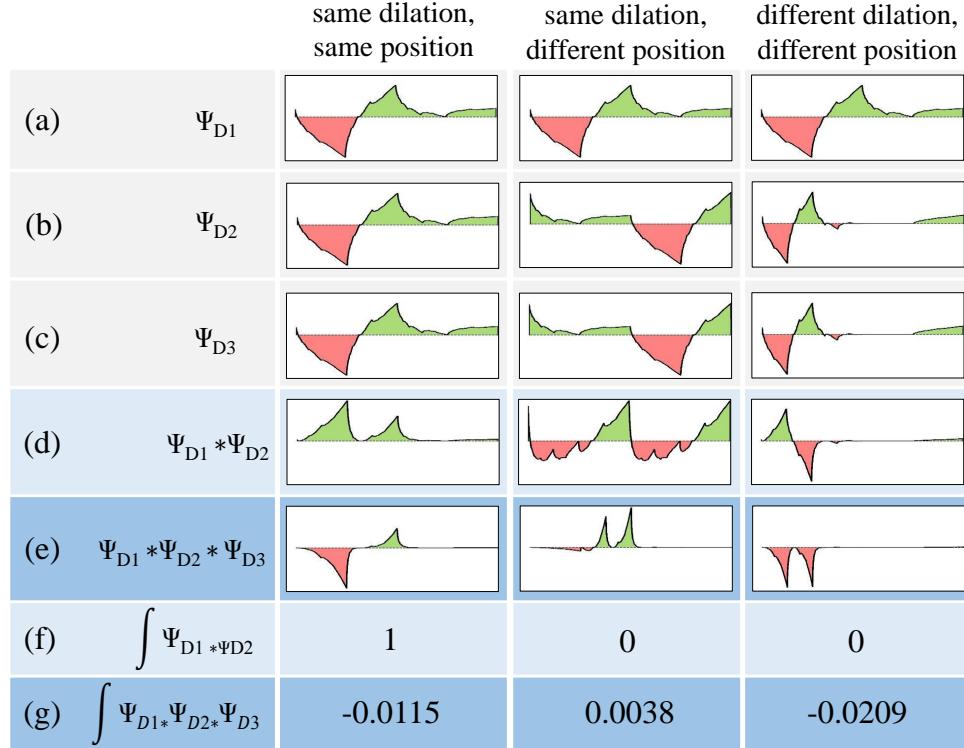
		same dilation, same position	same dilation, different position	different dilation, different position
(a)	$\Psi_{H1}$			
(b)	$\Psi_{H2}$			
(c)	$\Psi_{H3}$			
(d)	$\Psi_{H1} * \Psi_{H2}$			
(e)	$\Psi_{H1} * \Psi_{H2} * \Psi_{H3}$			
(f)	$\int \Psi_{H1} * \Psi_{H2}$	1	0	0
(g)	$\int \Psi_{H1} * \Psi_{H2} * \Psi_{H3}$	0	0	-1

**Figure 3.3: Double and triple product of Haar wavelets.** (a), (b) and (c) are the different basis functions  $\Psi_i$ ,  $\Psi_j$  and  $\Psi_k$ . Each column represents a permutation of  $i$ ,  $j$  and  $k$  that create basis functions with different dilation and/or position. The double product function is depicted in (d) with integral (f). The triple product functions is shown in (e) with integral (g). Green represents the positive part of the basis functions, red the negative part.

integral of simple Haar basis functions. Ng et al. realized that the different binding coefficients are very redundant for a simple Haar basis [Ng et al., 2004]. Their final tripling coefficient theorem has three cases, with variations for scaling and wavelet combinations. Furthermore, Haar wavelets with smaller dilation factors are always fully contained in a subset of the parents' support with constant value. These characteristics allow for a fast sublinear time algorithm that iterates over all non-zero coefficients. How the Haar tripling product algorithm works is detailed in Section 2.2.2.

### 3.3.2 General Tripling Coefficient Theorem

In contrast to the Haar wavelets, high-order wavelets are not piecewise constant functions anymore. Since high-order wavelets have a larger support, the number of non-zero combi-

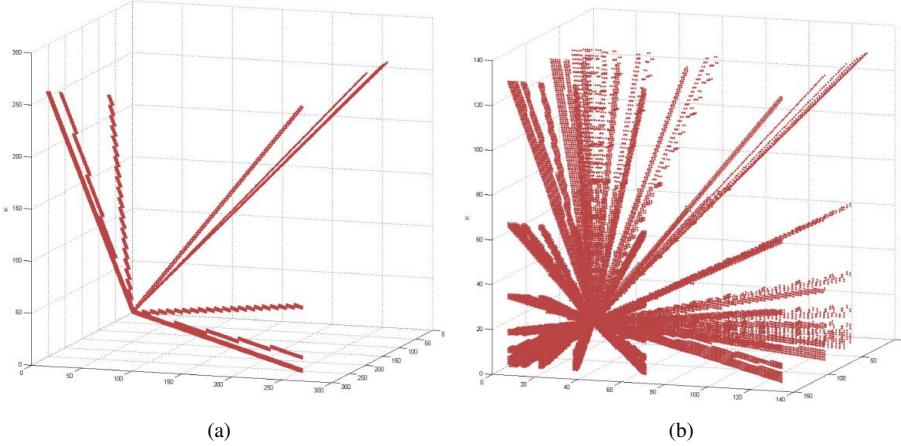


**Figure 3.4: Double and triple product of Daubechies wavelets.** (a), (b) and (c) are the different basis functions  $\Psi_i$ ,  $\Psi_j$  and  $\Psi_k$ . Each column represents a permutation of  $i$ ,  $j$  and  $k$  that create basis functions with different dilation and/or position. The double product function is depicted in (d) with integral (f). The triple product functions is shown in (e) with integral (g). Green represents the positive part of the basis functions, red the negative part.

nations will increase rapidly. A manual approach does not scale well when using smoother wavelets, since it is no longer feasible to manually identify and enumerate all the various cases. In addition, Haar wavelets with smaller dilation never overlap more than one coarser Haar wavelet. This is not true for high-order wavelets.

High-order wavelets have three properties that prevent them from having simple binding coefficients:

1. Their larger support results in more overlap and circular wrapping.
2. Children are no longer entirely contained by the support of their parent.
3. The subset of the parent support that a child overlaps is not necessarily constant.



**Figure 3.5:** Tensor of tripling binding coefficients for (a) Haar wavelets and (b) high-order wavelets. The Haar tensor clearly shows the three cases identified by Ng et al. [Ng et al., 2004]. High-order wavelets have a broader support, resulting in more non-zero binding coefficients. However, the tensor still remains sparse.

The support of the high-order wavelets remains compact, however, only being enlarged by a constant factor. Because of this, the tensor of binding coefficients remains sparse.

It can be seen in Figure 3.4(g) that the triple product integral for high-order wavelets is more complex. If we iterate over all combinations of  $i$ ,  $j$  and  $k$  and put them in one large tensor we get Figure 3.5(a) for Haar wavelets and Figure 3.5(b) for Daubechies wavelets. With this tensor precalculated, only the sparse non-zero coefficients need to be evaluated at runtime. Figure 3.6 visually compares the overlapping properties for Haar and the high-order Daubechies wavelets. It can be seen that the convolution of Haar wavelets yields a much sparser tensor than the convolution of high-order wavelets. This chapter shows results for the Daubechies wavelet, but the same approach can be applied to any other high-order wavelets like Coiflets and Symlets.

In this dissertation, we tackle the problem with a computational approach and calculate the tripling coefficients for a wide range of wavelets automatically. In the next section, we show how to calculate a tensor with binding coefficients of  $n$ -product integrals of a wide range of wavelet basis functions. Mixing of various basis types is supported. Our work mainly focuses on double and triple coefficients, but the approach can easily be extended to quadruple or higher product integrals. We will then apply some analysis of the tensor characteristics such as sparseness and symmetry and study the effect of mixing Haar and higher-order bases.

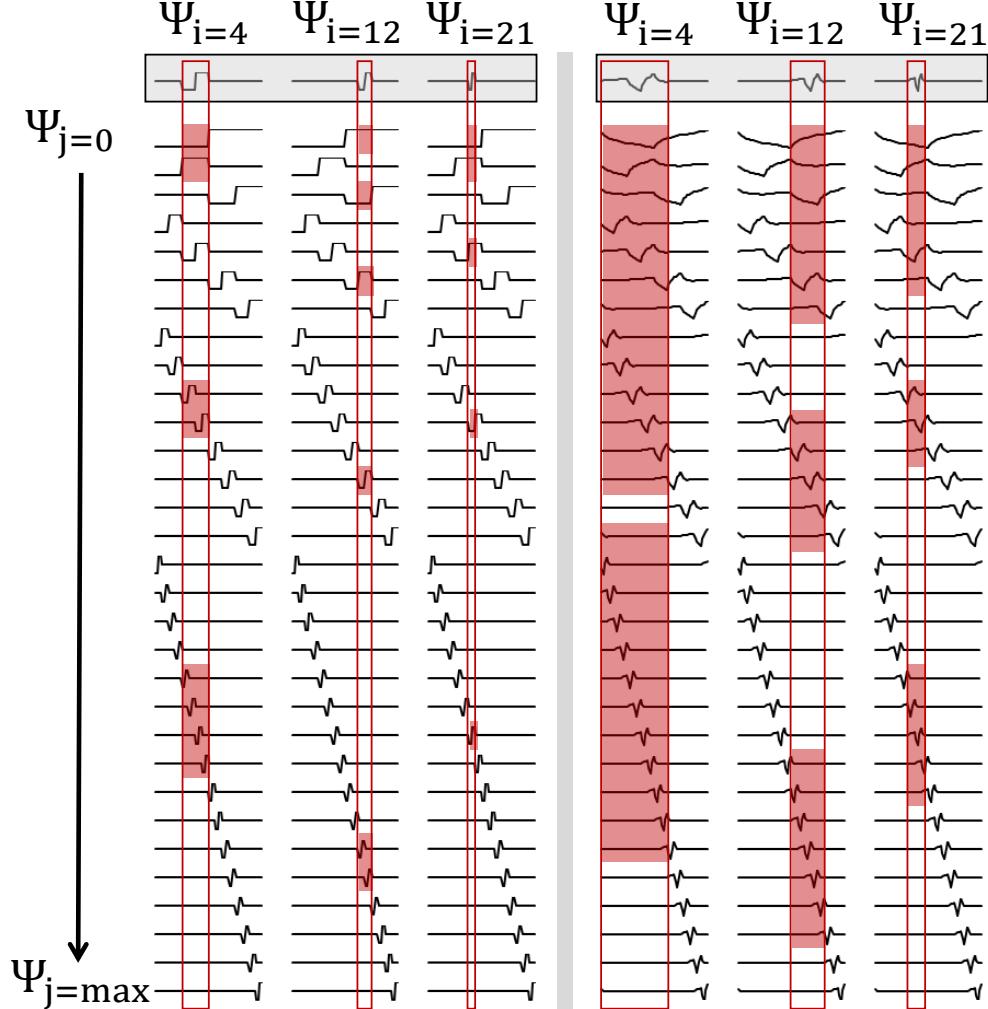


Figure 3.6: Hierarchically overlapping wavelet basis functions. Each wavelet basis function  $\Psi_i$  will have a fixed overlap with a set of other wavelet functions  $\Psi_j$ . The three columns on the left represent the overlap in case of Haar wavelets, specifically for  $\Psi_{i=4,12,21}$ . The three rightmost columns show the analogous cases for the Daubechies-4 wavelets. The overlapping part of the wavelets on each level that result in non-zero are marked in red. This figure clearly shows that when the support of the wavelet grows, the number of overlapping wavelets will increase.

## 3.4 Tensor of Triple Product Binding Coefficients

We experimented with different approaches to efficiently calculate the tensor of binding coefficients of different high-order wavelet functions, necessary for triple product rendering.

### 3.4.1 Naive Approach

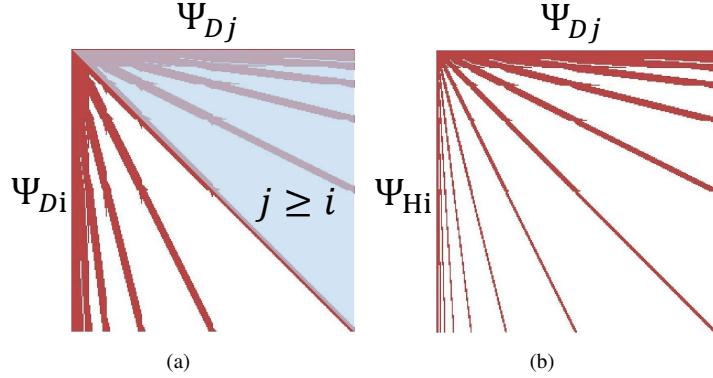
It is easy to envision a naive approach to calculate the tensor, based on the permutation of all  $i$ ,  $j$  and  $k$  and calculating the binding coefficients  $C_{ijk}$  (Equation 3.7). This quickly becomes intractable, due to its time complexity  $O(n^D * r^2)$  with  $D$  the dimensionality of the product integral (double, triple, quadruple, ...),  $n$  the total number of translations and dilations of the basis function and  $r^2$  the resolution of the 2D wavelet functions that are integrated. If the original signal contains many high-frequency perturbations,  $n$  can grow profoundly large. As a result the computation time increases drastically. For only a resolution of  $512 \times 512$  with 262144 wavelet scales and translations ( $n$ ), a triple product integral ( $D = 3$ ) for 2D wavelets requires an order of  $4.7 \times 10^{21}$  calculations. Clearly more sophisticated methods are required.

### 3.4.2 Hierarchical Approach

A second approach exploits the characteristics of higher-order wavelet bases. By design, each basis function has a fixed support. For example, a Daubechies-4 wavelet starts with a dilation factor of size 4 and will further dilate accordingly (4, 10, 22, 46, ...). As a result, a wavelet basis function  $\Psi_i$  will overlap only with a fixed number of other wavelet basis functions  $\Psi_j$ , each having a different position and dilation factor. Figure 3.6 illustrates the overlap of several basis functions  $\Psi_i$  with other basis functions  $\Psi_j$ . The figure displays that a basis function with a narrow dilation factor overlaps only with a small number of other functions. Only these specific combinations will possibly result in non-zero binding coefficients. All other combinations can safely be skipped. The time complexity is now reduced to  $O(n * C * (\log(n))^{D-1} * r^2)$  with  $C$  a constant related to the enlargement of support for high-order wavelets (in case of Daubechies-4,  $C \simeq 3$ ). The reconstruction of a  $512 \times 512$  resolution will now take approximately  $3.2 \times 10^{13}$  permutations, which is several orders of magnitude less than the naive approach.

### 3.4.3 Tensor Mirroring

When the mix of wavelet bases in a product integral is homogeneous, symmetry can be observed in the plotted tensor. For the double product case, the tensor is depicted in Figure 3.7(b). Since  $\int \Psi_i \Psi_j = \int \Psi_j \Psi_i$  is true, each combination of basis functions where  $j \geq i$  can be mirrored around the diagonal, eliminating half the work. In the case of a triple product integral with  $i$ ,  $j$  and  $k$  or a general product integral, taking advantage of the symmetry is done in an analogous manner.



**Figure 3.7: Tensor mirroring.** (a) Shows the overlap regions for a double product of Daubechies-8 wavelets  $\Psi_{Di}$  and Haar wavelets  $\Psi_{Hj}$ . (b) Shows the overlap regions for a double product where both  $\Psi_{Di}$  and  $\Psi_{Dj}$  are Daubechies-8 wavelet functions. In the case of two identical wavelet bases, the tensor is fully symmetric. The values in the upper triangle, indicated in blue, can be mirrored into the lower triangle. Note that in all cases the tensor is sparse.

### 3.4.4 Wavelet Sliding

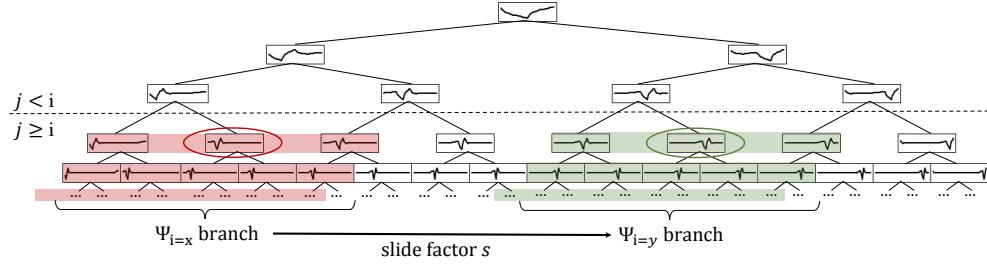
Previous optimizations have discussed improvements in computational complexity by reducing the number of wavelet permutations. The bottleneck for the overall tensor calculation is not just the number of wavelet permutations (e.g.  $10^5$  for  $512 \times 512$ ), but rather the actual work performed in each permutation ( $r^2$  or  $512^2$  multiplications for integration). Most of these calculations are redundant and can be avoided. Let us observe two basis functions  $\Psi_{i=x}$  and  $\Psi_{i=y}$  and their respective sets of overlapping basis functions  $O_{i=x}$  and  $O_{i=y}$  for which holds:

$$O_i = \{\Psi_j : (\int (\Psi_i \Psi_j) \neq 0, j \geq i\} \quad (3.9)$$

In Figure 3.8,  $\Psi_{i=x}$  is circled in red and  $O_{i=x}$  is marked in red.  $\Psi_{i=y}$  and  $O_{i=y}$  are marked green analogously. In the case of  $\Psi_{i=x}$  and  $\Psi_{i=y}$  having the same dilation factor, the number of overlapping functions will be equal ( $\#O_{i=x} = \#O_{i=y}$ ). However, the overlapping functions of  $\Psi_{i=y}$  are shifted with a sliding factor  $s$ :

$$s = (y - x) \times \text{support}(\Psi_{i=x}) \quad (3.10)$$

Using the observation above, only the overlapping functions of one specific translation need to be calculated, e.g. the red area in Figure 3.8. We call this a branch. The precalculated branch of binding coefficients can then be reused for all other translated basis functions with that specific dilation factor. This will reduce the amount of work drastically to  $O(C * (\log(n))^D * r^2)$ . The  $512 \times 512$  triple product example will now take approximately  $1.5 \times 10^9$  calculations.



**Figure 3.8: Wavelet sliding.** Given a wavelet basis function  $\Psi_{i=x}$  (circled in red), there is a branch in the wavelet tree of overlapping basis functions (red). The precalculated binding coefficient values of this branch can be reused for every translated  $\Psi_{i=y}$  (circled in green) with the same dilation. The branch is translated with slide factor  $s$  (see Equation 3.10).

### 3.4.5 Vanishing Moments

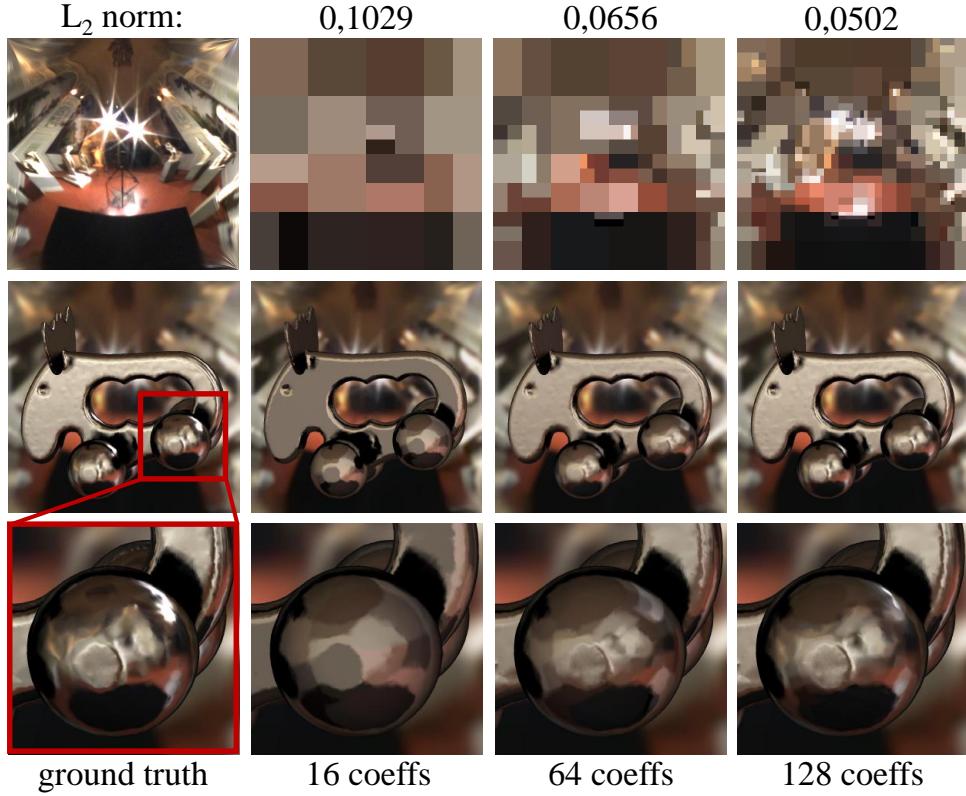
An additional advantage of high-order Daubechies wavelets is that they provide the maximum number of vanishing moments for wavelets of that specific support size [Daubechies, 1992]. In general, smoother wavelets generate more vanishing moments. The interesting property of wavelets with vanishing moments is that their product integral is not only zero when the basis functions do not overlap, but even for certain translations within their support. This yields extra sparsity in the tensor. The position of these vanishing moments are identified and their determination is incorporated in the wavelet sliding algorithm of the previous section. Afterwards, the data entries of vanishing moments can be removed from the precalculated branch.

### 3.4.6 Tensor Sparseness

The tensor of binding coefficients for different mixtures of wavelet bases is still rather sparse. Table 3.1 shows an overview of the sparseness for different mixtures of Haar, Daubechies and Coiflet wavelets. It is evident that a triple product, where the three factors are represented in the Haar wavelet basis, attain optimal sparseness. The tensors of binding coefficients for the high-order wavelets are clearly more dense. However, for a factor resolution of  $64 \times 64$ , they still achieve a sparseness ranging from 0.01% to 0.2% for different settings. The numbers clearly show that the support of the wavelet has a direct impact on the number of non-zero binding coefficients. If the spatial width of the wavelet function increases, it will result in more complex overlap cases, over all levels of the wavelet hierarchy.

**Table 3.1:** Tensor sparseness for different mixes of wavelet bases at different resolutions. The first three columns describe the type of wavelet used for the calculation of the triple product factors: Haar-2 (Haar, support 2), Daub-4 (Daubechies, support 4), Daub-6 (Daubechies, support 6) and Coiflet-5 (Coiflet, support 5). The tensor achieves its best sparseness for three Haar wavelet representations. However, one can observe that for other mixtures of high-order wavelets, the tensors remains relatively sparse as well. The increasing support of the wavelets has the most impact on the amount of sparseness.

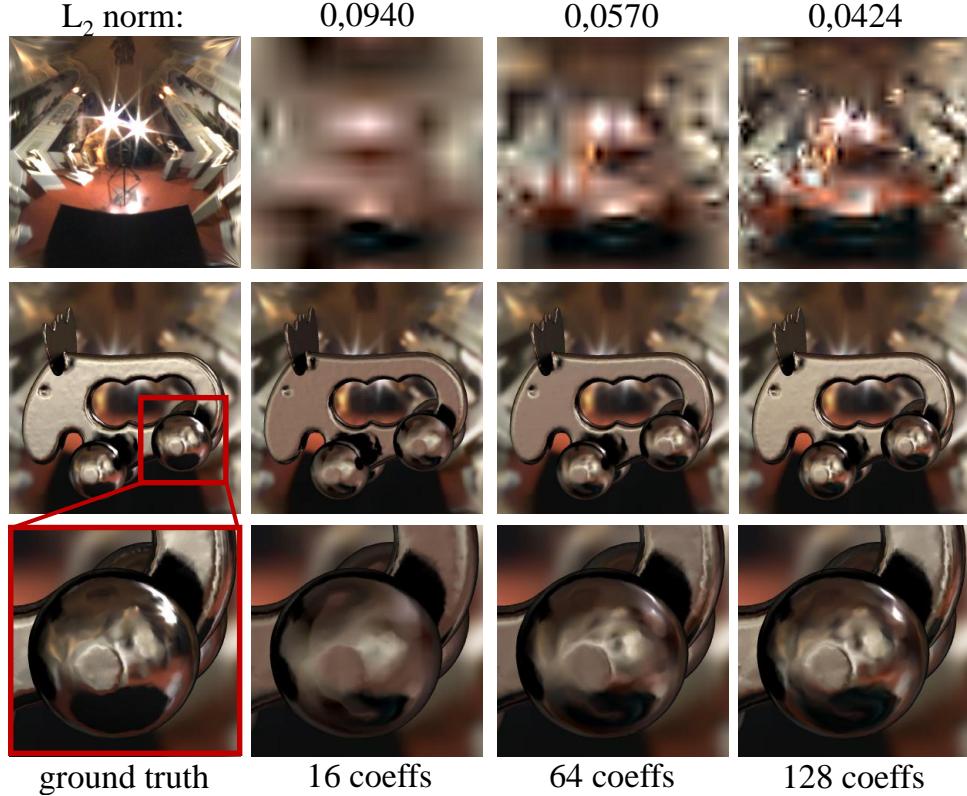
$\Psi_i$	$\Psi_j$	$\Psi_k$	resolution	non-zero coeffs	total coeffs	sparseness
Haar-2	Haar-2	Haar-2	$4 \times 4$	184	4096	4.4922%
Haar-2	Haar-2	Haar-2	$8 \times 8$	1288	262144	0.4913%
Haar-2	Haar-2	Haar-2	$16 \times 16$	7432	16777216	0.0443%
Haar-2	Haar-2	Haar-2	$32 \times 32$	38920	1073741824	0.0036%
Haar-2	Haar-2	Haar-2	$64 \times 64$	192520	68719000000	0.0003%
Daub-4	Daub-4	Daub-4	$4 \times 4$	1516	4096	37.0117%
Daub-4	Daub-4	Daub-4	$8 \times 8$	99088	262144	37.7991%
Daub-4	Daub-4	Daub-4	$16 \times 16$	1380136	16777216	8.2263%
Daub-4	Daub-4	Daub-4	$32 \times 32$	9720040	1073741824	0.9052%
Daub-4	Daub-4	Daub-4	$64 \times 64$	47918464	68719000000	0.0697%
Daub-6	Daub-6	Daub-6	$4 \times 4$	1516	4096	37.0117%
Daub-6	Daub-6	Daub-6	$8 \times 8$	214252	262144	81.7307%
Daub-6	Daub-6	Daub-6	$16 \times 16$	4360864	16777216	25.9928%
Daub-6	Daub-6	Daub-6	$32 \times 32$	29582032	1073741824	2.7550%
Daub-6	Daub-6	Daub-6	$64 \times 64$	145473456	68719000000	0.2117%
Coiflet-5	Coiflet-5	Coiflet-5	$4 \times 4$	1516	4096	37.0117%
Coiflet-5	Coiflet-5	Coiflet-5	$8 \times 8$	186706	262144	71.2227%
Coiflet-5	Coiflet-5	Coiflet-5	$16 \times 16$	3496930	16777216	20.8433%
Coiflet-5	Coiflet-5	Coiflet-5	$32 \times 32$	16408816	1073741824	1.5282%
Coiflet-5	Coiflet-5	Coiflet-5	$64 \times 64$	48918464	68719000000	0.0712%
Haar-2	Daub-4	Daub-4	$4 \times 4$	2212	4096	54.0039%
Haar-2	Daub-4	Daub-4	$8 \times 8$	31960	262144	12.1918%
Haar-2	Daub-4	Daub-4	$16 \times 16$	270664	16777216	1.6133%
Haar-2	Daub-4	Daub-4	$32 \times 32$	1555120	1073741824	0.1448%
Haar-2	Daub-4	Daub-4	$64 \times 64$	7327168	68719000000	0.0107%
Haar-2	Coiflet-5	Coiflet-5	$4 \times 4$	2212	4096	54.0039%
Haar-2	Coiflet-5	Coiflet-5	$8 \times 8$	59902	262144	22.8508%
Haar-2	Coiflet-5	Coiflet-5	$16 \times 16$	626034	16777216	3.7315%
Haar-2	Coiflet-5	Coiflet-5	$32 \times 32$	2715112	1073741824	0.2529%
Haar-2	Coiflet-5	Coiflet-5	$64 \times 64$	8699044	68719000000	0.0127%



**Figure 3.9:** Quality comparison of the Elch dataset compressed in the Haar wavelet basis. The three right columns show the quality differences with respectively 16, 64 and 128 coefficients retained. The mesh is rendered with the Haar triple product integral [Ng et al., 2004]. The results have larger  $L^2$ -norms and converge slower to the ground truth compared to our more general high order wavelet approach (Figure 3.10)

### 3.5 Results and Applications

Our main motivation for using high-order wavelets is that they provide a compact and high-quality approximation of smooth functions. This is particularly interesting for triple product integration to evaluate the rendering equation. The visibility factor is a piecewise constant function, for which Haar wavelets are ideally suited. We argue that the lighting environment map and certainly the BRDF (bidirectional reflectance distribution function) exhibit profoundly more smoothness. These factors are better represented with a smoother wavelet, for example Daubechies. In contrast to previous methods, our algorithm is able to calculate the triple product integral with a heterogeneous mix of wavelet bases, where each factor is coded in a basis specifically tailored to the signal characteristics.



**Figure 3.10: Quality comparison of the Elch dataset compressed in the Daubechies-6 wavelet basis.** The three right columns show the quality differences with respectively 16, 64 and 128 coefficients retained. The mesh is rendered with our high order wavelet tensor calculation. Our results have a smaller  $L^2$ -norms and converge faster to the ground truth compared an existing Haar wavelet approach (Figure 3.9)

Figures 3.9 and 3.10 compare the visual quality of blocky Haar wavelets and smoother Daubechies-6 wavelets for various compression rates. A similar comparison is made for a second model in Figures 3.11 and 3.12. The ground truth rendering at full quality is included for reference, alongside zoomed pictures on areas with the largest differences. The  $L_2$  norm is provided with each rendering as a quantitative comparison measure. A rendering with a smaller norm is closer to the ground truth. It can be seen that smooth high-order wavelets outperform Haar wavelets both qualitatively and quantitatively.

Figure 3.13 demonstrates the ability of our application to render with different kinds of wavelets. It is possible to mix and match wavelets to optimally represent the signals of all factors in the product integral calculation. In general, the smoothness characteristics of

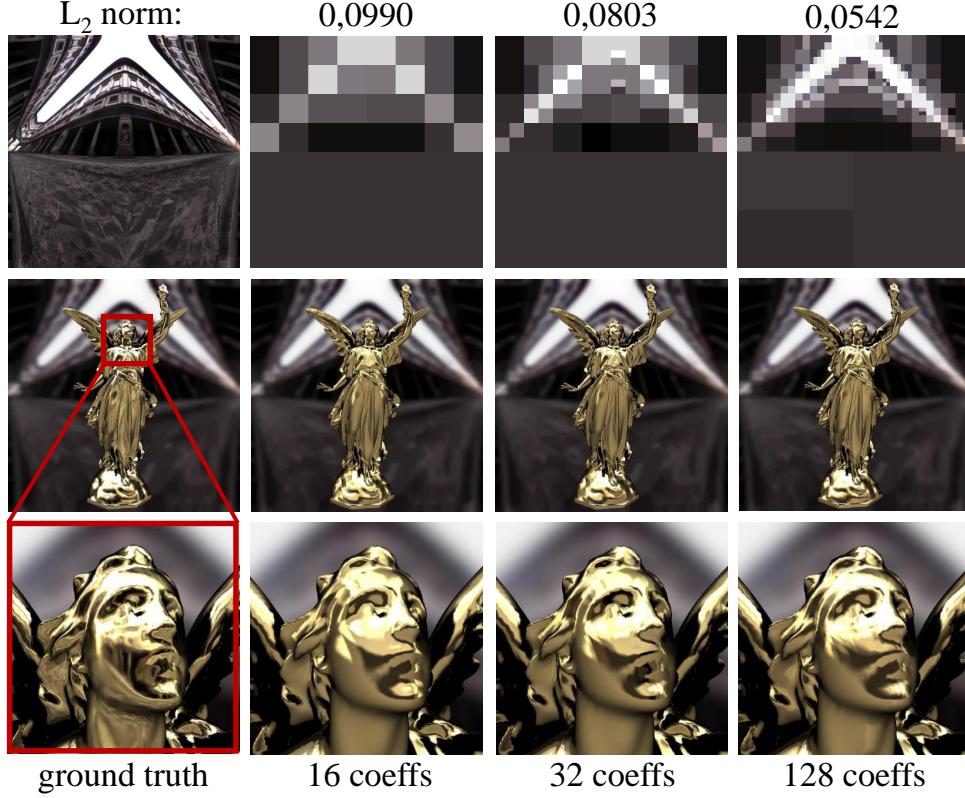
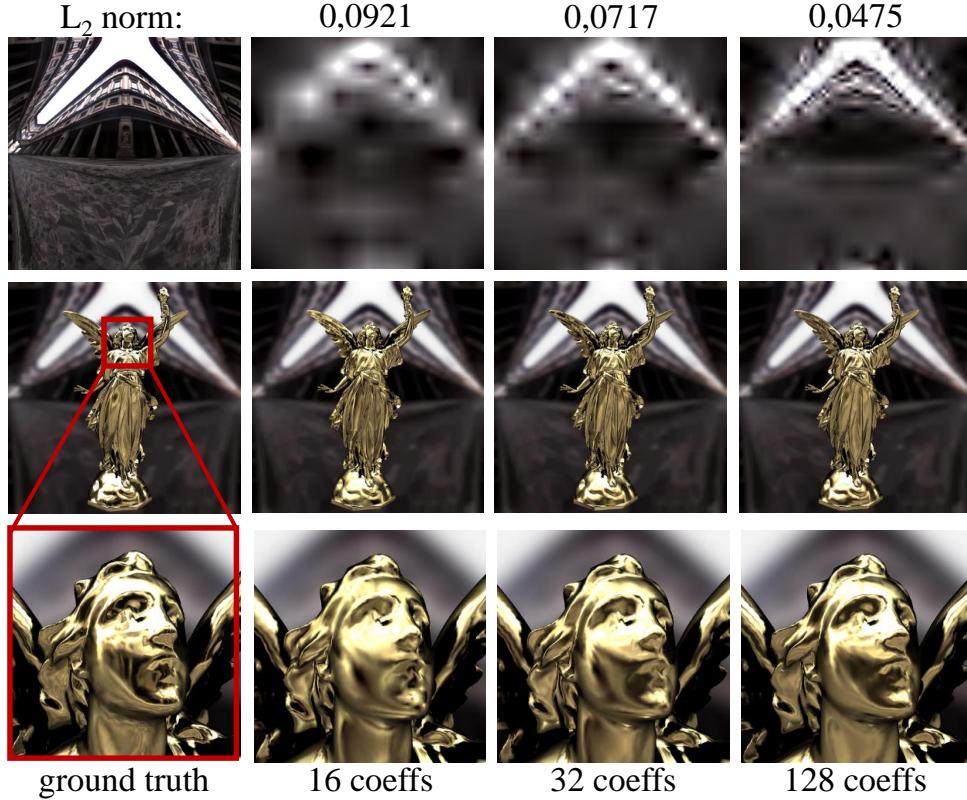


Figure 3.11: Quality comparison of the Lucy dataset compressed in the Haar wavelet basis. The three right columns show the quality differences with respectively 16, 64 and 128 coefficients retained. The mesh is rendered with the Haar triple product integral [Ng et al., 2004]. The results have larger  $L^2$ -norms and converge slower to the ground truth compared to our more general high order wavelet approach (Figure 3.12)

the wavelet basis should match those of the signal. In that case the wavelets will be able to represent the signal with a minimum of coefficients.

## 3.6 Discussion

In this chapter, we tried to find an answer to the question whether it is possible to use high-order wavelets in triple product rendering applications. We have developed a solution method where we automatically identify the binding coefficients of the product integral. Furthermore, we showed that by tailoring the choice of wavelet basis to the signal itself, a better compression performance can be reached. In this chapter we mainly focused on finding a solution



**Figure 3.12: Quality comparison of the Lucy dataset compressed in the Daubechies-6 wavelet basis. The three right columns show the quality differences with respectively 16, 64 and 128 coefficients retained. The mesh is rendered with our high order wavelet tensor calculation. Our results have a smaller  $L^2$ -norms and converge faster to the ground truth compared an existing Haar wavelet approach (Figure 3.12)**

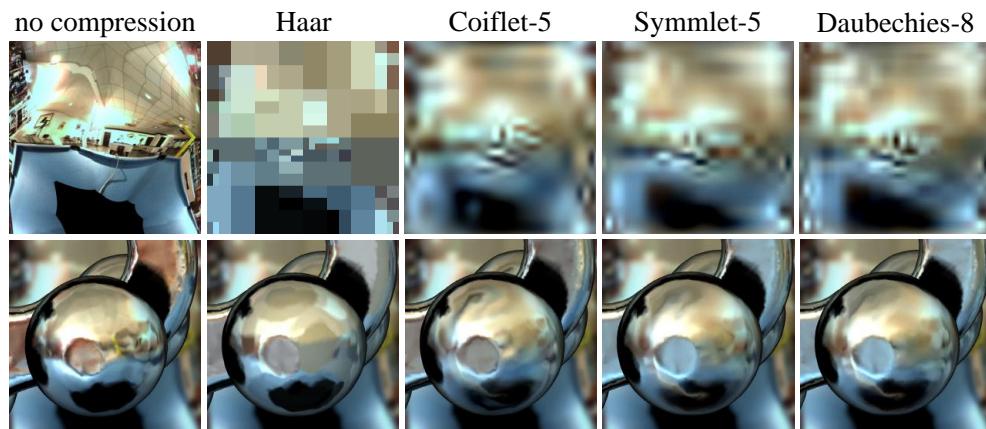
method rather than focusing on time complexity. A theoretical approach was used to quantify the complexity of the binding coefficients. It is hard to quantify their influence on the time performance. It is apparent that the tensor of binding coefficients for high-order wavelets is more dense compared to the Haar wavelet approach of Ng et al. [Ng et al., 2004]. However, their computation is still manageable (Table 3.1). Although the calculation of the tensor is efficient, we sacrificed some performance of the triple product calculation itself. The approach of Ng et al. [Ng et al., 2004] allowed for a linear time triple product integral calculation, because they were able to develop an efficient tree traversal algorithm that exploits the nature of the Haar wavelets. For example, overlap information of parent wavelets remain constant over its children. The parent overlap of a wavelet tree can be passed onto the entire subbranch of children. This allows them to only iterate over non-zero wavelet coefficients and to evaluate

only the exact cases where the binding coefficients result in non-zero. Unfortunately, in our more general approach, we can either iterate over the non-zero binding coefficients of the tensor or over the non-zero wavelet coefficients of the three signals. It is no longer possible to iterate over both at the same time. In the future, more research is required to improve the structural format of the tensor, which is now represented as a sparse tensor. For example, a tree structured representation that better matches the tree structure of the wavelets can help to diminish the overhead of zero evaluations.

A vital characteristic of high-order wavelets is its compression performance. The results clearly show that the number of coefficients of each signal can be significantly reduced, while at the same time a similar rendering quality is obtained. This has some pleasing benefits. The first is that a part of the time performance, lost in the triple product evaluation, can now be earned back, because fewer wavelet coefficients will result in fewer triple product evaluations. Furthermore, some applications (please refer to Section 6.2.2) require the smoothness behavior and good compression performance of high-order wavelets. The time performance for those applications is of secondary importance.

### **3.7 Conclusions**

In this chapter we have provided a novel method to calculate the binding coefficients of general product integrals. Our method is able to cope with a mix of heterogeneous bases. This allows the representation of factors in the product integral with piecewise constant or smooth basis functions, depending on the signal properties. The algorithm exploits both the hierarchical nature and vanishing moments of the wavelets basis, as well as the sparsity and symmetry of the tensor. Our rendering application has demonstrated that the tensor-based product integral leads to less blockiness in the results.



**Figure 3.13:** Product integral calculation with different wavelet basis functions. In this case, a compression with Haar, Coiflet-5, Symmlet-5 and Daubechies-8 is executed on the environment map. For each wavelet basis, the least significant coefficients are truncated, retaining only 1024 wavelet coefficients. The result of the triple product integral on the Elch dataset is shown respectively. Even though the same number of coefficients is used for each demonstrated wavelet basis, the visual quality of rendering can change drastically. This is due to the smoothness behavior of higher-order wavelets in contrast to the piecewise constant Haar wavelets.



# Chapter 4

---

## Spherical Radial Basis Functions for Interactive Triple Product Rendering

<b>4.1 Spherical Radial Basis Functions Product Integral .....</b>	<b>54</b>
<b>4.1.1 Types of Spherical Radial Basis Functions .....</b>	<b>54</b>
<b>4.1.2 Rendering with Spherical Radial Basis Functions.....</b>	<b>54</b>
<b>4.1.3 Spherical Radial Basis Functions versus Wavelets .....</b>	<b>55</b>
<b>4.1.3.1 Analytic Evaluation of the Product and Convolution of Two or More SRBFs ..</b>	<b>55</b>
<b>4.1.3.2 Fast BRDF Evaluation.....</b>	<b>55</b>
<b>4.1.3.3 Rotation to the Local Frame.....</b>	<b>56</b>
<b>4.1.3.4 High-frequency Information.....</b>	<b>57</b>
<b>4.1.3.5 Interactivity.....</b>	<b>57</b>
<b>4.1.4 Related Work .....</b>	<b>57</b>
<b>4.2 Dynamic Materials .....</b>	<b>60</b>
<b>4.3 Dynamic Visibility .....</b>	<b>61</b>
<b>4.3.1 Voxelization .....</b>	<b>62</b>
<b>4.3.2 Cone Tracing.....</b>	<b>63</b>
<b>4.3.3 Mapping Cones to SRBFs.....</b>	<b>64</b>
<b>4.4 Dynamic Lighting.....</b>	<b>66</b>
<b>4.4.1 SRBF Approximation .....</b>	<b>66</b>
<b>4.4.1.1 Hierarchical Grid of SRBFs using the Healpix Distribution .....</b>	<b>67</b>
<b>4.4.1.2 Residual Transform to Multiscale SRBFs .....</b>	<b>68</b>
<b>4.4.2 Calculating Overlapping SRBFs.....</b>	<b>71</b>
<b>4.4.3 GPU Implementation .....</b>	<b>72</b>
<b>4.4.3.1 GPU Execution and Architectural Model .....</b>	<b>72</b>
<b>4.4.3.2 Real-time Multiscale SRBFs .....</b>	<b>74</b>
<b>4.4.4 Peak Detection for High-Frequency Lighting .....</b>	<b>77</b>
<b>4.5 Results.....</b>	<b>78</b>
<b>4.6 Discussion.....</b>	<b>81</b>

<b>4.7 Conclusion and Future Work .....</b>	<b>85</b>
---------------------------------------------	-----------

---

Interaction between the real environment and virtual scenes is crucial in real-time relighting applications. Realistic integration of computer generated objects in real environments is extensively used in movie post-production and augmented reality applications. Fully interactive applications should allow immediate changes of three important factors. First, they allow changes in the lighting of a scene. These changes can affect both distant and near-field changes in illumination. Second, the geometry of the objects can be animated or changed by user input. Third and last, reflection properties of the objects can change as well. So far, a real-time evaluation of the rendering equation has remained impossible without constraining at least one of the three factors. Furthermore, it is also important that the content can contain high-frequency detail. There are several problems we need to address before reaching real-time frame rates.

To avoid the evaluation of the rendering equation in the pixel domain, precomputed radiance transfer with new bases like spherical harmonics [Sloan et al., 2002], Haar wavelets or eigenbases were introduced to sparsely represent the information in the rendering equation. These bases, however, place their own restrictions on the data or the operations they can perform on it. For example, spherical harmonics are not able to sparsely represent high-frequency signals. In Chapter 3, we have developed a *general triple product theorem* for wavelets which is an excellent solution for offline applications where very good compression rates are required, but it lacks flexibility, keeping us from a fast, dynamic and interactive renderer. Wavelets are constrained to static scenes, because the visibility needs to be precomputed and rotation is a complex operation in the wavelet domain. Our approach combines the advantages of previous approaches and will allow for real-time relighting of dynamic virtual scenes with real-time captured environment lighting. This is achieved by represented the underlying data of visibility, lighting and materials using *spherical radial basis functions* (SRBFs).

This chapter explains how to construct such a representation efficiently and how SRBFs will be used for triple product rendering. Section 4.1 explains in more detail how SRBFs can be used as an underlying basis for precomputed radiance transfer rendering. First, we give the definition of an SRBF (Section 4.1.1) and show how to use it to solve the rendering equation (Section 4.1.2). Then, we provide more detail in Section 4.1.3 on the comparison of our approach with respect to the wavelet basis for triple product rendering and explain why SRBFs will be more efficient. Last, Section 4.1.4 gives an overview of how SRBFs are used in state-of-the-art precomputed radiance transfer rendering and show how they often lack dynamic change of the three factors. Since interactivity is key, we developed new methods to ensure dynamic change of the three factors. Existing BRDF parameter models of the materials will be directly approximated using SRBFs (Section 4.2). The visibility needs to be sampled in real-time and we show how voxel cone tracing together with an efficient sampling scheme allow for fast and accurate soft shadows (Section 4.3). For environment lighting, we developed a multiscale residual algorithm for SRBF fitting (Section 4.4). We demonstrate

the effectiveness of our method with a real-time application (Section 4.5). Users can shine multiple light sources onto a camera and the animated virtual scene is relit accordingly.

## 4.1 Spherical Radial Basis Functions Product Integral

### 4.1.1 Types of Spherical Radial Basis Functions

A *spherical radial basis function (SRBF)* is a special case of a *radial basis function (RBF)*. A radial basis function is a function where the result depends on a distance measurement from a center point or any other point. Each function with this property is classified as a RBF. Radial basis functions are often used in artificial neural network applications. When they are defined on the sphere, they are called spherical radial basis functions. Example RBFs and SRBFs are Gaussians kernels, Poisson kernels and multiquadratics:

$$G_{Gaussian}(\omega \cdot c, \lambda, \mu) = \mu e^{\lambda(\omega \cdot c - 1)} \quad (4.1)$$

$$G_{Poisson}(\omega \cdot c, \lambda, \mu) = \mu \frac{1 - \lambda^2}{(1 - 2 * \lambda * (\omega \cdot c) + \lambda^2)^{\frac{3}{2}}} \quad (4.2)$$

$$G_{multiquadric}(\omega \cdot c, \lambda, \mu) = \mu \sqrt{1 + (\lambda * (\omega \cdot c))^2} \quad (4.3)$$

where  $\omega \cdot c$  is the dot product between a vector  $\omega$  and the center of the SRBF  $c$ , both defined on the sphere (i.e. the angle between the two vectors),  $\lambda$  is the bandwidth expressing the size of the SRBF and  $\mu$  the amplitude.

### 4.1.2 Rendering with Spherical Radial Basis Functions

As explained in the previous chapters, the simulation of light in a scene is formulated as a triple product integral of illumination, BRDF and visibility.

$$L_r(x \rightarrow \omega_o) = \sum_i \sum_j \sum_k V_i \tilde{L}_j \tilde{p}_k C_{ijk} \quad (4.4)$$

where  $C_{ijk} = \int_{\Omega} \Psi_i \Psi_j \Psi_k d\omega$  are called the binding or tripling coefficients of the three bases.

We choose to represent all three factors of the triple product integral in the *spherical Gaussians* basis  $G_{Gaussian}$ , as defined in Equation 4.1. By substituting the new Gaussian basis functions in the visibility, materials and lighting term of the rendering equation of Equation 4.4, we get a new rendering equation with the binding coefficients of spherical Gaussians.

$$L_r(x \rightarrow \omega_o) = \int_{\Omega} G_V(\omega \cdot c_V, \lambda_V, \mu_V) G_{\tilde{L}}(\omega \cdot c_{\tilde{L}}, \lambda_{\tilde{L}}, \mu_{\tilde{L}}) G_{\tilde{p}}(\omega \cdot c_{\tilde{p}}, \lambda_{\tilde{p}}, \mu_{\tilde{p}}) d\omega \quad (4.5)$$

$$= \mu_V \mu_{\tilde{L}} \mu_{\tilde{p}} e^{-(\lambda_V + \lambda_{\tilde{L}} + \lambda_{\tilde{p}})} \int_{\Omega} e^{\omega \cdot (c_V + c_{\tilde{L}} + c_{\tilde{p}})} d\omega \quad (4.6)$$

Tsai and Shih [Tsai and Shih, 2006] show how the integral of one or multiple Gaussian SRBF kernels can be reduced to a simple formula. Since the integral is rotation-invariant,  $\lambda_V * c_V + \lambda_{\tilde{L}} * c_{\tilde{L}} + \lambda_{\tilde{P}} * c_{\tilde{P}}$  is abstracted as  $r$ . They provide a proof of how Equation 4.6 can be further simplified to

$$L_r(x \rightarrow \omega_o) = \mu_V \mu_L \mu_{\tilde{P}} e^{-(\lambda_V + \lambda_L + \lambda_{\tilde{P}})} 4\pi \frac{\sinh(\|r\|)}{\|r\|} \quad (4.7)$$

The binding coefficients of Equation 4.6 are now reduced to a simple formula, resulting in a very efficient evaluation of the full rendering integral.

### 4.1.3 Spherical Radial Basis Functions versus Wavelets

The use of spherical radial basis functions or, more specific, spherical Gaussians have several advantages for triple product rendering compared to the wavelet equivalent. This section will detail how triple product rendering using SRBFs will result in a faster evaluation of the rendering equation for direct lighting.

#### 4.1.3.1 Analytic Evaluation of the Product and Convolution of Two or More SRBFs

Ng et al. [Ng et al., 2004] proved that the triple product of Haar wavelets can be implemented fast. They identified the limited number of cases where the triple product of three wavelets result in non-zero. However, this does not perfectly scale to other high-order wavelets like Daubechies. Here, more ingenious methods are required, as explained in Chapter 3. Similar to Haar wavelets, the triple product of SRBFs (e.g. spherical Gaussians) can be implemented fast. The triple product can be solved analytically [Tsai and Shih, 2006] and extracting triple binding coefficients is now straightforward.

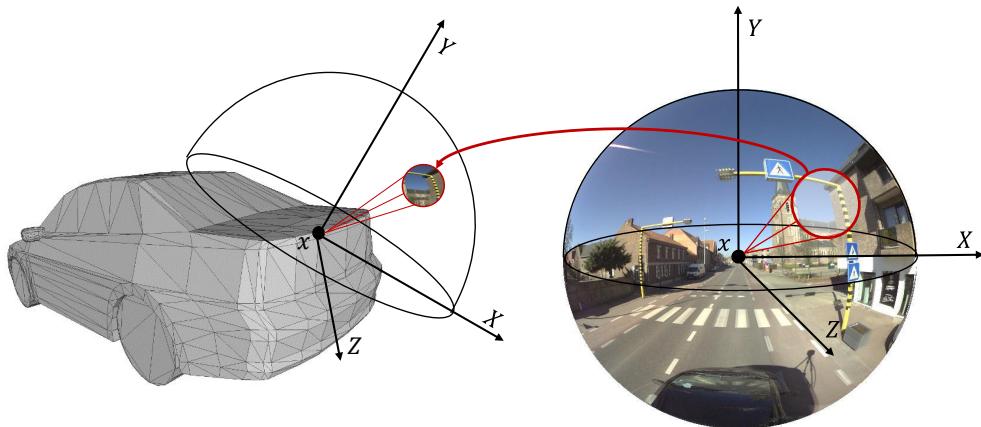
#### 4.1.3.2 Fast BRDF Evaluation

The reflectance properties of the rendering equation are modeled using a bidirectional reflectance distribution function (BRDF), which is a 6D function, depending on the incoming direction  $\omega_i$ , outgoing direction  $\omega_o$  and position  $x$  (more information in Section 2.1). The evaluation of the rendering equation requires the sampling of a 2D slice  $(\omega_i, \omega_o)$  of the BRDF for each pixel of the rendered image. As opposed to wavelets, there exists an efficient approach for spherical Gaussians to model and evaluate such 2D BRDF slices (see Section 4.2). Wavelets require that the 2D BRDF slice is first evaluated in the pixel domain, before it can be transformed to the proper 2D Haar wavelet representation. Although, slices can be buffered and reused, it still imposes quite a lot of overhead. SRBF approximations have no extra overhead, allowing for a run-time evaluation of reflectance for most BRDF models.

### 4.1.3.3 Rotation to the Local Frame

An important disadvantage of wavelet based methods is the absence of an efficient rotation operator in the wavelet domain. This is needed to rotate the environment into the local coordinate system of each vertex. State-of-the art wavelet techniques use an implementation of the Efficient Wavelet Rotation algorithm [Wang et al., 2006] to transform the environment map from the global frame to the local frame of each pixel. The algorithm creates bins for distinct normal directions ( $32 \times 32$ ). Each bin contains a rotation matrix that defines the transformation of an influence of each wavelet coefficient in the global frame to the local frame. Rotations in between bins of preprocessed normals are bilinear interpolated using the four nearest bins of rotation matrices. The preprocessed rotation matrices can be reused for different environment maps, nevertheless it requires a lot of computation, since for each wavelet coefficient, the influence on all other wavelet coefficients are calculated in the pixel domain. Furthermore, the preprocessed data is rather large (easily a few gigabytes of memory), making it inefficient to put in video memory and the rotation itself requires rather large matrix multiplications.

Unlike Haar wavelets, spherical functions are rotationally symmetric and thus allow for an efficient rotation. SRBFs are easily rotated on the sphere by rotating their center positions  $\mathbf{c}$ . For example, Figure 4.1 depicts the rotation of an environment map, defined in the global frame, to an environment map in the local frame.



**Figure 4.1:** Rotation of an environment map defined in the global frame (right) to an environment map in the local frame of a vertex (left). When using the SRBF basis, the rotation operator is done by simply rotating the SRBF centers.

#### 4.1.3.4 High-frequency Information

Similar to wavelets, SRBFs are an efficient representation for both low- and high-frequency information. SRBFs share the same characteristic of local support. Small detail can be approximated using small SRBFs. Overall, the number of SRBFs needed to make an approximation of a signal is generally larger compared to wavelets, especially when using high-order wavelets (Chapter 3). However, the amount of SRBFs is still exponentially smaller compared to spherical harmonics analysis. The time performance lost as a result of the small increase in coefficients will be compensated by the other advantages of SRBFs.

#### 4.1.3.5 Interactivity

Solving the triple product integral efficiently does not suffice for real-time relighting applications. A key property of most applications is live interaction. To achieve this, the change of all three factors should also be dynamic. For example, augmented reality applications often require a dynamic change in lighting conditions, as well as a dynamic change in geometry. On the other hand, inverse rendering techniques require a lot of iterations and permutations of different unknowns. To evaluate each set of unknowns, a dynamic update of visibility, lighting and materials is beneficial. In order to interactively change lighting conditions or alter scene geometry and materials, those factors need to be converted to the SRBF representation in a fast manner, in contrast to state-of-the-art SRBF or wavelet techniques that use slow preprocess steps to transform one or more factors to the proper representation. Our work mainly focuses on interactivity, where an update of the three factors into the SRBF representation should be real-time as well.

### 4.1.4 Related Work

The first precomputed radiance transfer rendering techniques were all based on spherical harmonics [Sloan et al., 2002; Kautz et al., 2002; Sloan et al., 2003]. Spherical harmonics allow for fast triple product rendering, but have the drawback of being limited to low-frequency lighting.

Ng et al. [Ng et al., 2004] represent the visibility term, the environment map and the BRDF slice that corresponds to the viewing direction  $\omega_o$  in the Haar wavelet basis  $\Psi$ . This basis is defined over the hemisphere using the hemi-octahedral parametrization as introduced by Praun and Hoppe [Praun and Hoppe, 2003]. To keep the data in the spherical domain, triple product rendering is sometimes evaluated using a spherical wavelet basis [Schröder and Sweldens, 1995; Ma et al., 2006; Put et al., 2014]. In Chapter 3, we extended this to work with high-order wavelets [Michiels et al., 2013, 2014b]. Wavelet representations are good for compression and can be used efficiently in the triple product integration of precomputed radiance transfer rendering. However, the factors themselves cannot be transformed easily to the wavelet representation and require offline processing. Liu et al. [Liu et al., 2004] and

Haber et al. [Haber et al., 2009] used the aforementioned approach to efficiently compress and render with the three factors. However, none of the three factors are dynamic. For example, in the approach of Haber et al. [Haber et al., 2009], all BRDF slices are first sampled in pixel domain before they are transformed to wavelets and the visibility is preprocessed using ray tracing. Liu et al. [Liu et al., 2004], on the other hand, used clustered principal component analysis (CPCA) to compress Haar wavelet segments of the lighting cube map. The same CPCA compression is applied to the preprocessed visibility cube maps. A new basis to represent the three factors of the rendering are spherical radial basis functions (SRBFs). The main advantage of SRBFs is an efficient rotation operator.

Tsai and Shih [Tsai and Shih, 2006] proposed a method for rendering with all-frequency lighting using SRBFs. The lighting Gaussians are fitted using an optimization process which results in high compression ratios, but can take up to several minutes to fit the full lighting factor. Furthermore, the visibility is precomputed and compressed using Clustered Tensor Approximation (CTA) and is therefore limited to static scenes only.

Wang et al. [Wang et al., 2009] also presented rendering with SRBFs for all-frequency lighting. They lack the possibility to render dynamic scenes since the visibility is precalculated using a spherical signed distance function (SSDF). Moreover, SSDFs are not able to accurately render detailed shadows. To represent the lighting factor, they use the same elaborate fitting process as Tsai and Shih [Tsai and Shih, 2006].

Lam et al. [Lam et al., 2010] proposed a hierarchical method for transforming environment lighting to a SRBF representation. The positions and bandwidth parameters of the SRBF are determined by the Healpix [Gorski et al., 2005] distribution. The coefficients are obtained by a least-square projection [Sloan et al., 2003].

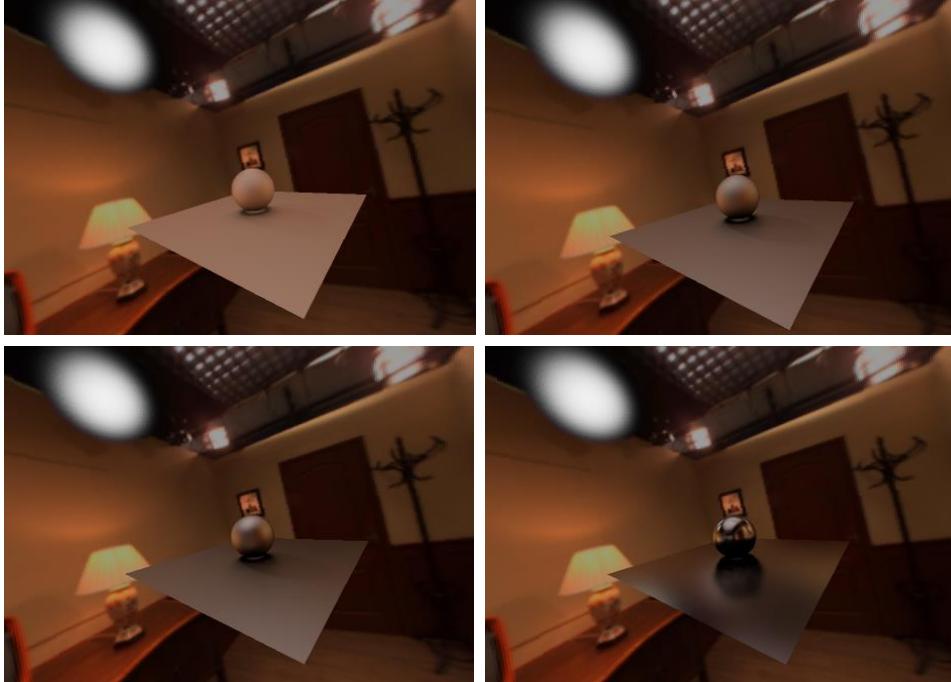
Meunier et al. [Meunier et al., 2010] explained how cosine lobes can be used to represent each term of the rendering equation. The use of a cosine lobe representation for compactly representing a reflectance function is common. The authors also showed how direct spherical light sources can be approximated with cosine lobes as well as showed how to offline simplify the geometry into a tree of spheres. By splatting these spheres onto a quad-tree, the visibility is sampled. Only rigid transformations of the geometry are possible.

Iwasaki et al. [Iwasaki et al., 2012] developed a real-time renderer for dynamic scenes which is able to render under all-frequency lighting using spherical Gaussians. The lighting can be rotated, but cannot be updated dynamically. The visibility is dynamically estimated by projecting bounding volumes onto a grid of patches of the hemisphere. Occluded patches are represented with spherical Gaussians. However, updating the bounding volume tree is strongly dependent on the complexity of the scene. Besides, their tree traversal uses suboptimal branching operation on GPUs.

**Table 4.1:** Comparison of our approach to other techniques that use precomputed radiance transfer for triple product rendering. All compared techniques use spherical harmonics (blue), wavelets (red) or spherical Gaussians (green) to solve the rendering equation. However, they all constrain at least one of the factors. Our approach allows for real-time updates for as well lighting, materials and visibility.

	all-frequency	dyn. lighting	dyn. geometry	dyn. shading
Sloan et al. 2002	✗	✗	✗	✗
Kautz et al. 2002	✗	✗	✗	✗
Sloan et al. 2003	✗	✗	✗	✗
Ng et al. 2004	✓	✗	✗	✗
Liu et al. 2004	✓	✗	✗	✓
Ma et al. 2006	✓	✗	✗	✗
Tsai and Shih 2006	✓	rotation only	✗	✗
Haber et al. 2009	✓	✗	✗	✗
Wang et al. 2009	✓	rotation only	✗	✓
Lam et al. 2010	✓	rotation only	✗	✗
Meunier et al. 2010	✓	limited	rigid	✓
Iwasaki et al. 2012	✓	rotation only	low-poly	✓
our approach	✓	✓	✓	✓

Our approach also uses spherical radial basis functions to represent all three factors. In contrast to previous techniques, we are able to construct and update all three factors in real-time. We show how a residual transformation technique can be used to efficiently transform the lighting information to spherical Gaussians. Cone tracing together with peak-detection of powerful and high-frequent lights allow for interactive rendering of soft shadows. Table 4.1 gives an overview of all the discussed techniques. Note that only our technique allows real-time adjustment of the lighting. Other techniques support only static lighting or rotation of the environment map. Furthermore, the performance of our real-time visibility tracing is not dependent on the number of polygons.



**Figure 4.2: Dynamic materials.** The lobe of a BRDF is approximated using one or few Gaussians. This direct mapping allows for real-time changes in appearance of the material. For example a virtual object can be rendered with different kinds of materials: Lambertian (top left), diffuse Phong (top right), glossy Phong (bottom left), specular Phong (bottom right), Cook-Torrance, ...

## 4.2 Dynamic Materials

A good material representation must have the following characteristics:

1. The representation should be compact.
2. Easy rotation of the BRDF into the global lighting and visibility frame.
3. Fast evaluation of double and triple product integrals.

SRBF functions can approximate almost any BRDF. They are rotationally symmetric and easily rotatable. They also allow for a compact all-frequency representation, due to their multiscale nature. It is known that BRDF functions expressed in terms of a normal distribution function can be approximated with Gaussian lobes [Ngan et al., 2005; Wang et al., 2009]. These Gaussians are then integrated in the triple product rendering integral. Instead of sampling the 2D slices of the BRDFs in the pixel domain as we did for wavelets, these can now be directly approximated with few SRBFs. The BRDF models used in this dissertation

all are based on microfacet theory, meaning that the specular lobe  $\tilde{p}$  can be described in terms of a normal distribution function (NDF)  $D(h)$ , with  $h$  the halfway vector and a remaining factor  $M(\omega_i)$ :

$$\tilde{p}(\omega_o, \omega_i) = M(\omega_i)D(h) \quad h = \frac{\omega_o + \omega_i}{\|\omega_o + \omega_i\|} \quad (4.8)$$

BRDFs that are expressed in terms of the normal distribution (NDF) can be approximated with spherical Gaussians.  $D(h)$  is approximated using a single spherical Gaussian for isotropic models or multiple spherical Gaussians for anisotropic models.  $M(\omega_i)$  is very smooth and can be approximated by a constant. We implemented multiple BRDF models. For example, fitting the Blinn-Phong model [Blinn, 1977] with a Gaussian lobe is done as follows:

$$\begin{aligned} M(\omega_i) &= \frac{n+2}{2\pi} \\ D(h) &= (h \cdot n)^\lambda \approx G(n; h, \lambda, 1) \end{aligned} \quad (4.9)$$

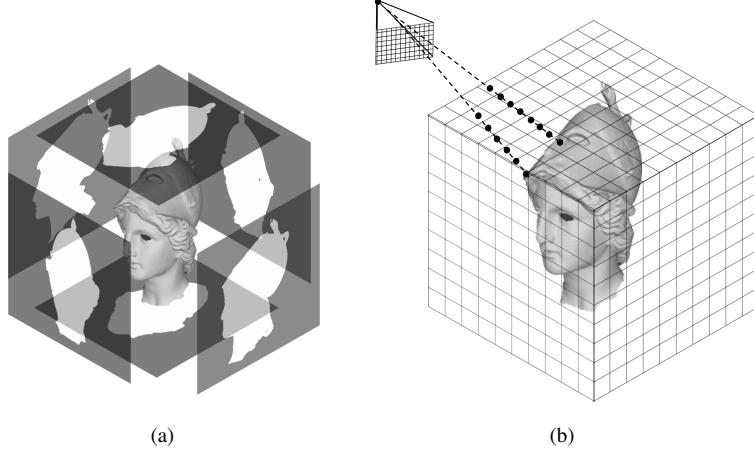
To obtain the 2D BRDF slice  $\tilde{p}(\omega_o, \omega_i)$  for a specific view direction  $\omega_o$ , we first need to warp the lobe described in terms of the halfway vector into the lobe defined in terms of the view direction. The warp for every lobe is specified as:

$$\begin{aligned} p_{\omega_i}^W &= 2(\omega_o \cdot h)h - \omega_o \\ \lambda_{\omega_i}^W &= \frac{\lambda_{\omega_i}^D}{\tau(p_{\omega_i}^D)} \\ \mu_{\omega_i}^W &= \mu_{\omega_i}^D \end{aligned} \quad (4.10)$$

where the differential area of the warp  $\tau(h) = 4\|h \cdot \omega_o\|$  is defined as the determinant of the Jacobian of the warp function. The details of fitting different BRDF models and warping them on the sphere are described in detail by Wang et al [Wang et al., 2009].

### 4.3 Dynamic Visibility

A fast and dynamic approximation is required to evaluate the complex visibility term of the rendering equation in real-time. Previous techniques often relied on precomputation. Haber et al. [Haber et al., 2009] used ray tracing to sample the visibility of each vertex in the pixel domain, before they were transformed to the Haar wavelet domain. Different approaches are developed for approximating visibility with spherical radial basis functions. Tsai and Shih [Tsai and Shih, 2006] used clustered tensor approximation which gives good compression rates but takes a lot of time to calculate. Wang et al. [Wang et al., 2009] represented the visibility term as a preprocessing step into a spherical signed distance function (SSDF). Iwasaki et al. [Iwasaki et al., 2012] were able to sample the visibility dynamically by projecting bounding volumes on the hemisphere. However, updating the bounding volume tree



**Figure 4.3:** A voxel volume is created by orthographically projecting the shape in the six main axis directions in a single render pass. (a) Illustration of the six rendered projections. (b) Once the voxel texture is filled, the volume can be previewed using ray casting. For each pixel of the image plane, a ray is cast into the volume. The position where the ray penetrates the voxel volume is calculated using a bounding box test. This serves as a start position. Then small steps are traced along the ray until it hits a non-zero voxel.

is strongly dependent on the complexity of the scene. Furthermore, their tree traversal uses suboptimal branching operation on GPUs.

When changes in the geometry occur frequently, these solutions are no longer feasible. In our framework, we combine the precomputed radiance transfer rendering technique with voxel cone tracing [Crassin et al., 2011b,a] to evaluate the visibility in real-time and entirely on the GPU. The scene is prerendered into a voxel volume for each frame. Once the volume is filled, the visibility can be evaluated for each point in space by integrating visibility information over larger areas in the volume. The visibility of each surface point is approximated by tracing one or multiple cones covering the entire hemisphere of the surface point. By tracing cones instead of single rays (ray tracing), a larger area of the hemisphere is traced at once.

### 4.3.1 Voxelization

The voxel volume can be represented in different ways. A tree structure is often used to efficiently traverse the voxel volume with logarithmic time complexity. It only requires simple bounding box testing. Crassin et al. [Crassin and Green, 2012] proposed an efficient sparse voxel octrees implementation on GPU. However, their GPU implementation still requires a lot of branching operations. Another approach to efficiently trace cones on GPU is to represent the voxel volume in a 3D texture. During voxelization, the geometry is mapped to

the size of the 3D volume texture. The voxels that coincide with the shape of the rendered object are filled in. Then, a mapping is defined from 3D world coordinates  $(x, y, z, 1)$  to voxel coordinates  $(v_x/v_w, v_y/v_w, v_z/v_w)$ , that transforms a point in 3D enclosed by a bounding box size  $(b_x, b_y, b_z)$ , centered around  $(c_x, c_y, c_z)$ , to a point in the 3D texture with a resolution of  $N^3$ :

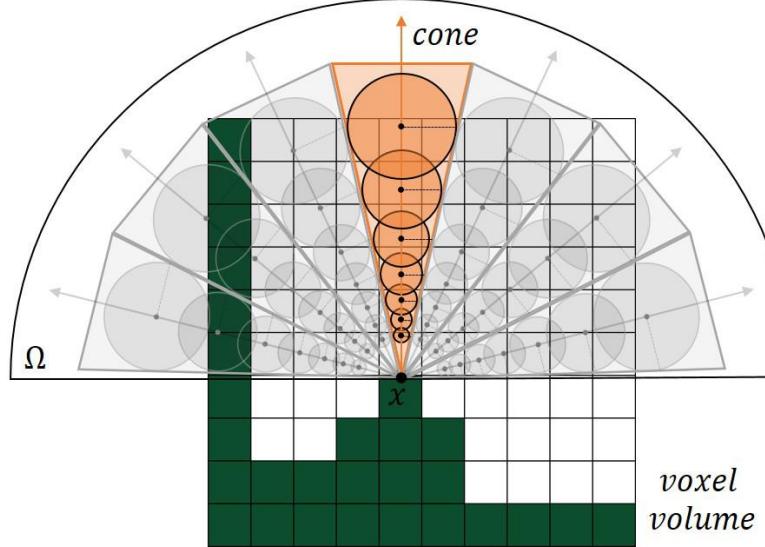
$$\begin{bmatrix} v_x/v_w \\ v_y/v_w \\ v_z/v_w \\ 1 \end{bmatrix} = \begin{bmatrix} N/b_x & 0 & 0 & c_x \\ 0 & N/b_y & 0 & c_y \\ 0 & 0 & N/b_z & c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4.11)$$

Generally, all polygons are projected onto the voxel volume in six render passes. Each render pass will project the polygons orthographically to one of the six main directions, parallel to the main axes. Each orthographically projected fragment is stored in the volume texture by mapping its world coordinates to the correct volume coordinates, using the mapping above. To improve performance, we implemented a single-pass voxelization algorithm [Crassin and Green, 2012], where the six render passes are merged and reduces to only one. A geometry shader is used to project all polygon fragments to their dominant axis. Based on the eye space normal's dominant direction, the correct swivel matrix for the orthographic projection is assigned for each triangle. Conservative rasterization is applied by expanding each triangle with a small value. This is mandatory to deal with small errors at the borders of the triangles and to prevent the occurrence of small gaps. Figure 4.3(a) illustrates how the voxel volume is generated. The voxel volume can be previewed with volume ray casting on the 3D texture, which is illustrated in Figure 4.3(b).

3D MIP-maps are generated to access larger areas of the voxel volume more efficiently. Each MIP-map level is filled in by a compute shader that uses trilinear interpolation of the 8 voxels in the MIP-map level above. Each voxel now contains the mean value of its underlying, more detailed, voxels. These MIP-mapped versions of the volume will be extensively used in the cone tracing for applying level of detail, since the footprint of cones will increase with the distance to the center. The lower MIP-maps are ideal for returning a mean value of a larger footprint.

### 4.3.2 Cone Tracing

The second step is to perform cone tracing in the voxel volume. Each visible pixel corresponds to a 3D position in voxel space. From this position, new cones are constructed and traced throughout the voxel volume. The size of each cone is defined by its cone ratio, which is the ratio of the radius of the cone at distance one. This is equal to the tangents of the half angle of the cone. Using the similarity theorem of triangles, the size of the cone  $s$  at a specific distance  $d$  along the cone can be obtained using the cone ratio:  $s = \text{coneratio} \cdot d$ . This size is used to calculate the proper MIP-map level, where only one value needs to be accessed, representing the mean value of the number of voxels present under that specific cone area.



**Figure 4.4: Cone tracing of the visibility function.** The position  $x$  of each surface point in the voxel volume is calculated using Equation 4.11. Starting at point  $x$ , multiple cones (indicated in gray) are traced along the hemisphere. The sampling step size is variable and is calculated based on the cone ratio. The footprint increases in proportion to the distance along the cone. The MIP-map levels are used to efficiently access the mean of a larger area in the voxel volume with only one fetch. The sampling ends when the boundary of the voxel volume is reached or the occlusion value is accumulated to one.

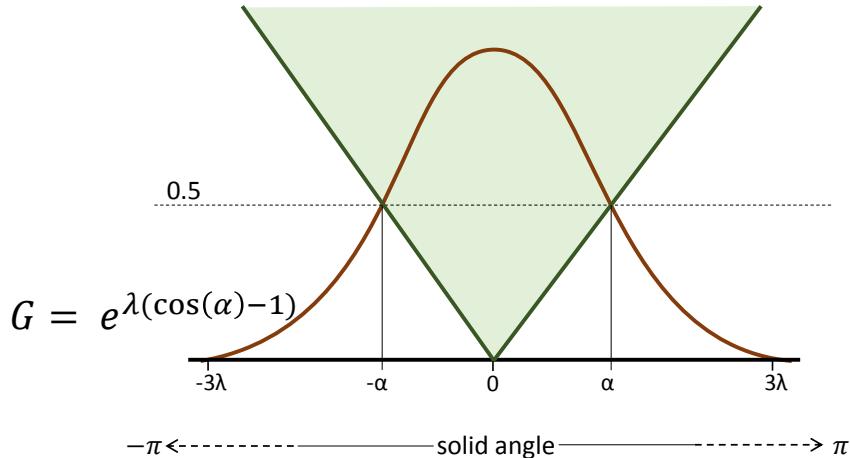
The mean coverage of the cones are accumulated during the tracing along the cone until the border of the volume is reached or the value is accumulated to one, meaning the entire cone is blocked by voxels. Often multiple cones need to be traced to cover the entire hemisphere of a surface point. Figure 4.4 shows how different cones are traced over the entire hemisphere of a surface point  $x$ .

### 4.3.3 Mapping Cones to SRBFs

Current voxel cone tracing is often used to perform ambient occlusion or low-frequency global illumination effects. However, in our application, we shoot cones to evaluate the visibility factor of the triple product integral. For this to work, a mapping from cones to spherical Gaussians is essential.

Suppose we have a Gaussian with size  $\lambda$ . Let us fit a cone that coincides with the distance  $\alpha$  where the Gaussian evaluates to 0.5:

$$e^{\lambda(\cos(\alpha)-1)} = 0.5 \quad (4.12)$$



**Figure 4.5:** Mapping of a visibility cone to a spherical Gaussian SRBF. A SRBF is approximated with a visibility cone (indicated in green) by calculating the angle of the cone where it overlaps with the region where the influence of the SRBF is halved (Equation 4.13).

Solving for  $\alpha$  results in

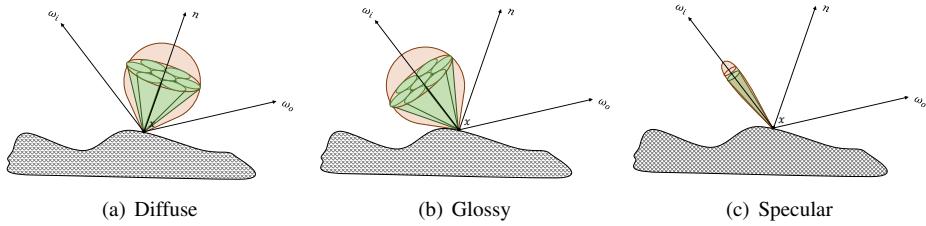
$$\alpha = \cos^{-1}\left(\frac{\log(0.5)}{\lambda} + 1.0\right) \quad (4.13)$$

where  $\alpha$  is used as the solid angle for the cones to trace. This is illustrated in Figure 4.5.

The last step is to select the appropriate cones to trace. A straightforward method would trace one visibility cone for each BRDF lobe. However, important high-frequency detail might be missed. For BRDF lobes with a large solid angle, all visibility details are integrated over a large area of the hemisphere. We use importance sampling in order to know where to trace cones for visibility. This is done by subsampling the BRDF lobe with smaller visibility cones. The subsampling density depends on the characteristics of the BRDF. A circle packing algorithm is used to determine the center position and radius. Figure 4.6 shows our subsampling scheme. For diffuse Lambertian BRDF lobes, a small number of uniformly traced visibility cones will suffice. For more glossy BRDF lobes, the subsampled cones are scaled and rotated into the support of the lobe. The number of cones used for subsampling determines the desired level of quality. Because the size of a BRDF lobe dictates the recoverable frequencies [Ramamoorthi and Hanrahan, 2001], our subdivision of scaled cones within the BRDF lobe retains approximately the same level of quality for all lobe sizes. In our implementation we obtained good quality renderings using a subdivision of 7 cones.

The quality and accuracy of cone tracing is highly dependent on the resolution of the voxel volume. We compare the rendering of ambient occlusion and hard shadows for different

resolutions of the voxel volume in Figure 4.7. It is clear that when the voxel volume resolution is too low, eg.  $64 \times 64 \times 64$ , much of the detail in the shadows is lost. When the resolution is increasing, more and more detail becomes visible.



**Figure 4.6: Adaptive subsampling of visibility cones.** (left) Gaussian of a Lambertian BRDF lobe (orange) approximated with a number of visibility cones (green). Middle: Gaussian of a glossy BRDF lobe (orange) approximated with the same number of visibility cones (green), rotated and scaled to enclose the BRDF. (right) In the case of a specular BRDF lobe (orange) it suffices to only trace a single visibility cone (green) that coincides with the lobe.

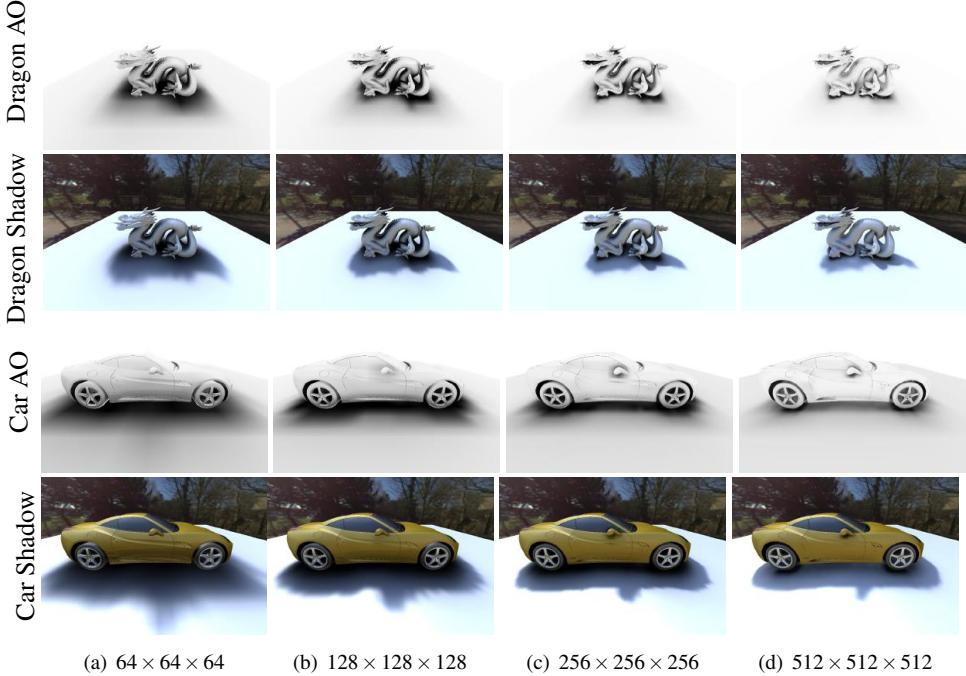
## 4.4 Dynamic Lighting

Real-time changes to the lighting environment will drastically alter the look of rendered three-dimensional models. The entire distant lighting sphere is typically captured using an omnidirectional camera (e.g. with a Ladybug3 [PointGrey, 2014]). Each omnidirectional frame can be used as an environment map in precomputed radiance transfer rendering. The data is represented in the pixel domain using a certain spherical mapping parametrization and needs to be transformed to the proper SRBF representation.

### 4.4.1 SRBF Approximation

Different algorithms for approximating lighting using SRBFs already exist in state-of-the-art research. For example, Tsai and Shih [Tsai and Shih, 2006] and Wang et al. [Wang et al., 2009] have implemented an algorithm that achieves high compression rates by fitting spherical Gaussians, but it requires a slow optimization process that takes up to several minutes to estimate both SRBF centers and bandwidths. Lam et al. [Lam et al., 2010] achieved better time performance by obtaining the SRBFs with a least-square projection. Here, the coefficients are distributed evenly over the sphere in a multiscale manner. However, though the transformation process is linear, it is far from real-time because of slow inverse matrix calculations.

In order to make the entire process more interactive, we prefer an approximation process that avoids such costly optimization algorithms. At the same time it should be able to run on GPU.



**Figure 4.7: The accuracy of the visibility depends on the resolution of the voxel volume. More resolution will result in more detailed ambient occlusion (AO) and casted shadows.** (a), (b), (c) and (d) show ambient occlusion and shadow renderings with an increasing resolution in voxel volume. The example of the car shows clearly that the shadows cast by the side-mirror are much more detailed for a 512 resolution compared to the lower resolutions.

To achieve this, we propose a fast multiscale algorithm to transform environment map frames to a SRBF representation. Our algorithm is inspired by the RBF fitting algorithm of Ferrari et al. [Ferrari et al., 2004]. It uses a fixed number of SRBFs on each scale, where each scale is approximated separately, using a straightforward weighted means approach. The residue for each scale is passed onto the next scale.

#### 4.4.1.1 Hierarchical Grid of SRBFs using the Healpix Distribution

The environment map frames are approximated with a *multiscale SRBF model*. The domain of the spherical function is subdivided in different hierarchical layers of SRBFs with increasing density and decreasing lobe size. The SRBF centers of a specific level are uniformly distributed over the spherical surface. We use spherical Gaussians as a basis representation for lighting, similar to visibility and materials. In Section 4.1, we have shown that the lighting  $L$  is approximated using a basis  $\Psi$ . If we substitute this basis with a spherical Gaussian basis

we get the following:

$$L(\omega) = \sum_i \mu_i G_L(\omega \cdot c_i, \lambda_{L_i}) \quad (4.14)$$

Here the lighting  $L(\omega)$  for a specific direction on the sphere  $\omega$  is approximated as the sum of  $i$  different spherical Gaussians with different centers  $c_i$ , different bandwidths  $\lambda_{L_i}$  and different weights  $\mu_i$ . To allow for a hierarchical approach, the different Gaussians are organized in a fixed grid of SRBFs with decreasing lobe sizes over different scales. Equation 4.14 can now be rephrased in function of the multiscale hierarchy:

$$L(\omega) = \sum_{s=0}^N L_s(\omega) \quad (4.15)$$

$$L_s(\omega) = \sum_{c=0}^{M_s} \mu_c G(\omega \cdot p_c, \lambda_s) \quad (4.16)$$

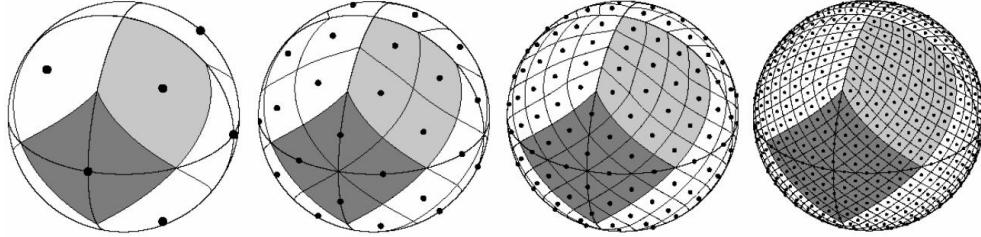
The full approximation of  $L(\omega)$  is the sum of the approximations  $L_s(\omega)$  for each level, with  $N$  the number of levels. A level  $s$  is approximated with a fixed number of Gaussians ( $M_s$ ) in the grid. The spherical Gaussians in each level have equal bandwidth  $\lambda_s$ .

This dissertation uses the Healpix [Gorski et al., 2005; Lam et al., 2010] distribution on the sphere to select the centers for the different Gaussians over the different scales. The Healpix distribution is optimal for distributing the spherical surface in equal area parts. Moreover, it is suitable for hierarchical algorithms, since one Healpix quadrilateral on a specific level can be easily subdivided in four new equally-sized Healpix quadrilaterals on the next level. An illustration of the Healpix distribution on a sphere is shown in Figure 4.8. The number of Healpix centers for the first five levels are respectively 12, 48, 192, 768 and 3072.

The lobe size  $\lambda$  of the Gaussians on a certain level is derived from  $e^{\lambda(\cos(\frac{\alpha}{2})-1)} = \frac{1}{2}$ , which specifies that the Gaussian of that level has only a support of 0.5 at the halfway point of two neighboring Gaussian with centers  $c_1$  and  $c_2$ , where  $\alpha$  is the angle between the two centers. Since all Gaussians are fixed in the Healpix grid, the position of the centers and hence the angle  $\alpha$  is known. Solving for the lobe size results in  $\lambda = \ln(\frac{1}{2}) / (\cos(\frac{\alpha}{2}) - 1)$ . The train of thought is that the sum is approximately one in between two Gaussians. The lobe sizes for the first five levels are respectively 5.50, 22.55, 91.50, 364.51 and 1449.29.

#### 4.4.1.2 Residual Transform to Multiscale SRBFs

Both the SRBF positions and the lobe sizes are fixed for each level in the multiscale hierarchy. The only unknown parameters required for an SRBF approximation are the amplitudes  $\mu$  of each Gaussian. These are estimated using a *residual transformation algorithm*. The *residue function* is the remainder of pixel values in the lighting function  $L(\omega)$  that is not fully



**Figure 4.8:** The Healpix distribution scheme subdivides the sphere in equal area parts. The Healpix quadrilaterals of a specific level can be subdivided in 4 new equally-sized Healpix quadrilaterals on the next level. The Healpix subdivision scheme is used to place the environment lighting SRBFs in a multiscale grid. Courtesy of Gorski et al. [Gorski et al., 2005].

approximated by the SRBFs at a certain moment in the algorithm. In the first iteration, the input lighting data is assigned as residue. The approximation of the spherical signal into a SRBF representation is started at the root level of the multiscale hierarchy, consisting of only 12 Healpix SRBF centers. Only the SRBFs that decrease the residual error of the signal will be inserted, i.e. where  $\mu$  is not equal to zero. The weight  $\mu$  for a specific spherical Gaussian  $G$  at a level  $s$  is extracted by a local weighted mean measurement in the residue function, i.e. the average of all pixels in the residue  $r_s$  weighted by the influence of the spherical Gaussian on all the pixels [Ferrari et al., 2004].

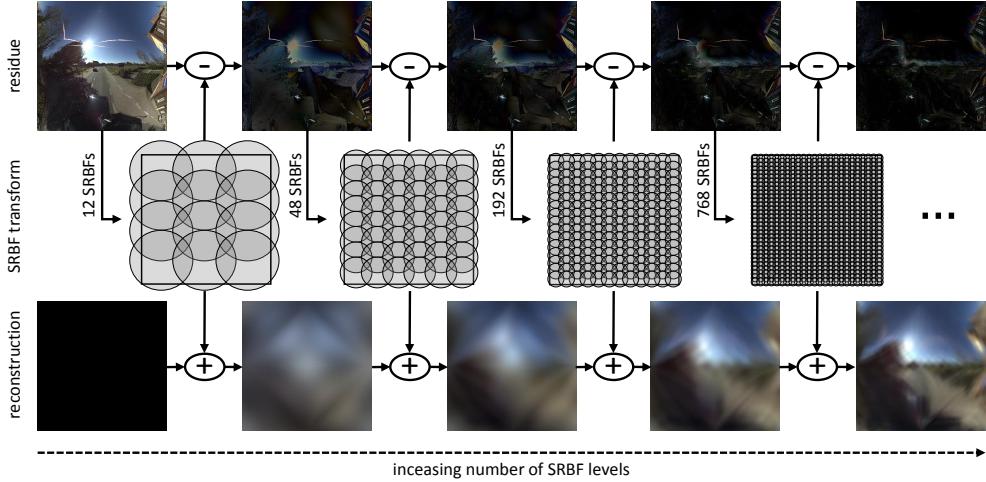
$$\mu_c = \frac{\sum_{\omega \in R(p_c)} r_s(\omega) G(\omega \cdot p_c, \lambda_s)}{\sum_{\omega \in R(p_c)} G(\omega \cdot p_c, \lambda_s)} \quad (4.17)$$

Only the directions  $\omega$  in the neighborhood of the illumination function that are influenced by the SRBF are important to consider. That is why only a region  $R(p_c)$  of the residue around the center is taken into account. In practice the overlap is calculated using a logarithmic bounding box test between the directions in the residue and the tree structure of the Healpix hierarchy. The overlap testing will be preprocessed and put on GPU. Section 4.4.2 will detail more on the efficient calculation of this overlap testing.

Once the weights for all the SRBFs of level  $s$  are estimated, Equation 4.16 is used to calculate  $L_s(\omega)$ , which is a reconstruction of the spherical signal for each direction  $\omega$  using the estimated SRBFs. This reconstruction is used to calculate an updated residue that is passed onto the next level. The residue  $r_{s+1}(\omega)$  for a level  $s+1$  is the difference of the residue and the reconstruction of the current level using the estimated SRBFs:

$$r_{s+1}(\omega) = I(\omega) - \sum_{i=0}^s L_i(\omega) = r_s(\omega) - L_s(\omega) = r_s(\omega) - \sum_{c=0}^{M_s} \mu_c G(\omega \cdot p_c, \lambda_s) \quad (4.18)$$

The new residue  $r_{s+1}(\omega)$  will serve as an input signal for the next level (for example 48 Healpix SRBF centers for the second level) and, then again, Equation 4.17 is used to approx-



**Figure 4.9: Residual transform of an environment map in the pixel domain to an underlying SRBF representation.** The algorithm starts approximating the environment map with the first level of SRBFs. The coefficients of the SRBFs are estimated using a local weighted mean measurement (Equation 4.17). Then, based on the reconstruction of the coefficients, the residue is calculated and passed onto the next level, where, again, the residue of the environment map is reconstructed with the next level of SRBFs. This process is iterated until a max number of levels is reached.

imate the weights of the new SRBFs. The process is iterated until the maximum level of detail is reached or the residue error is below a certain threshold. This residue error is defined as the average value of the residue function over the entire range of values. The described algorithm is fast and allows for a real-time transformation. Unlike other methods, no complex optimization is used.

The algorithm above is defined in the spherical domain. For more efficient calculations, we will use the parametrization of Praun and Hoppe [Praun and Hoppe, 2003] to represent the 3D spherical data in the 2D domain. The lighting data can now be represented in the pixel domain, called an *environment map*, and each pixel represents a direction  $\omega$  on the sphere and is initialized as the residue image  $r_0(\omega)$ . The process is illustrated in Figure 4.9. Here we initialize the residue with an outdoor environment map. In the first level the residue is approximated with 12 SRBFs and then used to calculate the new residue  $r_1(\omega)$ , which is passed onto the next level where it is approximated with 48 SRBFs. This process is repeated until a maximum depth is reached. The estimated SRBF coefficients can be used to reconstruct the original signal, as shown in the bottom row of the figure. In addition, the estimated SRBF coefficients can now be used directly as a lighting representation in the rendering equation.

#### 4.4.2 Calculating Overlapping SRBFs

The number of lighting SRBFs is variable and starts from 12 SRBFs for a level 1 reconstruction up to 4092 SRBFs for a level 5 reconstruction. The render speed will drop down drastically if each pixel of the rendered output image needs to evaluate all 4092 SRBFs. However, each point in the scene will only reflect a certain part of the hemisphere in the direction of the pixel on the image plane. Not all SRBFs will affect each rendered pixel because most of the SRBFs in a deeper level of the hierarchy will not overlap with the reflected area of the hemisphere. They are redundant and evaluating them will have a great impact on the time performance of rendering. Remember that the area of reflection is defined by the BRDF parameter model, and more precisely its SRBF lobe size and position, as explained in Section 4.2. The rendering speed is directly linked to the type of BRDF. The fact that a BRDF is diffuse, glossy or specular will define the number of overlapping environment SRBFs that need to be evaluated and thus have a direct influence on the time performance. Two mechanisms are used to calculate the overlapping SRBFs.

First, the BRDF lobe size defines what part of the hemisphere is reflected into the pixel direction. It is clear that a big lobe size has overlap with more spherical Gaussians of the environment map.

Second, we argue that for a glossy reflection, and thus a bigger lobe size, the detail of the environment map is integrated over a larger area and thus less detail of the environment map is required. It is redundant to evaluate SRBFs at a deeper level when the BRDF will remove the detail after integration. The coarser SRBFs already provide us with a good approximation of the mean lighting for a larger area on the hemisphere. For this, we correlated the depth of the SRBFs with the lobe size of the BRDF. For example, an entire diffuse SRBF will only require the SRBFs of the top level, whereas very specular BRDFs require the use of all five levels.

We anticipated these unnecessary calculations by preprocessing all the overlapping SRBFs for a fixed set of different BRDF lobe sizes and positions. The overlap information is stored in a texture buffer that is utilized during rendering to only iterate over the overlapping SRBFs. There is a direct correlation between the number of overlapping SRBFs and the BRDF lobe size and position. To allow our overlapping scheme to work optimally for all possible BRDFs, we created bins for different positions on the hemisphere and different BRDF sizes. Each bin represents a distinct BRDF with a certain position with a certain lobe size. Then for each bin, the overlapping SRBFs are preprocessed and stored in the bin. The number of overlapping SRBFs per bin decreases exponentially with respect to a decreasing lobe size. On the other hand, only smaller bin resolutions for theta and phi are needed for the coarser levels. The idea is that they need less detailed overlap information since almost all SRBFs will overlap. This way, the preprocessed overlap data is kept to a minimum, because the smaller BRDF

bins have more detailed bin resolution for theta and phi, but they have much less overlapping SRBFs. To allow for a BRDF to be positioned in-between bins, we calculate overlapping SRBFs quite conservatively. Table 4.2 gives an overview of memory consumption of the preprocessed data and the number of overlapping SRBFs for each bin. Figure 4.10 illustrates how the overlap data is preprocessed.

**Table 4.2: Mean number of overlapping SRBFs per bin. For each level of the environment map grid, a set of bins with different BRDF centers are constructed. The number of bins is dependent on the lobe size of the BRDF. The number of overlapping SRBFs is not drastically increasing because they have a small chance of overlap in the deeper levels.**

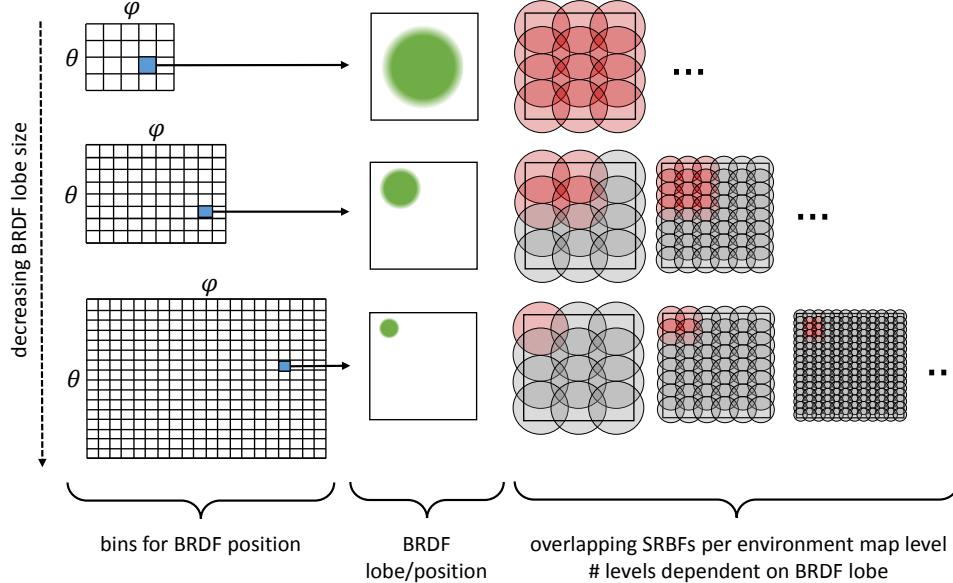
Bin Level	BRDF $\lambda$	# bins	mean # of overlapping SRBFs	max Level
1	2.0	45	60.00	1
2	4.0	153	252.00	2
3	16.0	561	311.55	3
4	32.0	2145	607.82	4
5	128.0	2145	181.83	5
6	256.0	8320	108.668	5
7	512.0	33153	70.4046	5

### 4.4.3 GPU Implementation

The residual transformation algorithm is executed in two phases, for each level of the Healpix hierarchy. First, the influences of all pixels in the neighborhood of each spherical Gaussian are projected onto the Gaussian to extract its amplitude  $\mu$ . This phase is responsible for solving Equation 4.17 of the previous section. Second, a new residue image is calculated by subtracting the residue image of the previous level from the reconstruction of the current level. This requires solving Equation 4.18. When taking a closer look at both equations, it is apparent that the calculations for each SRBF are independent of each other. This allows for an efficient parallel implementation on GPU hardware. This section will provide more detail on how our parallel algorithm is built. First we need to give some background on GPU programming.

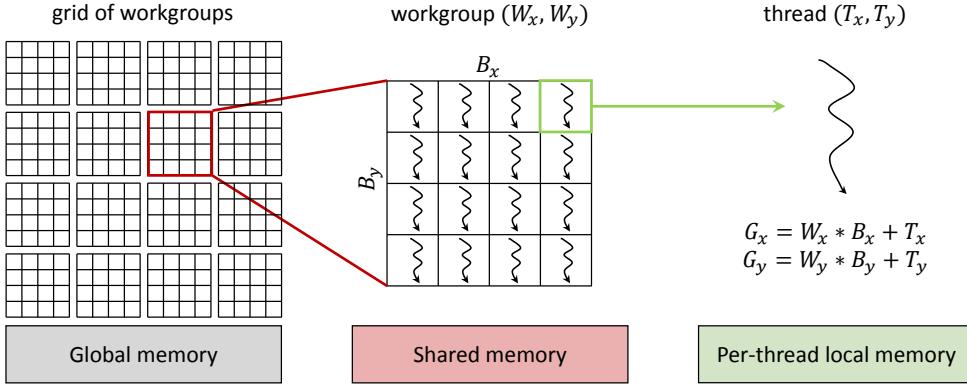
#### 4.4.3.1 GPU Execution and Architectural Model

Even though the best performance is achieved with knowledge of the device and its architectural details, we will give an abstraction of the GPU memory layout that applies to most hardware architectures and is consistent with CUDA [NVIDIA, 2016], OpenGL Shading Language [OpenGL, 2016] and OpenCL [Khrinos Group, 2016]. Figure 4.11 illustrates the abstraction of the GPU architectural model and memory layout. Each GPU has its own number of *independent execution threads*, where each thread is able to execute the same instruc-



**Figure 4.10: Preprocessing of overlapping lighting SRBFs for each BRDF lobe size and position.** A more diffuse BRDF will have more overlap with the lighting SRBFs, but requires less detailed overlap bins. Only the coarser levels of the lighting SRBFs are required, since they are able to approximate the mean value of the lighting function with respect to the integration behavior of the BRDF lobe. The more narrow the BRDF lobe gets, the more detailed bins for the position are required. Furthermore, narrow BRDF lobes can reflect much more detail, thus more detailed levels of the lighting SRBFs are needed. On the other hand, the smaller a BRDF lobe gets, the less overlap it will have on each level of the SRBF hierarchy.

tions in parallel. The set of instructions is called a *shader*. The number of threads are bundled together in blocks of the same size, called a *workgroup*. The size of the blocks are variable, but are limited by the maximum number of threads possible on the underlying hardware. The *global position* ( $G_x, G_y$ ) of each thread is calculated using the block size ( $B_x, B_y$ ), its position in the grid ( $W_x, W_y$ ) and the thread index ( $T_x, T_y$ ):  $(G_x, G_y) = (W_x * B_x + T_x, W_y * B_y + T_y)$ . By dispatching the correct grid of workgroups that matches your problem, each thread can solve its part of the problem. There are three main types of memory. First, each thread is assigned a block of *local memory*, defined by the underlying hardware, which is fast and optimal for local calculations. Second, each workgroup of the grid is assigned a block of memory. This memory is shared between all threads in the block. Generally this block of memory, called *shared memory*, is larger than the local memory and it has the same access speed. It is optimal for caching data required by multiple threads in the same block. Thread synchronization is often required to allow for consistent memory access. Third and last, all threads and the host machine can randomly access the *global memory*—also called video memory—of the



**Figure 4.11:** The set of threads are grouped in a grid of workgroups, where the threads of each workgroup are executed in parallel. Each thread will run the same set of instructions, which is called shader code in GLSL. The thread extracts its global position ( $G_x, G_y$ ) by using his local thread index ( $T_x, T_y$ ) and the known block size ( $B_x, B_y$ ) and workgroup index ( $W_x, W_y$ ).

GPU, which is generally very large but also relatively slow compared to the local and shared memory. Access and especially data transfer to and from the global memory should be held to a minimum. This type of memory is required to share data between the host machine and the GPU, for example textures and buffers.

#### 4.4.3.2 Real-time Multiscale SRBFs

Similar to a CPU implementation, the residual transform is calculated in a top-down manner. It starts from the first level and iterates until the maximum level of detail is reached. Each level alternates between two main shaders: one for the projection of the pixels onto the SRBFs and one for calculating the reconstruction used to update the residue. We allocate one texture for the residue image, which is initialized with the input environment map. A second texture is preprocessed with the directions for each pixel using the 2D to 3D parametrization of the environment map. Furthermore, we initialize buffers for the SRBF parameters and a buffer for fast overlap calculations, as explained in Section 4.4.2. Last, a target buffer is allocated to store the evaluated SRFB weights  $\mu$ . The general outline of the invocation of the shaders is given in Algorithm 1.

The first shader solves for Equation 4.17 and is parallelizable for all SRBFs. The pseudocode is given in Algorithm 2. Each SRBF estimation requires the sampling of a window of neighboring environment map directions around its center. The number of threads are divided over the different SRBFs. For example, for the first level (12 SRBFs) there are 12 concurrent threads that calculate the  $\mu$  by iterating over all overlapping environment map directions. Since the number of SRBFs are rather small in the first levels (12 for the first level), we have

---

**Algorithm 1** Invocation of SRBF shaders for multiscale SRBFs reconstruction.

---

```

 $r_0 \leftarrow \text{environmentmap}$ 
for level = 0 to maxLevel do
    numPixels  $\leftarrow \text{getBoundingBox}(\text{level})^2$ 

    # Phase 1: extracting  $\mu$  (see Algorithm 2)
     $B_x \leftarrow \text{numThreads}$ 
     $B_y \leftarrow \text{numThreads}/B_x$ 
    numWorkgroupsx  $\leftarrow \text{numSRBFs} / B_x$ 
    numWorkgroupsy  $\leftarrow \text{numPixels} / B_y$ 
    bindBuffer( $\text{srbfWeights}_i$ , WRITE)
    dispatch projection(numWorkgroups_x, numWorkgroups_y, B_x, B_y)

    # Phase 2: calculate new residue  $r_{\text{level}+1}$  (see Algorithm 3)
    bindBuffer( $r_i$ , READ_WRITE)
    dispatch reconstruction(width / 32, height / 32, 32, 32)
end for

```

---

not yet reached a full parallel solution. That is why the remainder of concurrent threads are divided over the pixels in the neighborhood. For example, if the underlying hardware supports 1536 concurrent threads, we distribute the threads over all 12 SRBFs of the first level, where each single SRBF is assigned 128 remaining threads. Unfortunately, this introduces an extra challenge, since at the end, the different 128 blocks of pixel calculations need to be merged together. As a consequence, we included an aggregation step where the multiple solutions of the different threads are merged together. Each block of directions will store its result in shared memory (in our example, 128 values for each SRBF). The shader waits until all threads have written their result in the shared memory (thread synchronization). Then, a subset of the threads (12 in our example) are used to aggregate all the different values stored in shared memory into one  $\mu$ , which is stored in the output SRBF buffer.

The second shader will use the estimates of  $\mu$  to reconstruct the detail of the current level. The new residue  $r_{i+1}$  is then estimated by subtracting the previous residue from the reconstructed level. The workgroups are divided over the resolution of the residue image. In our solution, we use a block size of 32. Each thread will evaluate for one direction of Equation 4.18. Since not all SRBFs will overlap the evaluated direction, we use the preprocessed overlap information as shown in Section 4.4.2 to identify the required SRBFs. The pseudocode is given in Algorithm 3. Some examples of environment maps that are approximated with our real-time residual transformation algorithm is shown in Figure 4.12. The reconstruction is done up to a level 6 approximation.

**Algorithm 2** Shader 1:  $\text{projection}(\text{numWorkgroups.x}, \text{numWorkgroups.y}, b_x, b_y)$ 

**Input:**  $w_{xy}$ ,  $\text{thread}_{xy}$ ,  $\text{srbfProps}$  (buffer containing the centers and lobe sizes of all SRBFS),  
 $\text{residueTex}$ ,  $\text{directionTex}$

**Output:** SRBF coefficients for level  $s$  stored a buffer  $\text{srbfCoefs}$

```

1:
2: srbfId ←  $w_x * \text{numWorkgroups.x} + \text{thread}_x$ 
3: pixelBlock ←  $w_y * \text{numWorkgroups.y} + \text{thread}_y$ 
4: [center,  $\lambda$ ] ← texture(  $\text{srbfInfo}$ , srbfId )
5:
6: # Calculate influence of a subset of the neighboring pixels  $R(\text{center}, \text{pixelBlock})$  around
   the center of the SRBF
7: influencePixelsOnSRBF ← (0,0,0,0)
8: for pixelId in  $R(\text{center}, \text{pixelBlock})$  do
9:   pixelVal ← texture(residueTex, pixelId)
10:  pixelDir ← texture(directionTex, pixelId) # direction of pixel value on sphere
11:  influence ← gaussian(dot(pixelDir, srbfCenter),  $\lambda$ )
12:  influencePixelsOnSRBF.xyz ← influencePixelsOnSRBF.xyz + pixelVal * influence
13:  influencePixelsOnSRBF.w ← influencePixelsOnSRBF.w + influence
14: end for
15:
16: # Store the influences of each block in shared memory
17: M[thread.x*numPixelThreads + thread.y] ← influencePixelsOnSRBF
18:
19: syncthreads()
20:
21: # Block aggregation for the pixel influences of SRBF
22: srbfWeight ← (0,0,0)
23: for i in  $F(\text{center})$  do
24:   srbfWeight ← srbfWeight + M[thread.x*numPixelThreads + i]
25: end for
26:
27: # Calculation and store the weight  $\mu$  for the current SRBF (Equation 4.17)
28: srbfWeight ← srbfWeight / srbfWeight.w
29: texture(srbfCoefs, srbfIdx) ← srbfWeight

```

**Algorithm 3** Shader 2: reconstruction(*numWorkgroups.x*, *numWorkgroups.y*, *b<sub>x</sub>*, *b<sub>y</sub>*)

**Input:** The SRBF weights stored in a buffer *srbfCoeffs* and the residue image *residueTex* for a certain level *s* together with its properties stored in the buffer *srbfProps*. An *overlapBuffer* containing preprocessed overlap information for each SRBF.

**Output:** Residue image for level *s + 1*

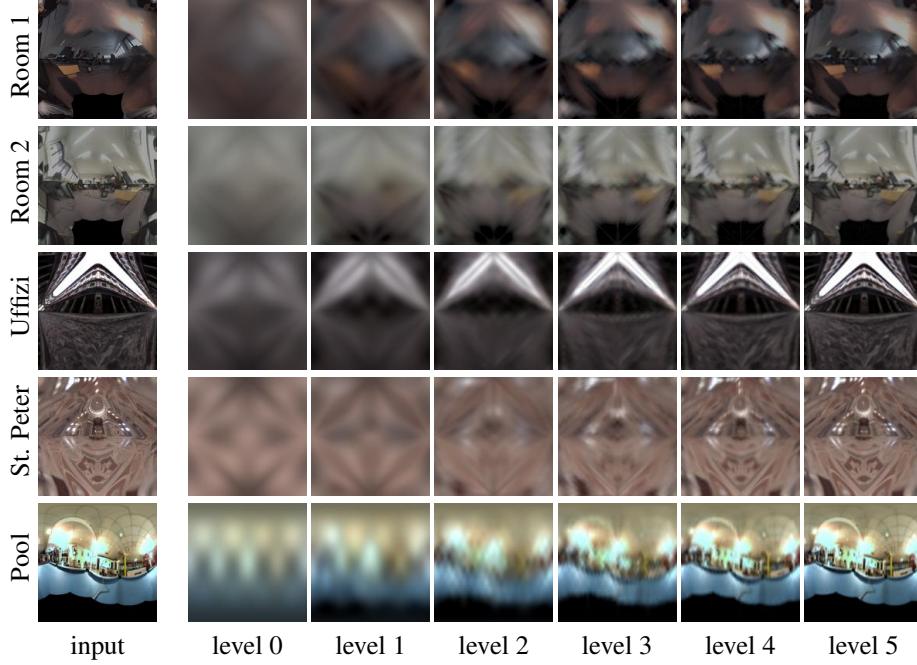
```

1:
2: pixel.xy ← w.xy * numWorkgroups.xy + thread.xy
3: pixelDir ← texture(directionTex, pixel.xy) # direction of pixel value on sphere
4: [θ, φ] ← cartesianToSpherical(pixelDir)
5:
6: # Get bin with overlapping SRBFs for position (θ, φ) and a very narrow lobe of λ = 1000
7: overlappingSRBFs ← getOverlapBin(overlapBuffer, θ, φ, 1000)
8:
9: # Calculate reconstruction
10: reconstructionPixel ← (0, 0, 0, 0)
11: for srbf in overlappingSRBFs do
12:     [center, λ] ← texture( srbfInfo, srbf )
13:     μ ← texture( srbfCoeffs, srbf )
14:     reconstructionPixel ← reconstructionPixel + μ * gaussian(dot(pixelDir, center), λ)
15: end for
16:
17: # Subtract the pixel from the current residue and store as the residue of level s+1
18: texture(residueTex, pixel.xy) ← texture(residueTex, pixel.xy) - reconstructionPixel

```

#### 4.4.4 Peak Detection for High-Frequency Lighting

Remember the subsampling for visibility cones as described in Section 4.3. More diffuse and glossy BRDF lobes have a bigger area on the hemisphere for tracing visibility events and thus require a subsampling of visibility cones to allow for more accurate visibility. The subsampling scheme performs well for large area light sources and low-frequency lighting. For these cases, the size of the visibility cones can remain relatively large. In the case of very bright area lights, subsampling of visibility cones would quickly become too fine-grained to be tractable. We treat these lights as a special case. Their areas are easily identified with a peak detection algorithm. First, the incoming environment map lighting is thresholded, so that only the pixel values with the most intensity and influence remain. Then, a connected component algorithm is used to group pixels into the lights they constitute. Lastly, a number of Gaussians (1-3) are fitted to match the shape of each connected component. We trace a visibility cone for every Gaussian. Figure 4.13 shows some examples of our peak detection algorithm applied on different environment maps. Figure 4.13(b) shows the detected peaks in false colors. These peaks are then approximated using separate lighting SRBFs in

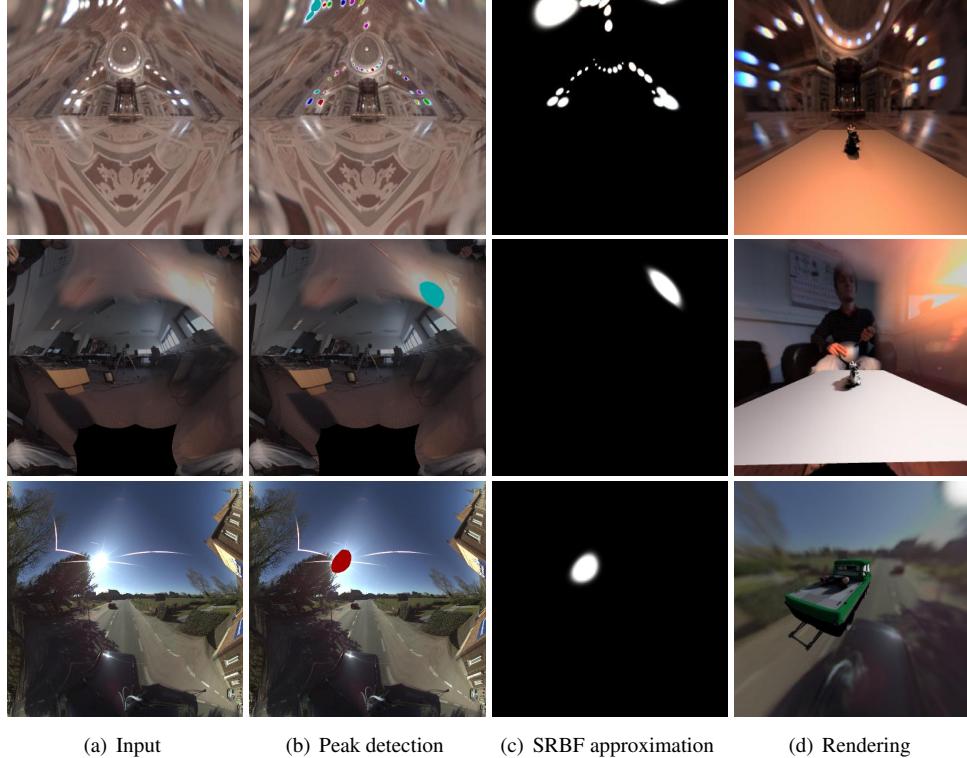


**Figure 4.12: Multiscale residual transformation of different environment maps into SRBF representation from a coarse scale (left) to more refined scales (right). Each level adds extra detail to the reconstruction.**

Figure 4.13(c). These extra lighting peaks, called *free form SRBFs*, are added as a separate phase during rendering where extra visibility cones are traced to accurately render the shadows of the high intensity light sources. In Figure 4.13(d), one can clearly see that the shadows now match the high intensity light sources.

## 4.5 Results

We have designed a custom application to demonstrate the relighting capabilities of our technique. The application was tested on an Intel Core i7 3.4 GHz processor with 8 GB of RAM and a GeForce 770 GTX. The triple product calculation for material, visibility and lighting is done entirely with GLSL shaders on the GPU. The application can be used to place virtual models under real-world lighting conditions. Our peak detection process of Section 4.4 is performed on the distant illumination environment map in real-time. Afterwards, the model can be viewed under these lighting conditions and changes can be made. Unique to our approach is that all factors can be changed simultaneously and in real-time. The lighting can be altered, the model can be moved or deformed and the materials can be edited. If there are



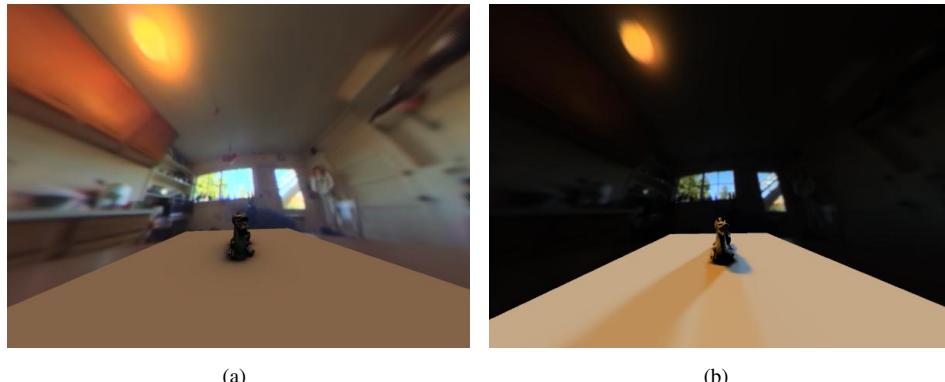
**Figure 4.13: Peak detection.** The input HDR environment map is thresholded to keep only the high-intensity values. These peaks are then detected using connected components. Each connected component is approximated with a number of Gaussians, also called free form SRBFs. These Gaussians are used to accurately render soft shadow for high-frequency lighting.

spatially-varying material parameters in the scene, they can be encoded in textures defined over the virtual model.

Figure 4.14 shows results with animated geometry. This geometry causes dynamic changes in the visibility factor. Our algorithm is able to cope with this in real-time by using voxel cone tracing. The sampling scheme of visibility cones, as explained in Section 4.3, is implemented to quickly get an integrated visibility value over a certain solid angle, defined by the BRDF properties. Extra importance sampling of visibility cones is done by finding peaks in the lighting and is a key feature to efficiently render accurate high-frequency area light sources. Figure 4.14 clearly shows that a change in geometry causes different visibility and that our system is able to render with soft shadows. Figure 4.15 illustrates the importance of peak detection for shadows. The same environment map is used to render both images, but the



**Figure 4.14:** Dynamic visibility. Geometry can be changed using animations or user input. Adaptive cone tracing in the voxel volume based on the subsampling scheme and peak detection allows for real-time rendering of soft shadows.



**Figure 4.15:** Peak detection is mandatory for finding the important visibility cones to trace. Left: traced visibility cones over entire hemisphere using the subsampling scheme of Section 4.3. Further subsampling of the hemisphere with smaller cones will accurately solve the visibility, but will be intractable for real-time applications. Right: extra importance sampling of the hemisphere by using peak detection of bright light sources. By tracing extra cones in the direction of the main area light sources, realistic soft shadows are acquired.

rendering on the right is done with peak detection whereas on the left only the subsampling scheme for visibility is used.

To measure the influence of scene complexity on performance, the scene was rendered with models of varying complexity and lighting conditions with a different number of peaks. An

**Table 4.3: Influence of scene complexity on performance.** All results are rendered at 1280x720 resolution.

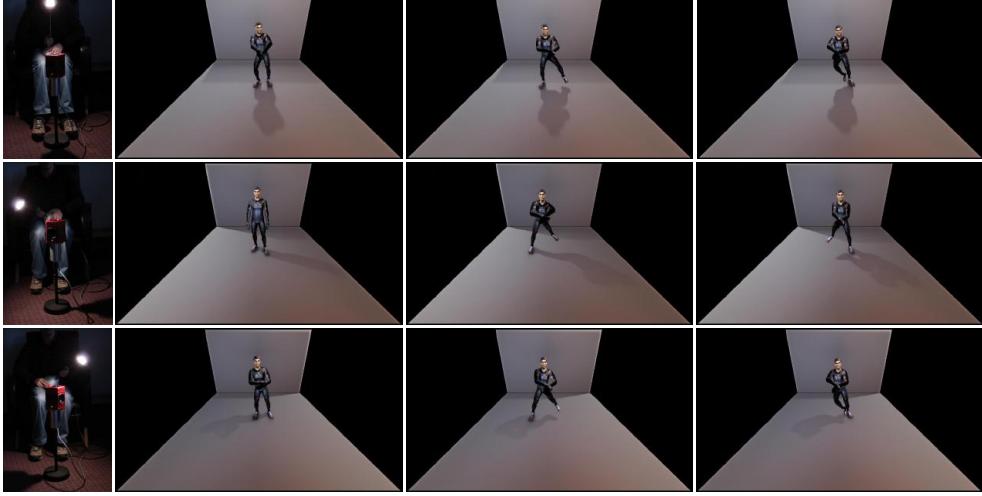
scene	peaks	triangles	FPS
fighter: Fig.	6	16k	43
dragon: Fig.top	6	871k	45
dragon: Fig. bottom	1	871k	75
dragon: Fig. 4.15	11	871k	36
sphere: Fig. 4.2	6	3k	58
fighter/armadillo: Fig. 4.14 top	11	708k	34
dancer: Fig. 4.14 bottom	1	16k	67
dancer: Fig. 4.16	1	16k	75

overview of these experiments is given in Table 4.3. It can be seen that geometric complexity has only a small influence on the framerate. Environmental complexity has a more noticeable influence, but scales well with the number of peaks.

To demonstrate dynamic visibility in combination with interactive live lighting, we placed an omnidirectional camera, such as the Ladybug3 [PointGrey, 2014] of Point Grey, near the desired position of the virtual object in the scene. The residual transformation algorithm is implemented to extract the spherical Gaussians to render with. Now, a virtual model can be relit with one or more flashlights, held before the omnidirectional camera. The results are given in Figure 4.16. Note the high-quality soft shadows resulting from our peak detection algorithm. In Figure 4.17 we show a rendering of BB8 with different properties for mixing Lambertian and Phong materials. The user is able to specify these parameters on-the-fly. The more complex Cook-Torrance model for BRDF is shown on the Dragon dataset in Figure 4.18. The model is illuminated with the Uffizi environment map. We show different results when the Fresnel and Roughness parameters of the BRDF are changed. Last, in Figure 4.19 we show renderings of different realistic models under different environment lighting. We demonstrate the rotation of the complex environment maps and the effect on the reflections and shadows on the rendered result.

## 4.6 Discussion

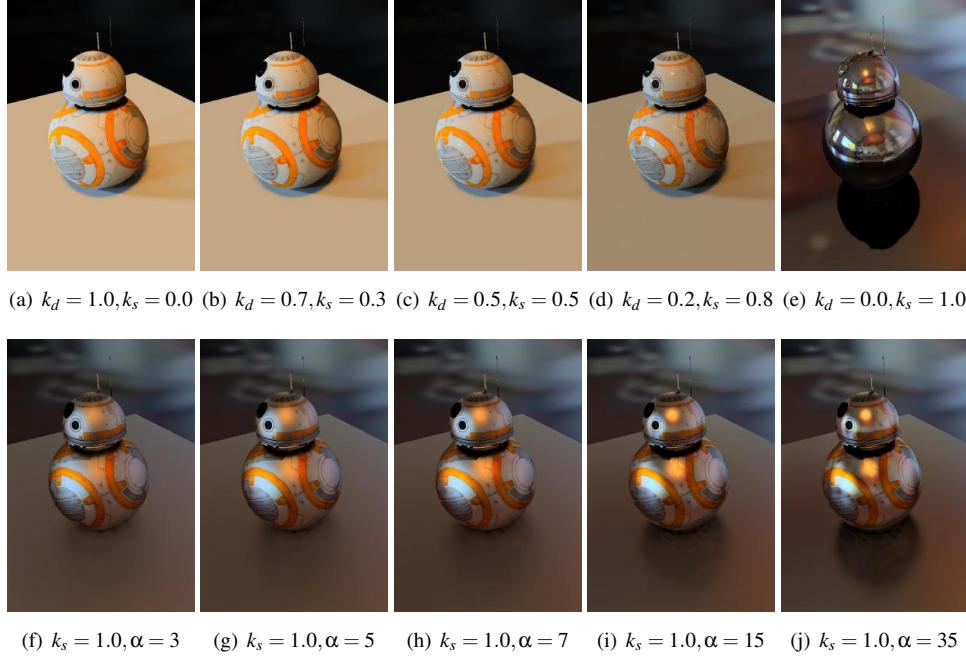
Our results clearly show improvements in terms of time complexity compared to the generalized wavelet theorem of the previous chapter. We have sacrificed some quality performance in order to achieve this. The greatest sacrifice is the sampling of the visibility map. This is mainly a design decision. Wavelet approaches are not able to sample the visibility factor on-the-fly. They rely on preprocessing the visibility map in the pixel domain for each vertex, before they are transformed to the wavelet domain. This is known to be highly accurate, but



**Figure 4.16: Dynamic lighting.** The environment lighting is captured using the Ladybug3 [Point-Grey, 2014] omnidirectional camera. Our peak detection algorithm is performed in real-time and passed onto the renderer. The virtual object is now rendered with the lighting conditions from the captured frames. Note how the virtual shadow positions are aligned with the real light source.

not very fast. The same is true for retrieving wavelet representations for the BRDF factor, where the 2D BRDF slices are first sampled in the pixel domain. Similar approaches could be performed with SRBFs, but are not very useful for interactive applications. With our SRBF framework we tried to provide a range of possibilities, where one can choose its desired time and quality performance.

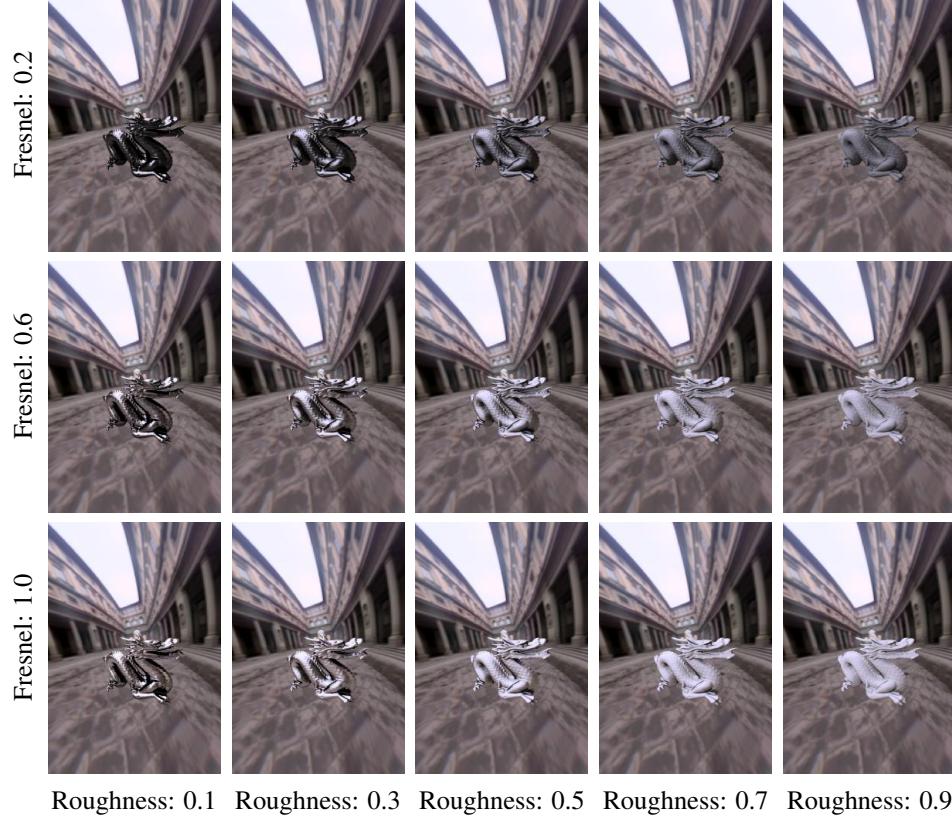
To allow for real-time frame rates, our SRBF approach samples the visibility in real-time using cone tracing. The quality of the visibility factor is highly dependent on the number of cones traced. The number of cones can be greatly reduced by importance sampling the BRDF lobe. Although it is not fully quantified, we have empirically observed that we achieve good results with approximately 7 visibility SRBFs. In the limit we can achieve similar quality compared to a ray traced visibility map. This can be done by increasing the number of visibility cones, at the cost of extra processing time. This flexibility is simply not possible when using wavelets. In addition, the environment map uses slightly more SRBF coefficients compared to the wavelet approach. Nevertheless, using SRBFs will also diminish some bottlenecks that have no influence on the quality of rendering. For example, SRBFs are rotation in-variant. The complex rotation operator, as proposed by Wang et al. [Wang et al., 2006], is no longer necessary. The rotation of the environment lighting is now applied by simply rotating all its SRBF centers. Furthermore, the use of SRBFs for BRDF representations is a giant leap forward, because most BRDFs can be represented with only one or two SRBF lobes.



**Figure 4.17:** Rendering surface materials by changing parameters of the Lambertian and Phong BRDFs. The top row shows different mixes of Lambertian ( $k_d$ ) and Phong ( $k_s$ ) BRDFs. The bottom row shows the Phong BRDF with different shininess ( $\alpha$ ) properties.

Wavelets on the other hand require on-the-fly sampling of BRDF slices in the pixel domain, before they are transformed to a wavelet tree. Although efficient caching of BRDF slices can reduce some of the preprocessing time, the overhead is entirely removed when using SRBFs.

If we look at the time performance of the triple product integral, we can observe that there are different elements that have an influence on the time performance. The amount of SRBFs to evaluate, is mainly dependent on the type of BRDF. The lobe of the BRDF determines the number and width of the visibility cones (Section 4.3.3). Visibility SRBFs are the most expensive to evaluate, since they require cone tracing in the voxel volume. Reducing the amount of voxel cones will increase the rendering speed. In addition, the BRDF lobe determines the overlapping environment lighting SRBFs. The overlap calculation is inspired by the hierarchical approach of wavelets (Section 3.4.2). Here, it is mainly used to find the overlapping environment SRBFs for certain BRDF lobe sizes. This overlap information is preprocessed in a texture on GPU. The binding coefficients of the overlapping SRBFs are evaluated on-the-fly. To summarize, it is clear that the BRDF lobe dictates the amount of triple product evaluations. In general, one can assume that adding an extra BRDF to the material property of

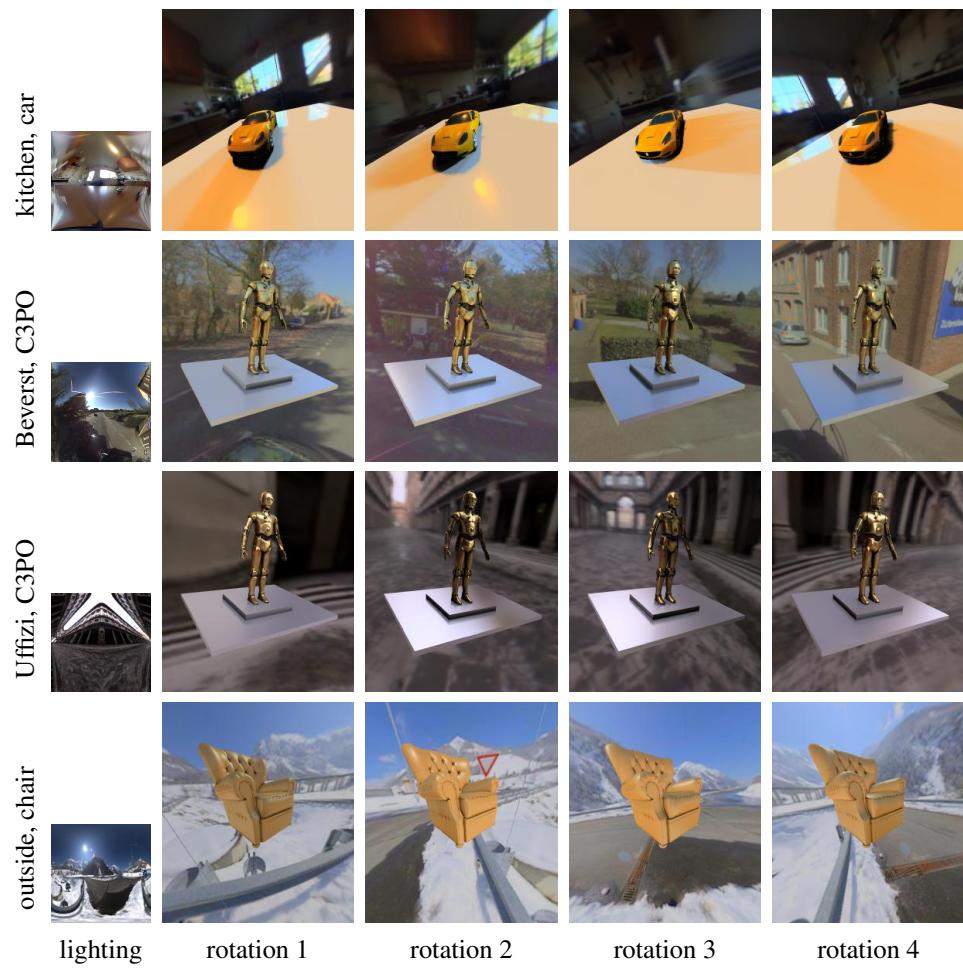


**Figure 4.18: Rendering with Cook-Torrance BRDF. Dragon dataset rendered under Cook-Torrance reflection with varying roughness  $R$  and Fresnel  $F$ . Best viewed digitally.**

the object will double the amount of time to render the image. Overall, we have learned that the choice of these settings and basis functions are mainly dependent on the target application and on the time versus quality performance the application requires. Some applications will require great compression performance, for which wavelets are ideal. Other applications require real-time previews and interactions, for which SRBFs are the better choice.

## 4.7 Conclusion and Future Work

This chapter presented a unique combination of techniques to allow for real-time triple product rendering where all three factors are dynamic. Spherical Gaussians were used to represent the incoming light, the BRDF lobes and the visibility over a solid angle. This representation enables efficient light integration on the GPU. Important regions of visibility detail are cone traced using an adaptive subsampling scheme. Furthermore, peak detection is used to trace complex area shadows. We have shown that the three factors of material, visibility and lighting are dynamic and can be updated in real-time. We therefore combine all required features as listed in Table 4.1, compared to previous techniques. We explained how we use cone tracing to support dynamic scenes. Note that our technique inherits some of the difficulties of cone tracing with respect to the handling of large scenes, as well as thin geometry. This is because our current 3D texture voxelization is not sparse. This limitation will be overcome by the new partially resident textures on new GPUs [Sellers, 2013] or by applying state-of-the-art lossless compression of the 3D texture [Guthe and Goesele, 2016]. A second limitation is the number of visibility cones. Each new visibility cone introduces a new triple product calculation. The visibility subsampling scheme, together with the peak detection algorithm, will limit the number of visibility Gaussians. However, environment maps filled with different small HDR light sources will still influence the rendering performance.



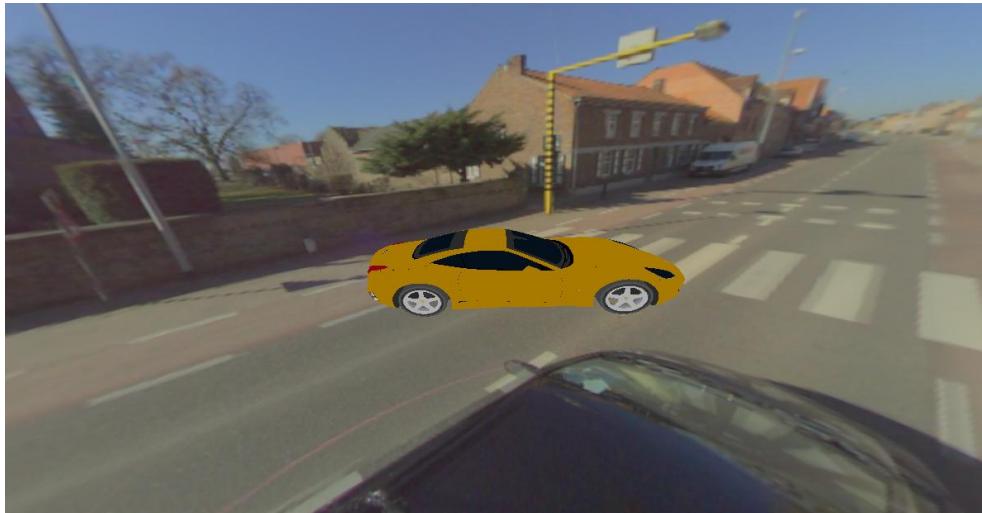
**Figure 4.19:** Rotating the environment lighting during rendering. Different models are rendered under different environment lighting. Rotating the environment lighting will alter the reflections and shadows in the scene accordingly. Best viewed digitally.

# Chapter 5

---

## Use Case: Relighting of Virtual Objects

<b>5.1</b>	<b>Augmented Reality Application.....</b>	<b>89</b>
<b>5.2</b>	<b>Related Work.....</b>	<b>91</b>
<b>5.3</b>	<b>Relighting in Augmented Reality .....</b>	<b>92</b>
<b>5.3.1</b>	<b>Omnidirectional Video Capture and Rendering.....</b>	<b>92</b>
<b>5.3.2</b>	<b>Camera Tracking using Structure-from-Motion .....</b>	<b>93</b>
<b>5.3.3</b>	<b>Real-Time Augmented Rendering .....</b>	<b>96</b>
<b>5.4</b>	<b>Results.....</b>	<b>97</b>
<b>5.5</b>	<b>Conclusion.....</b>	<b>99</b>



(a)



(b)

**Figure 5.1: Real-time augmented rendering of virtual objects.** Most augmented reality applications lack the photorealistic rendering of lighting conditions and reflections and rather use a more flat shading approach similar to the rendered car in (a). This dissertation would like to use the lighting conditions of the real scene and apply correct materials and lighting conditions to the virtual object with accurate reflections and shadows. An example of the same car is shown in (b). Notice the reflections of the building in the metallic surface of the car and how the shadows are cast by the sun.

In the previous chapters, we have developed different algorithms for accurate and efficient triple product rendering with different basis representations. Our high-order wavelet or spherical radial basis evaluation methods can be applied to many different applications. An important conclusion of the previous chapters was that an appropriate basis should be assigned, based on the needs of the application, depending on the required quality, interactivity or compression performance. Relighting applications are the main focus of this dissertation. In this chapter, we will show how we can utilize our novel triple product rendering framework in a relighting application for virtual objects, more specifically in an *augmented reality application*.

## 5.1 Augmented Reality Application

In an augmented reality application, a direct or indirect view of the real-world environment is overlaid (augmented) with computer generated content. In general, this is done in real-time. Up until now, augmented reality applications often lack realistic lighting [Haller, 2004]. This is because no lighting information is present. An example of a non-photorealistic augmented video is shown in Figure 5.1(a).

With the introduction of new technologies like the Microsoft Hololens [Microsoft, 2016], Oculus Rift [VR, 2012] or a 360° projection screen [Raskar et al., 1998; Dumont et al., 2011], augmented reality applications will become much more important in the future. These devices are developed to increase the immersive experience for users. If we would use fairly flat shading without any accurate lighting effects in an augmented reality application on such devices, the virtual objects will not blend in, causing a negative effect on the immersive experience for the users. These new devices achieve an immersive experience by adding view-dependent effects. The content changes, based on the direction the user is looking at. In the case of an augmented reality application, they fail to include all view-dependent effects. Lighting and reflections are also an important view-dependent effect and change based on the view direction of the user. Nowadays, relighting of virtual objects with accurate view-dependent lighting effects can only be done using a time consuming offline photorealistic renderer or by preprocessing scenes and lighting conditions. This is, however, not applicable to augmented reality applications where interactivity is key. A user needs to be able to interact with the objects, as well as change shape, material properties and lighting conditions dynamically.

In contrast, we have developed a novel interactive SRBF triple product renderer that is able to simulate all direct lighting effects for virtual objects where the geometry, materials and environment map are fully dynamic. However, in order to render, it requires information about the lighting conditions of the real scene, which is not freely available. Is it possible to render objects with our interactive renderer, where the lighting is consistent with the real



**Figure 5.2: Illustration of real-time augmented rendering of virtual objects in an omnidirectional environment. The 360° video is used as an environment map for realistic lighting.**

world and where the virtual objects are seamlessly integrated into the augmented environment? One possible solution is to find the main light source of the scene using the shadows in the original image [Arief et al., 2012]. This way, realistic shadowing can be achieved, but all high-frequency lighting and reflection information is lost. Fortunately, the new ways of showing an augmented reality application in a head-mounted display or on a 360° projection screen require real-time omnidirectional video data. Omnidirectional video is an emerging medium that gives viewers a 360° panoramic experience. Each frame of an omnidirectional video can be seen as an environment map lighting for one position in the scene. We can use the existing environment lighting from the captured 360° video to render the virtual object much more realistically.

In this chapter, we explore how to use an omnidirectional video as environment lighting for our triple product renderer. The main goal is to render virtual objects with lighting that is consistent with the captured video and to make objects more seamlessly integrated in the real environment. An example is shown in Figure 5.1(b), where the same car is rendered with more realistic lighting conditions, including accurate reflections and shadows of the real scene. To achieve this, we first reconstruct the trajectory of the camera of an omnidirectional video using structure-from-motion. The structure is used to accurately track the camera poses in a large environment. Extending this omnidirectional content with camera position tracking creates a global coordinate system in which virtual objects can be augmented. We show how we leverage the tracking information and use it to feed the correct environment frame to a realistic renderer. This is necessary to synchronize the lighting information with the exact position of the augmented virtual object. The correct environment map frame is passed to our SRBF triple product renderer which ensures high quality rendering with view-dependent high-frequency lighting and detailed spatial and angular reflections. Our system is demon-

strated with an application in which an augmented vehicle can be controlled through an urban environment (Figure 5.2).

Possible applications are real-estate, interactive worlds, architectural modeling and advertisement. For example, imagine a walk-trough of a house you wish to buy using a head-mounted display. Our augmented reality application would make it possible to add new virtual objects, e.g. new interior designs, which are seamlessly integrated with the real video of the house. The customer will have a more realistic preview of the new possibilities and visual outlook of the house.

## 5.2 Related Work

Debevec [Debevec, 1998] was one of the first to propose image-based lighting (IBL), where the capture of real-world lighting is used to render virtual objects. The environment lighting is described as a single light probe image. An extensive overview of photorealistic and non-photorealistic augmented reality applications is made by Haller [Haller, 2004]. They compared both approaches using ARToolKit markers to track the camera's pose. In the photorealistic approach, they show how to use shadow volumes and bump mapping.

Agusanto et al. [Agusanto et al., 2003] showed an application where image-based rendering techniques are used to incorporate virtual objects in augmented reality content using environment illumination maps. They did not capture any environment map, but used existing single shot light probes. The disadvantage is that they are not able to render with interactive and dynamic lighting.

Other techniques [Kanbara and Yokoya, 2004; Arief et al., 2012] have focused on shadow effects in augmented reality applications. They identified the most important light sources in the light probe and used them to cast shadows of the augmented objects to make them consistent with the real world. In terms of other lighting and reflectance effects, they lack visual realism.

Grosch [Grosch, 2005] showed how to make the light of a room consistent with the rendered virtual objects by reconstructing the geometry of a single light probe. The reconstruction makes it possible to add new light sources to the light probe, as well as to place new objects in the room and to project the light probe for different positions in the room. They mainly focus on consistency of the light conditions in a room, but not on how to realistically render objects with these new light conditions. Additionally they only use one light probe for the entire scene, which makes their algorithm not applicable to larger scenes. Later, Grosch et al. [Grosch et al., 2007] studied how to use irradiance volumes to place virtual objects in a real-life Cornell Box. The external illumination is captured with a HDR camera and a fisheye

lens. Direct and indirect lighting effects are simulated for the virtual objects. The rendering is done at interactive frame rates, but the geometry of the objects cannot be changed easily, which makes the rendering of dynamic scenes impossible.

Papagiannakis et al. [Papagiannakis and Foni, 2005] and Gierlinger et al. [Gierlinger et al., 2010] have integrated the full precomputed radiance transfer (PRT) [Sloan et al., 2002; Ng et al., 2003, 2004] in a mixed reality application using HDR input images. They used spherical harmonics to precompute the three factors of the radiance transfer. They were able to render high quality and realistic lighting in real-time, but required a great amount of pre-computation. Since precomputation of visibility is necessary, the results are limited to static scenes. Additionally, the use of spherical harmonics are suboptimal for high-frequency lighting conditions.

## **5.3 Relighting in Augmented Reality**

The goal of this use case is to augment omnidirectional video (ODV) with new virtual objects that are realistically lit by their environment. The user must be able to interact with the video by moving around the virtual object, while at the same time the lighting and reflectance conditions stay consistent with the captured environment. This requires a system with the following three stages:

1. Stitching and rendering of ODV;
2. Offline reconstruction of the camera trajectory of ODV using structure-from-motion;
3. Real-time realistic rendering of virtual objects using the ODV for lighting.

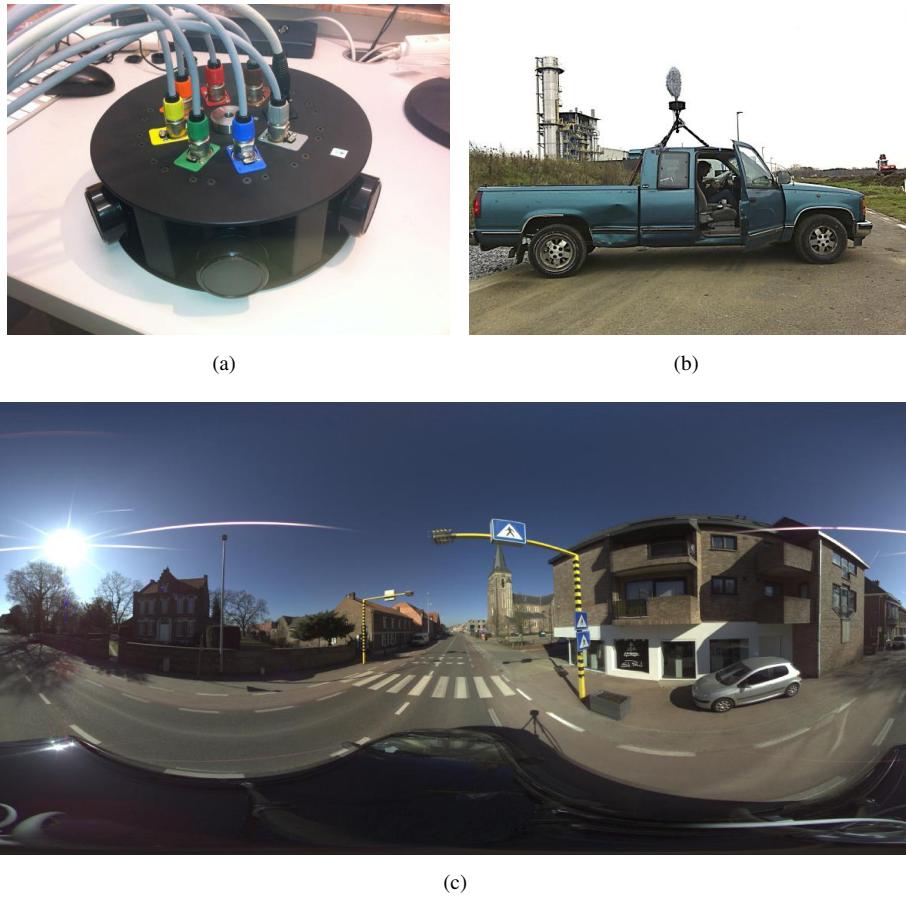
The three stages will now be explained in more detail.

### **5.3.1 Omnidirectional Video Capture and Rendering**

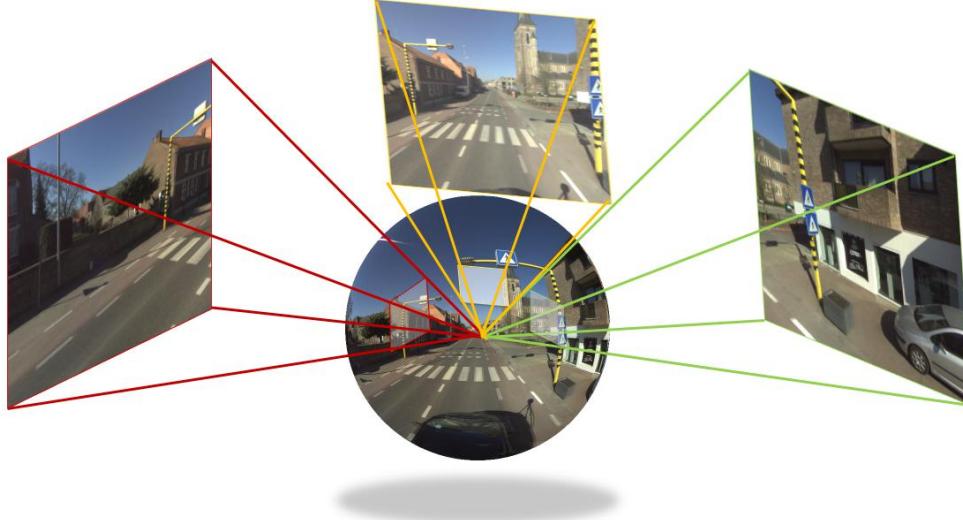
Capture is performed with a custom designed camera, built with six cameras at  $60^\circ$  intervals with approximately 50% coverage between adjacent cameras. The omnidirectional camera is depicted in Figure 5.3(a). Each camera has a resolution of  $1600 \times 1200$ . All six cameras are synchronized and capture at 25 frames per second. Next, we mounted the camera on top of a vehicle and drove through a local city, which is demonstrated in Figure 5.3(b). The stitching and rendering of the different camera frames are performed in real-time on the graphics card. Figure 5.3(c) depicts an example of such stitched frame in an equirectangular representation. The rendering of such frames requires the warping of the equirectangular stitched frame onto a sphere where the virtual camera is positioned within the sphere. This way, the camera can be rotated in any direction, so that the user can freely experience a  $360^\circ$  walk-through in the video environment, as illustrated in Figure 5.4. The process of capturing the six cameras, stitching them and rendering them is performed in real-time.

### 5.3.2 Camera Tracking using Structure-from-Motion

The ability of adding new virtual objects in the captured video of the previous section requires the pose estimation of the omnidirectional camera throughout the sequence. This step is of significant importance and aims to determine the alignment between synthetic objects and the real world in order to render them at the correct position. Once the alignment is complete, an appropriate omnidirectional video frame can be identified for each virtual object, which will serve as environment lighting so that realistic lighting can be achieved. This section will explain how to adapt a structure-from-motion algorithm to make it work with omnidirectional



**Figure 5.3: Capture of omnidirectional video.** (a) Our custom designed omnidirectional camera consists of 6 cameras, all capturing at 25 fps with a resolution of  $1600 \times 1200$ . (b) The omnidirectional camera mounted on a vehicle to captures the streets of an urban environment. (c) The stitched panoramic frame using all six cameras, which is generated in real-time.

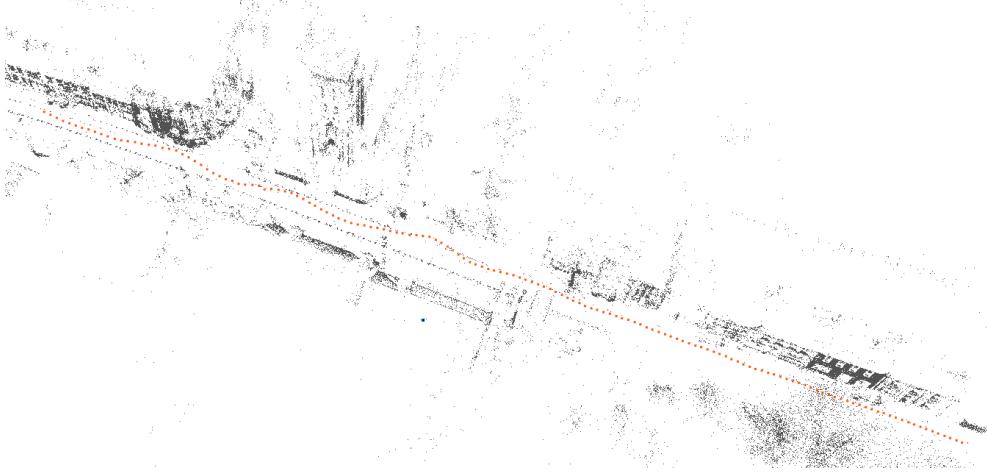


**Figure 5.4:** Rendering of omnidirectional content. A virtual camera (red, yellow or green) can be placed in the spherical warped ODV frames. This gives the viewer a full 360° interactive experience, since one can freely look around with the camera.

content. The result is accurate tracking information of the omnidirectional camera in a large environment.

The first step of a structure-from-motion algorithm is feature tracking. Since the stitched omnidirectional video frames still have some ghosting artifacts which will possibly degrade the tracking accuracy, feature tracking was done on the individual camera images (the frames from the cameras that make up the omnidirectional camera). Since the poses of the individual cameras relative to the center of the ODV camera are already known (this information is extracted during ODV camera calibration), we do not need to estimate the pose of each individual camera for each frame. Instead, the pose of the complete ODV camera is estimated using only the image information of the individual cameras, resulting in a more accurate pose estimation.

Before applying a feature detection algorithm, the radial distortion in the images of the individual cameras needs to be removed. Then, we apply a Shi-Tomasi corner detector on the undistorted input images [Shi and Tomasi, 1994]. Only features with quality are stored. The features are then tracked to the next frame of the corresponding camera using Lucas-Kanade optical flow tracking [Lucas and Kanade, 1981]. We chose the Lucas-Kanade tracker instead of a descriptor based matching method, because we want the number of frames in which a feature is tracked to be as high as possible. Longer feature tracks result in more shared

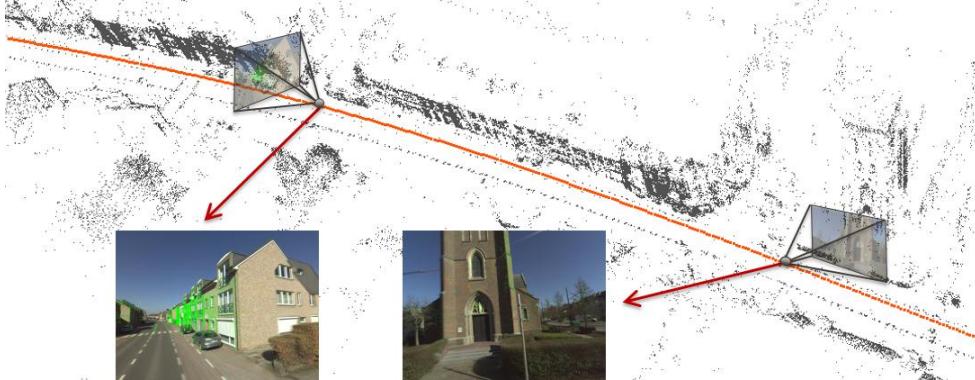


**Figure 5.5: Initial reconstruction of the camera poses using visual odometry without sparse bundle adjustment. The accuracy of the poses is very poor.**

information between the poses of the ODV camera and thus result in a more stable pose estimation. In case of short feature tracks, there is less shared information and the estimated poses often show more jitter when rendering synthetic objects. The optical flow algorithm, however, can result in erroneous matches. Therefore, we use the trifocal tensor constraint to filter out incorrect matches. The trifocal tensor is calculated for three adjacent ODV frames with a RANSAC approach [Fischler and Bolles, 1981]. The input of the RANSAC step consists of the features that are visible in all three frames. A subset of these features is used to determine a trifocal tensor, and the trifocal tensor is used to check whether a feature is an inlier or not (the position in the third frame needs to be close to the estimated position).

For each feature a SIFT descriptor [Lowe, 1999] is extracted, necessary to match the features between the cameras. Feature matching between the cameras allows us to connect shorter feature tracks into longer tracks. This is especially useful when a feature leaves one camera and shows up in another one. A cross-check is applied to make sure that the features match in both directions. The remaining features are checked against the epipolar constraint.

Another advantage of estimating the trifocal tensor for outlier removal is that one can extract the fundamental matrices (from camera 1 to 2 and from camera 1 to 3). These encode the relative position and orientation of the cameras and thus can be used to determine an estimate of the poses of the ODV camera. This process is similar to visual odometry (VO) [Scaramuzza and Fraundorfer, 2011; Fraundorfer and Scaramuzza, 2012] and, as is to be expected with VO, the resulting positions are far from optimal since the pose of the camera is only estimated in relation to the previous pose. The initial estimation of poses is shown in Figure 5.5.



**Figure 5.6: Structure-from-motion is used to reconstruct the trajectory of the road (depicted in orange). The trajectory is used to align the rendered objects with the real-world scene.**

A better option is to globally estimate the pose using the information gathered in all ODV frames. This is done by applying bundle adjustment.

Bundle adjustment simultaneously updates the poses of the cameras and the 3D world positions to minimize the mean error between the reprojected world points and their corresponding features. The error metric is defined as a function of the distance between the position of the feature and the reprojection. The sparse bundle adjustment implementation of Lourakis and Argyros [Lourakis and Argyros, 2009] is used. As input, the algorithm requires initial estimates of the poses and the world positions of the features. These initial world positions are obtained by doing a SVD based triangulation, followed by an optimization step that also reduces the reprojection error.

The results of bundle adjustment are refined ODV camera poses and a sparse 3D structure that aligns to objects in the world. Figure 5.6 shows the poses and the structure of the track after bundle adjustment, as well as examples of the structure rendered on top of the omnidirectional video using an estimated pose. One can clearly see that the 3D points are aligned with their corresponding features in the video and that the estimated positions of the camera better represent the true positions of the camera, compared to the initial poses of visual odometry in Figure 5.5.

### 5.3.3 Real-Time Augmented Rendering

High quality and realistic rendering of objects will drastically increase the immersive experience of augmented reality applications. To achieve this, the rendering requires detailed natural lighting and realistic reflectance of materials. View-dependent effects—like a change in

eye direction across surfaces—will improve visual realism. This dynamically varying viewpoint will require a reflectance representation defined in 6D (light direction, view direction and surface position). Since detailed variations in reflectance will visually impact the end result, the 6D functions needs to be evaluated densely.

The tracking information of the previous section provides us with a global coordinate system of the video. We can now place new virtual objects in the same coordinate system, for example a car on the road of the urban environment. To render the car realistically in the environment, we need proper lighting information at the position of the car. The omnidirectional video frames can be seen as distant lighting environment maps. We use the 3D position of the car to assign an appropriate omnidirectional frame of the video. The assigned omnidirectional video frames can then be used as environment lighting for rendering the car, using our interactive spherical radial basis function triple product renderer of Chapter 4. The omnidirectional video frame, represented in the pixel domain, is transformed to a SRBF representation using the GPU implementation of our multiscale residual transform. This allows for the user to interactively change the position of the car in the real world and directly see the change in reflections and shadows on the object.

Notice that, for now, the environment map is handled as distant lighting. The reflections are therefore most accurate when the object is placed near the reconstructed path of the camera. In the future, when the user wants to deviate from the reconstructed trajectory, interpolation between multiple omnidirectional frames is required.

## 5.4 Results

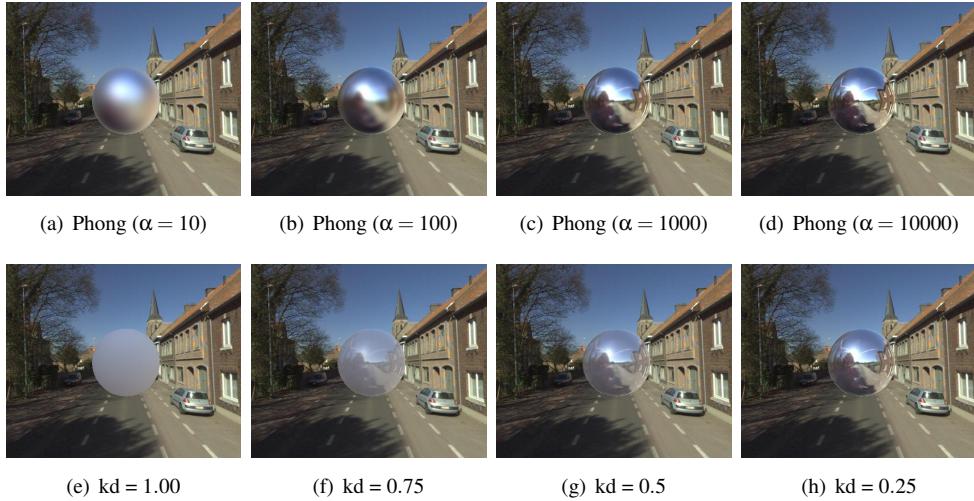
Our real-time augmented renderer is implemented on a system with an Intel Xeon<sup>TM</sup> dual six core processor and an NVIDIA GeForce GTX 780 graphics card. The stitching and rendering of the omnidirectional video is performed on the GPU in GLSL shaders. We created a viewer where the user can walk through the video and can manipulate the viewing direction of the camera. This viewer can be outputted to a normal screen or to more immersive hardware like Oculus Rift or a omnidirectional cave [Dumont et al., 2010].

Structure-from-motion is done offline as a preprocessing step. The focus of this step lies on quality and accuracy and not so much on time performance since the application only requires the reconstructed trajectory of a video that is not captured at runtime. The transformation of the omnidirectional video frames to spherical radial basis functions and the triple product rendering is done entirely on GPU using the algorithms from Chapter 4.

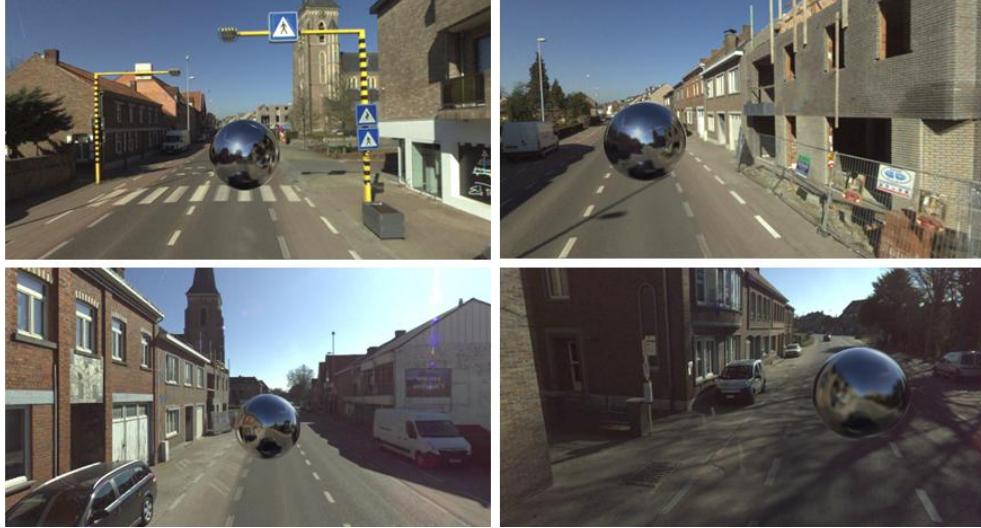
A main advantage of using SRBFs as a representation is that BRDFs can easily be approximated with few SRBFs which can be reconstructed at real-time using only the parameters

of the BRDF model. For each point in the virtual object, the system only needs to know its BRDF parameters and then an appropriate BRDF radial basis function can be assigned. We have support for the Lambertian, Phong [Phong, 1975], Blinn-Phong [Blinn, 1977], Ward [Ward, 1992] and Cook-Torrance [Cook and Torrance, 1981] BRDF models. In Figure 5.7, we show a sphere model inside our captured environment, all rendered with different material parameters. The top row of the Figure shows the Phong BRDF with an exponent parameter ranging from 10 to 10000. The bottom row shows the mixing of a specular Phong BRDF with a Lambertian BRDF. Now the percentage of mixing the diffuse Lambertian is ranging from 0.25 to 1.00.

Figure 5.10 depicts the results of a more realistic model, i.e. a vehicle. A model of a Mercedes with texture mapping is inserted into the scene. Furthermore, we assigned a realistic material with a mixture of BRDFs to the model. The figure clearly shows accurate view-dependent lighting with detailed spatial and angular reflections when the car is moved in the environment. In Figure 5.8 we show the same sphere and assigned a high specular material to make sure all lighting effects of the environment onto the sphere can be seen. We chose a Phong BRDF with a big exponent factor ( $N = 10000$ ). It is now possible to perceive a change in reflection when the sphere is moved around in the urban environment.



**Figure 5.7:** Rendering with different material properties. **Top row:** Phong BRDF with an exponent  $\alpha$  ranging from 10 to 10000. **Bottom row:** mixing of materials with a  $kd$  percentage of a Lambertian BRDF added to a  $(1 - kd)$  percentage of a Phong ( $\alpha = 10000$ ) BRDF.



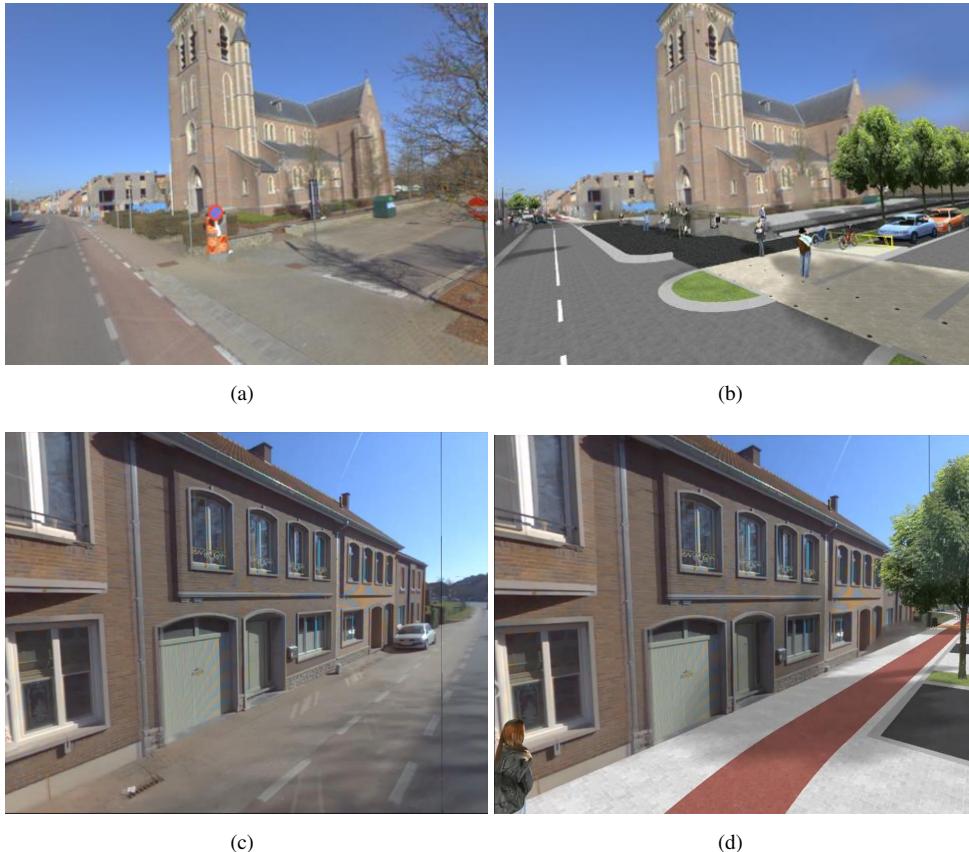
**Figure 5.8: Rendering of a sphere. Different positions of the sphere will result in different lighting conditions.**

## 5.5 Conclusion

In this chapter we presented an interactive relighting use case for augmenting omnidirectional video with interactive virtual objects rendered with realistic lighting and reflectance properties. The method used offline feature tracking to align positions of virtual objects with positions in the real-world. We proposed the use of omnidirectional video frames as environment lighting for the virtual objects and showed how to utilize tracking information to synchronize the correct environment map with the virtual objects.

Furthermore we explained how our novel SRBF triple product rendering framework can be utilized to transform environment lighting, BRDF properties and visibility of the augmented objects to a spherical radial basis functions and how to use them to render a virtual objects in real-time with realistic reflectance and lighting conditions.

The use case in this chapter is limited to distant light only. In the future, near-field lighting can be sampled similar to the cone tracing algorithm of the visibility factor. The rendering of distant light requires the virtual objects not to deviate too much from the reconstructed trajectory. To solve this, future work will require the interpolation of multiple omnidirectional video frames to extract the lighting condition of any physical position in the world. Further development of the structure-from-motion step will result in more dense geometry. This is shown in Figure 5.9. The dense geometry can be used to cast shadows onto the virtual objects



**Figure 5.9:** Demonstrator for merging virtual and real content in the context of the redesign of public roads (AVIE demonstrator 2014). A basis mesh reconstruction is used to accurately cast shadows from virtual objects onto real objects and vice versa.

as well as correct occlusions. In addition, the augmented objects can also cast more accurate shadows on the real world.

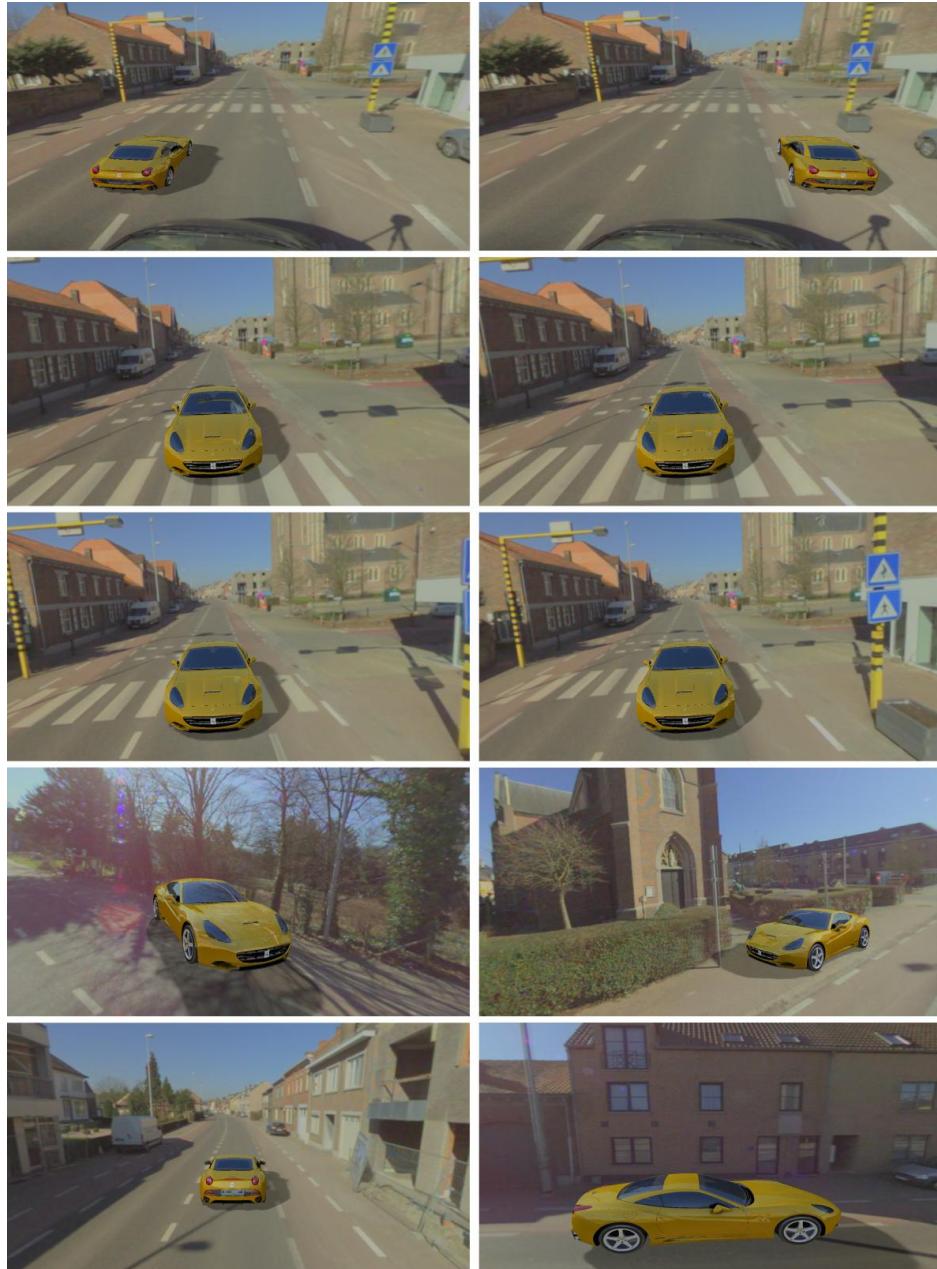


Figure 5.10: Rendering of a user-controlled vehicle in an urban environment. Row 1 shows the change in reflection when the user is horizontally moving the vehicle. Row 2 and 3 depict the change in reflection when driving the car forward. Row 4 and 5 show the car from different angles. Best viewed digitally.



# Chapter 6

---

## Use Case: Relighting of Real Objects

<b>6.1</b>	<b>Related Work</b>	<b>104</b>
<b>6.2</b>	<b>Inverse Rendering Application</b>	<b>106</b>
6.2.1	Background	107
6.2.2	Hierarchical Refinement and High-order Wavelets	108
6.2.3	SRBF Triple Product in Inverse Rendering	110
6.2.4	Near-Field Lighting	113
6.2.5	Results	117
<b>6.3</b>	<b>Intrinsic Image Decomposition Application</b>	<b>121</b>
6.3.1	Background	121
6.3.2	SRBF Triple Product Gradients Rendering	122
6.3.3	Results	124
<b>6.4</b>	<b>Conclusions</b>	<b>125</b>

Another key research topic, besides the relighting of virtual objects, is the relighting of real objects from photographs. It allows for interesting applications in the game industry, movie industry but also for augmented reality. Our new approach for renderer with SRBFs or high-order wavelets enables multiple interactive applications for the relighting of virtual objects. This raises the question whether we can also apply this to real objects. Imagine the same interactivity as the augmented reality use case but now with existing objects extracted out of photographs taken by the user itself. Before we can start changing parameters of the rendering equation, we first need to extract the basis parameters out of single or multiple images. This is the field of inverse rendering or intrinsic images. Most of the state-of-the-art techniques come down to defining a cost function and minimize this cost function using an optimization step. In this chapter we investigate if our novel triple product rendering framework can be used in existing inverse rendering and intrinsic images problems and how it would potentially improve existing approaches both qualitatively, but since our renderer is real-time, especially in time performance.

We implemented two techniques. The first technique is a pure multi-view inverse rendering approach designed by Haber et al. [Haber et al., 2009] which is built upon the linear BRDF factorization of Weistroffer et al. [Weistroffer et al., 2007] and represents the lighting, visibility and BRDF information in the all-frequency Haar wavelet domain. We show how our SRBF renderer can be used instead of Haar-wavelets and how this will greatly increase the time performance. Furthermore, our GPU implementation will allow for a per pixel optimization in the UV map domain, where the original technique can only estimate material properties per vertex. A UV-map representation will allow us to visualize the estimated data in any 3D editing software. The second technique is an implementation of the intrinsic image decomposition problem as stated by Barron and Malik [Barron and Malik, 2015]. They rely on priors or cost functions imposed on reflectance, lighting and shape. Our triple product rendering approach can be beneficial to use since it will add more accurate material specific effects as well as self-occlusions, which lack in the original approach.

Section 6.2 shows how our renderer can be applied to an inverse rendering problem and how it decreases the execution time from hours to minutes and shows more accurate per pixel inverse rendering. Furthermore we show an improved factorization by including novel near-field lighting effects like near-field lighting. In Section 6.3, we insert our dynamic SRBF rendering in an existing intrinsic image decomposition framework.

## 6.1 Related Work

There are different kinds of techniques proposed in the research domain. Most of them vary on the number of unknowns. There is shape-from-shading [Horn, 1970, 1989; Zhang et al., 1999], that tries to extract geometry by making assumptions about shading and known illumination. There is structure-from-motion [Triggs et al., 2000; Hartley and Zisserman, 2003]

that also tries to extract geometry by ignoring shading and illumination. Furthermore there are techniques to extract shape by actively adding known and coded illumination in the form of structured light. An overview is given by Salvi et al. [Salvi et al., 2004]. Reflectance on the other hand, is often extracted in a controlled setup called a light-stage [Debevec et al., 2000] by densely sampling the BRDF. Other approaches work on a sparse set of images under known point light illumination [Marschner, 1998; Marschner et al., 1999; Lensch et al., 2003; Zickler et al., 2006] or more complex outdoor illumination [Yu and Malik, 1998; Debevec et al., 2004; Romeiro et al., 2008]. There are only a few techniques that try to extract shape, illumination and materials at the same time [Yu et al., 2006a,b; Haber et al., 2009; Barron and Malik, 2015]. There are a few challenges most of such inverse rendering techniques share.

The joint estimation of the shape, illumination and materials of a scene is interesting, because afterward the properties of the scene can be altered without another capture phase. This has important applications in for example the movie post-production. The problem is made difficult by the fact that multiple optical phenomena often give rise to the same pattern of pixel values in a picture. These ambiguities can arise during the estimation of geometry [Belhumeur et al., 1999], but reflectance and lighting are also inextricably linked [Pont and te Pas, 2006]. In the absence of additional information, this problem cannot be solved [Ramamoorthi and Hanrahan, 2001]. Therefore, other constraints are usually added in the form of multi-view information [Haber et al., 2009; Lee et al., 2012; Laffont et al., 2013; Duchêne et al., 2015].

A related technique to inverse rendering is the *intrinsic images* problem. Here, single images are decomposed into shape, albedo, reflectance and a basic representation of lighting. Recovering the intrinsic properties of a scene captured in an image is a long-standing and challenging problem [Land et al., 1971; Barrow, 1978]. The human perception of illumination, shading and lighting can differ from what computers perceive [Adelson and Pentland, 1996]. As in the original formulation, given an image of a scene, we try to retrieve a set of intrinsic images. These images are all registered with the original image, each describing one intrinsic characteristic. Different subfields in computer vision estimate different intrinsic characteristics: shape-from-shading techniques estimate normals and depth [Durou et al., 2008], specularity removal methods estimate image specularities [Artusi et al., 2011] and color constancy methods estimate the illuminant [Gijsenij et al., 2011]. State-of-the-art techniques in intrinsic images are able to reconstruct all factors quite well. For example Barron and Malik [Barron and Malik, 2011, 2012, 2015] try to tackle the texture-illumination problem with a probabilistic approach by defining priors for each of the factors. For example, they constrain the estimation of the albedo to a piecewise constant smooth function. Most of the priors use basic learning of multi-variate Gaussian distributions to learn characteristics of the factors. They achieve a relatively good separation of shape, albedo and reflectance. But the texture-illumination problem is far from being solved completely. This is partly due to the fact that they only support low-frequency lighting with a spherical harmonics representations

and no BRDF models. In addition to the static image estimation, some methods have been devised to work on videos [Ye et al., 2014; Imber et al., 2014; Meka et al., 2016].

## 6.2 Inverse Rendering Application

The main goal of an inverse rendering application is to extract all the physical properties and building blocks of a real scene, so that afterwards these properties can be reused to realistically reconstruct the scene in a different environment with accurate shadow, reflectance and lighting effects. A common approach in the literature is to use multi-view images to sample the scene and feed the samples to a basis optimization algorithm. An important step in the optimization algorithm is the evaluation of new estimates. This is done by evaluating the rendering integral for each sample in the input images and compare its result to the original pixel values. Current state-of-the-art algorithms are able to extract good estimates for the unknown factors, but they are far from solving the texture-illumination ambiguity. We have observed a few limitations that might have an impact on the quality of the inverse rendering algorithm. We believe that our improved triple product rendering algorithms using high-order wavelets or spherical radial basis functions can be put to use to overcome some of these limitations.

First, Haar wavelets will allow for an all-frequency representation of the data, but they lack smoothness in their reconstruction. Inserting coarse level Haar wavelet coefficients will introduce blockiness artifacts and will affect the optimization process (see Section 3.2.2). A smoother high-order wavelet or spherical radial basis function representation will overcome this problem. A second limitation is the execution time. We have observed that 95% of the time, the optimization process is used for evaluating the rendering equation with new estimates of the unknowns. Each estimate requires a preprocessing step and a full evaluation of the rendering equation for each input image. This step is often done on CPU and will scale linearly in execution time with the number of input images. This leads to multiple hours of optimization. If we are able to use our real-time SRBF framework (Chapter 4) where all three factors are fully dynamic, it will make the evaluation of a new estimate very efficient. It will bring down the execution time from a couple of hours to a couple of minutes. A last limitation we have observed is that the rendering does not allow for all lighting effects. Current techniques are limited to distant lighting to keep the number of unknowns scalable. We propose an extra factorization step that allows for near-field lighting effects without the need to estimate a full environment map per pixel.

To evaluate our three improvements, we have built upon an existing state-of-the-art inverse rendering technique [Haber et al., 2009]. In the next section we will give a background on the optimization process of such an inverse technique. Then, we detail further on how we developed our three improvements and how they influence the inverse rendering quality.

### 6.2.1 Background

The original approach, as authored by Haber et al. [Haber et al., 2009], treats the inverse rendering problem by minimizing a least-squares objection function  $O$  of a set of measured samples  $\{y_p\}$  in the set of input images  $\{I\}$  of the surface points  $x(y_p)$  and its rendered counterparts  $L_{x(y_p)}$ :

$$O = \sum_{p=1}^N (y_p - L_{x(y_p)})^2 \quad (6.1)$$

The rendering of the reflectance for each measurement is done using the triple product rendering approach of the previous chapters:

$$L_{x(y_p)} = \sum_i \sum_j \sum_k V_i \rho_j L_k C_{ijk} \quad (6.2)$$

with binding coefficients  $C_{ijk} = \int_{\Omega} \Psi_i(\omega) \Psi_j(\omega) \Psi_k(\omega) d\omega$ , where the visibility  $V$ , BRDF  $\rho$  and lighting  $L$  can be represented in any basis. Haber et al. [Haber et al., 2009] chose the Haar-wavelet basis since at the time of their work, it was the only basis allowing for high-frequency lighting effects with good compression performance.

To allow for an efficient optimization, the triple product integral is rewritten in a matrix notation by incorporating the tripling coefficients with the visibility factor ( $T_{v,km} = \sum_l C_{klm} V_l$ ). The rendering equation is now factorized to make the system linear for both  $\rho$  and  $L$ , resulting in a bilinear system of equations

$$L_{x(y_p)} = \rho_{x(y_p)}^T T_{x(y_p)} L \quad (6.3)$$

The bilinear system is solved using an iterative approach where we alternatively optimize for  $\rho$  and  $L$ . Each optimization step is now reduced to a convex Quadratic Programming problem which can be efficiently solved using the primal-dual interior point algorithm [Gertz and Wright, 2003]. It requires rephrasing of the full bilinear system in the quadratic form:

$$\min_x \|Mx - Y\|^2 \quad (6.4a)$$

$$\Leftrightarrow \min_x x^T Qx - 2x^T c + Y^T Y, \quad (6.4b)$$

where  $Mx$  can be seen as the evaluation of the render equation for pixel measurements  $Y$ ,  $Q = M^T M$  and  $c = M^T Y$ . This approach allows for posing any linear equality ( $Ax = b$ ) and inequality ( $Cx \geq d$ ) constraint onto the system. For example, when we optimize for the illumination  $L$ , we will solve for  $x$  with  $x \in \mathbb{R}^{\#coeffs \times 1}$  the unknown coefficients of the illumination  $L$ . The influences of the coefficients of  $L$  onto the rendered pixels are stored matrix  $M \in \mathbb{R}^{\#pixels \times \#coeffs}$ .

## 6.2.2 Hierarchical Refinement and High-order Wavelets

In general, the inverse rendering is an underconstrained and ill-posed problem [Marschner, 1998; Ramamoorthi and Hanrahan, 2001]. Only a sparse set of input images can be sampled to estimate a great number of wavelet coefficients of the environment map. A lot of techniques have been proposed to apply regularization on the underconstrained system. The idea of regularization is to guide the system in the most plausible direction. A common regularization technique is to limit the number of unknowns to estimate. In the case for the environment map estimation, it would be better if the number of estimates remains sparse and thus most coefficients can be set to zero. To achieve this, a hierarchical approach is often used [Peers and Dutré, 2003; Matusik et al., 2004; Peers and Dutré, 2005; Haber, 2015]. A hierarchical approach as implemented by Haber et al. [Haber, 2015] is showed in Algorithm 4. They exploit the tree structure of the wavelet basis and start from the top node of the wavelet tree. The idea is to start with a smaller wavelet tree and adaptively add more detail coefficients by splitting the leaf nodes. By only splitting at interesting nodes in the hierarchy, large portions of less important coefficients can be skipped and set to zero. This will add extra constraints to the system. The key of this algorithm is to split nodes of the wavelet tree that contribute to the solution of the system  $M$ . For each added coefficient, the rank is used to check if the system stays well-posed and thus contributes to the system.

---

**Algorithm 4** Hierarchical Refinement Scheme [Haber, 2015]

---

```

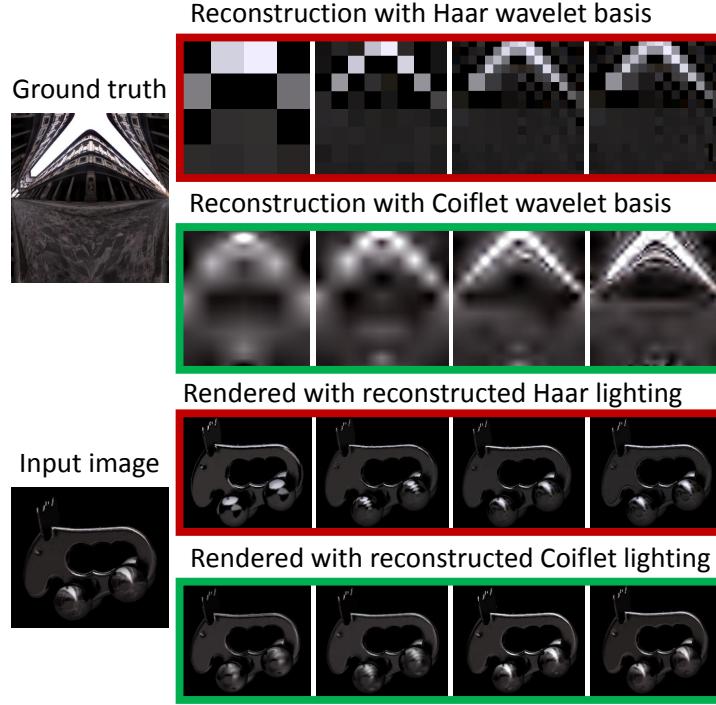
 $M$ : initialize system to solve at root node of wavelet tree
repeat
     $K$ : set of possible nodes for refinement
    for  $k \in K$  do
        concatenate  $k$  to  $M$ :  $M = M|M_k$ 
        if rank( $M$ )  $\neq$  full then remove  $k$  from  $K$ 
        end if
    end for
    Solve  $M$  for  $\tilde{L}$ 
    until no splits left

```

---

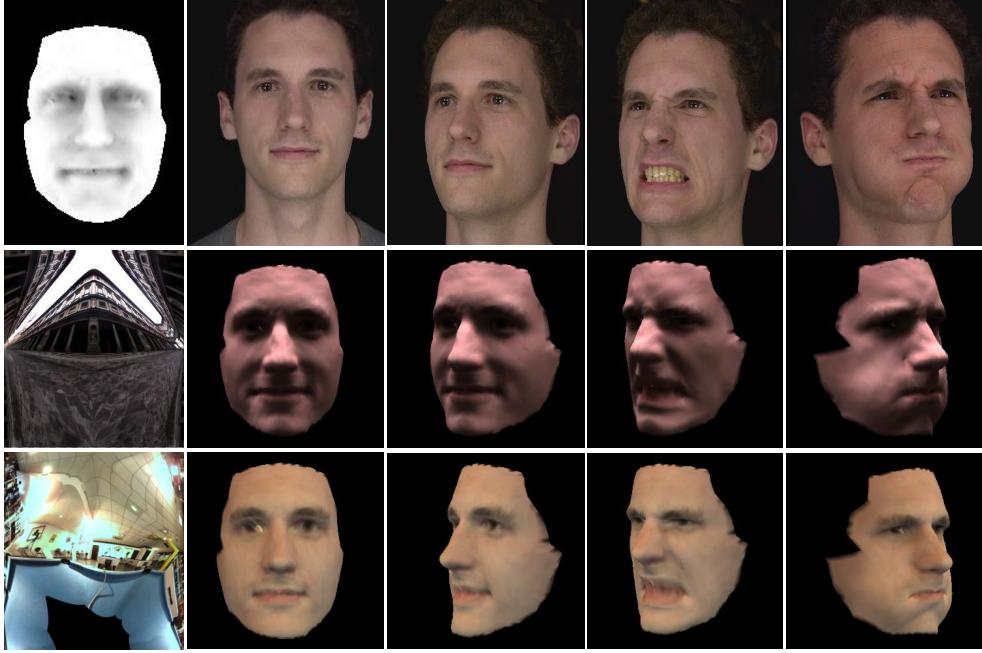
However, the hierarchical refinement scheme is not ideal when using Haar wavelets. Haar wavelet functions are by design piecewise constant and when we start from a smaller wavelet tree, only blocks of information can be inserted. In the lower levels, this will cause a lot of artifacts that need to be undone in the more detailed coefficients. This causes a slower convergence.

As detailed in Chapter 3, high-order wavelets are much smoother and are much better at representing smooth signals such as the environment map. The same is true for the smooth Gaussians used in our spherical radial basis functions rendering. Because of these reasons, it



**Figure 6.1:** Here, a hierarchical refinement scheme is used to estimate the lighting environment map. Step-by-step, the most interesting coefficients on a certain detail level are added to the sparse wavelet tree, based on a splitting criterium. Reconstructions for both Haar and the smoother Coiflet wavelet bases are shown. Haar has a tendency to introduce disturbing high frequencies around edges.

makes sense to use such smooth high-order wavelet or SRBF basis functions in a hierarchical refinement scheme. It will improve the convergence and will result in less noticeable artifacts. The same is true for spherical radial basis functions. They also have smooth characteristics in the coarser levels. Figure 6.1 gives a comparison of the hierarchical method for both a piecewise constant Haar basis and a smoother Coiflet basis. Reconstruction of a temporal dataset with this hierarchical method is shown in Figure 6.2.



**Figure 6.2:** Reconstruction of a temporal face dataset under different lighting conditions and estimated with the hierarchical refinement method. Ray traced occlusion maps, BRDF slices and lighting environment map are combined in the triple product integral calculation.

### 6.2.3 SRBF Triple Product in Inverse Rendering

Building the bilinear system of equations (Equation 6.3) requires the evaluation of the rendering integral for each sample in the multiview input images. More specifically, it requires the influence of each unknown onto each rendered pixel. The bilinear system is solved using an iterative approach that alternatively optimizes for material properties and lighting coefficients. When we first look at the environment map optimization, we solve the lighting coefficients  $L$  by minimizing the following system:

$$\min_L \|ML - Y\|^2 \quad (6.5)$$

where an initial guess of the material properties is incorporated in the influence matrix  $M$ . Each row of the influence matrix  $M$  consists of the joint term  $\rho_{x(y_p)}^T T_{x(y_p)}$ .

Normally, the coefficients  $L$  are represented in the Haar wavelet domain and the influence matrix  $M$  is also constructed using Haar wavelet triple product rendering. It works perfectly well, but it has one main drawback of being rather slow. Aside from an efficient logarithmic triple product traversal algorithm, it still requires a lot of preprocessing and has a slow

rotation operator (Section 4.1.3). During optimization of the environment map and material properties, the estimates change constantly, making the evaluation of the influence matrix a great bottleneck. It takes up until 95% of the execution time. Instead of inverse rendering with Haar wavelets, we wish to use our SRBF triple product theorem framework (Chapter 4), which is proven to be real-time and allows for dynamic changes in all three factors. Not only would the execution time decrease drastically, it would also allow for much more iterations in the optimization process. In addition real-time previews of the current estimates become possible. Such instant visual feedback can help the user to guide the optimization process to the correct minimum.

Since our dynamic SRBF triple product renderer is entirely evaluated on the GPU, we also prefer to build the influence matrix  $M$ , required for optimization, on the GPU. The matrix  $M \in \mathbb{R}^{\#pixels \times \#SRBFs}$  is generally too large to keep on the GPU. Furthermore, our evaluation is done per-pixel instead of the original per-vertex approach, making the matrix  $M$  even larger. Unfortunately, the solver does not require  $M$  directly, but uses  $Q = M^T M$  ( $Q \in \mathbb{R}^{\#SRBFs \times \#SRBFs}$ ), which contains much less data. Nevertheless, evaluations of  $M$  are necessary to calculate the enormous matrix multiplication of  $M^T M$ . An efficient algorithm to evaluate the values of  $Q$  is imperative, without the necessity of addressing the entire matrix of  $M$  at once. An important side note is that we want to avoid double evaluations of  $M$ .

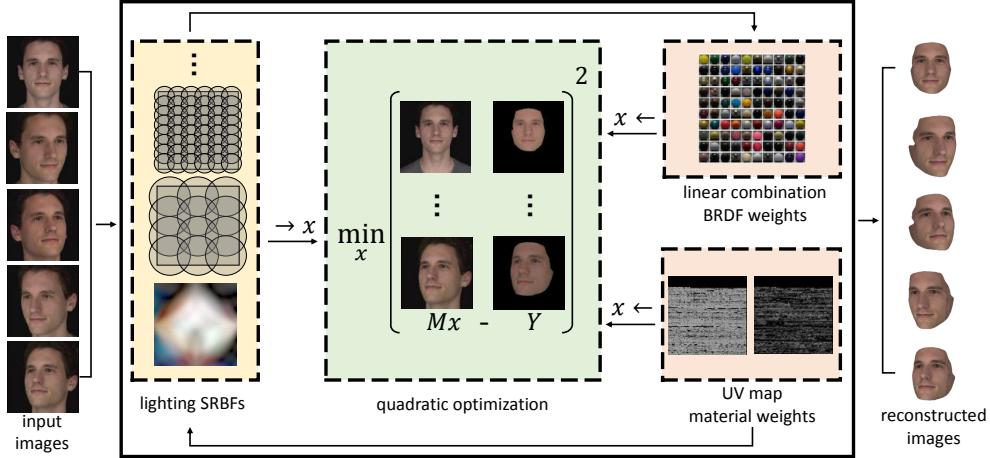
To efficiently build  $Q$  on GPU we divide the pixel measurements in  $N$  blocks and for each block we execute a two pass algorithm. The first pass stores the influences per SRBF for each pixel of the block in a buffer  $M_N$ , which is essentially a subset of  $M$  ( $M = M_1 \cup M_2 \cup \dots \cup M_N$ ). We have adapted our SRBF triple product rendering framework to calculate the influence of each individual SRBF on the rendered pixel and store the influences in a column of  $M_N$ . The second pass evaluates the quadratic  $M_N^T M_N$  and updates the  $Q$  matrix ( $Q = M_1^T M_1 + M_2^T M_2 + \dots + M_N^T M_N$ ). To implement a fast matrix multiplication, we store the influences of the block  $M_N$  in shared data to avoid redundant texture operations. The multiplication is then performed on the shared data.

The second part of the iteration is the optimization of BRDF and material properties, which is based on the linear BRDF factorization of Weistroffer et al. [Weistroffer et al., 2007]. The factorization states that each scene consists of a fixed number of materials, defined as a linear combination of basis BRDFs stored in a database. Then, each surface point of the scene is assigned a linear combination of such materials. In the original approach of Haber et al. [Haber et al., 2009], the linear combination of materials and BRDFs are estimated per vertex in the scene. As a consequence the quality of inverse rendering is greatly influenced by the amount of vertices available. Our SRBF renderer, on the other hand, is able to render scenes in a per pixel manner. To improve the quality of estimation, we wish to estimate the material properties more densely. To achieve this, we estimate in the texture space of the

object rather than in the geometry space. All triangles of the object are packed in a 2D texture map.

The texture map is constructed using a two-dimensional bin packing algorithm of Jylänki [Jylänki, 2010]. The triangles are sorted based on size, where the largest are inserted first and the remaining area of the texture map is filled by the smaller ones. Instead of optimizing material properties for each vertex in the scene, we can now estimate for each texel of the texture map. A texel refers to a unique 3D position in the scene, where we solve a bilinear system to estimate the linear combination of materials and BRDFs. Similar to the estimation of the lighting SRBF coefficients, the influences of the material weights onto the rendered pixels are calculated on GPU for each texel of the 2D texture map using an adapted SRBF triple product rendering shader. To construct the cost function, we also require the input images in the same texture map space. The projection of each input image onto the texture map is done in a separate shader and is executed as a preprocessing step. The resolution of the texture map can be dynamic and is mainly dependent on the resolution of the input images and the amount of texture resolution required. A main advantage of using a texture map as a representation for the materials is that they can be directly used in any off-the-shelf modeling or rendering framework.

An overview of the optimization process is given in Figure 6.3. A couple of input images (shown on the left) are passed onto the optimization framework. The optimization is executed in an iterative manner and alternates between solving for lighting SRBFs and material weights. The number of BRDFs and materials—a material is defined by a linear combination of BRDFs—is defined in advance. The initial guess for the material weights, which are represented in UV maps, is obtained by k-means clustering of the colors in the input images. Each cluster is assigned to a different material. The mean color of the material is assigned as a BRDF weight for the diffuse Lambertian component. The remaining BRDF weights are set to zero. Once the initial data is set, the optimization framework starts solving. Optimization is performed using a quadratic solver. The solver minimizes the  $L^2$ -norm of samples of the input images, denoted with  $Y$ , compared with a rendered equivalent  $Mx$ , where  $x$  is the unknown to estimate. The evaluation of the matrix  $M$ , which contains the influences of an unknown  $x$  on the rendered image  $Mx$ , is similar to forward rendering on GPU. First, the lighting factor is assigned as unknown  $x$ . Here, the lighting SRBFs for all levels of the hierarchy are estimated. In the next step, the estimated lighting SRBFs are used to solve for materials. This part is twofold. Firstly, the BRDF weights are kept as unknown. Secondly, we solve for the UV maps which represent the material weights. This process repeats itself until a maximum number of iterations is reached. Once the optimization step is performed, the estimated data can be used to render a reconstructed version of the scene (shown on the right).



**Figure 6.3: Overview of the inverse rendering framework using SRBFs.** A set of input images is passed onto a solver. The solver estimates lighting SRBFs, BRDF weights and material weights in the form of UV maps. Quadratic optimization is used to minimize the  $L^2$ -norm of the input samples  $Y$  compared to rendered samples  $Mx$ . Once the maximum number of iterations is reached, the estimates can be used to reconstruct the input images or to apply different lighting conditions.

#### 6.2.4 Near-Field Lighting

Up until now, we have focused on the underlying basis representation to improve the quality and time performance of rendering and inverse rendering applications. However, nothing was changed regarding the formulation of the factors itself and how they are used within the current evaluation of the triple product rendering integral. The factorization is based on precomputed radiance transfer, where the shape data is represented as a visibility map, the materials as a BRDF slice and the lighting as an environment map. One important drawback of this factorization is that the environment lighting is limited to distant lighting. Each light source in the scene is assumed to be at infinity, because each surface point in the scene uses a constant environment map lighting during triple product rendering. This is no longer true when the lights source itself is part of the scene. The irradiance of each surface point will differ based on the relative position of the light source with respect to the surface point.

A straightforward solution to include near-field lighting effects would be to let every surface position of the scene have a different lighting map. In the case of forward rendering, this will still be feasible since one can ray-trace a lighting map for each surface point and compress it using wavelets or spherical radial basis functions as a preprocessing step. During rendering, each surface point can access its own preprocessed lighting map, allowing for near-field lighting effects. This technique is memory bound and is limited by the number of surface points to render. Nevertheless, this is no longer applicable to an inverse rendering algorithm.

Most inverse rendering systems in general are already ill-posed and underconstrained when estimating the unknown coefficients of only one constant environment map for the entire scene. The number of unknowns and coefficients to be estimated will grow exponentially if we allow for per-vertex lighting maps, making the system heavily underconstrained. We can conclude that, at this point, near-field lighting effects are impossible to estimate. But how important are these near-field lighting effects? Why can't we just ignore near-field lighting effects in an inverse rendering technique?

We have investigated the effect of such near-field lighting effects in collaboration with Put et al. [Put et al., 2015], whose main goal is to improve the separation of material estimation and lighting estimation to solve for the texture-illumination ambiguity. In general, when the object is relatively small compared to the outside environment, one environment map for the entire scene will suffice. For example, for an outside environment where the sun is the main light source, we can assume that the light is approximately at infinity. However, when the light sources are considerably closer the object, near-field lighting effects become much more apparent, especially for colored light sources. The closer a light source to an object, the more lighting parallax it will be subjected to. This lighting parallax effect results in regions on the object with less incident light and regions with more incident light. The colors of the light source will be much more noticeable on certain regions of the object. Furthermore, it will have a subtle influence on how the shadows are cast. Current state-of-the-art inverse rendering techniques do not take this light parallax into account and will falsely spread the aforementioned near-field lighting effects in the estimates of material coefficients or distant environment map coefficients.

As mentioned before, the system will become too underconstrained and unstable if we estimate coefficients of the lighting maps per surface point. We need to find a solution to include a near-field lighting term in the triple product rendering integral without drastically increasing the time complexity. In addition, we should be able to estimate the coefficients for near-field lighting and at the same time keep the system well-posed by avoiding too many extra unknowns. We propose a new factorization by splitting the illumination in a distant and a near-field lighting factor. The distant illumination factor is similar to the original approach and is estimated directly for its compressed wavelet or SRBF coefficients. We can assume that all lighting of a scene is additive, so we can handle the near-field lighting factor as a separate rendering step. A second assumption we make is that the local emitter is visible in the scene. The next step is to cluster the scene in possible emitters. We cannot identify each triangle in the scene as a possible emitter. Not only would the rendering performance drop drastically, but the number of unknowns will be too large as well, making the system too underconstrained. It would be better to use the high dynamic range input images to cluster triangles based on the radiance values of the pixels in the image. We identify clusters of pixels in the input image with similar radiance, using connected components. The calibration information is used to extract the underlying triangles for each cluster.

So far, we identified different clusters of emitters, but how can we apply their influence on the rendered pixels? We do this by evaluating the irradiance of each emitter on each surface point of the scene. To do this, we build irradiance maps per surface point for each cluster. More specifically, if the scene is clustered in  $N$  emitters, then we will have  $N$  irradiance maps for each surface point. Each irradiance map is defined around the hemisphere of a surface point and contains the hemispherical influence of a certain emitter on the surface point. The irradiance maps are compactly stored in a wavelet or SRBF basis and are built with raytracing as a preprocessing step taking the visibility, attenuation and the color of the emitter into account. Once all irradiance maps are preprocessed, they allow for efficient evaluation during rendering. Each emitter is assigned a power that adjusts the intensity of the emitter. These globally assigned powers will regulate the influence of an irradiance map on a surface point. Rendering with the near-field lighting is now the same as adding an extra double product integral of the BRDF and the irradiance maps to triple product rendering equation Equation 2.6:

$$B(v, \omega_0) = \underbrace{\sum_k \sum_l \sum_m C_{klm} \rho_k V_l \tilde{L}_m}_{\text{distant lighting}} + \underbrace{\sum_c \sum_k \rho_k \cdot (I_c P_c)}_{\text{near-field lighting}} \quad (6.6)$$

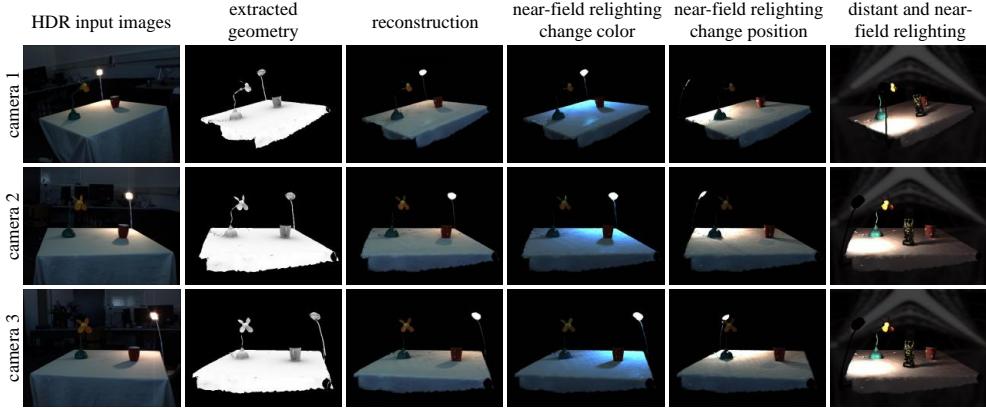
For every cluster  $c$ , visible over the hemisphere of vertex  $v$ , its irradiance map  $I_c$  is multiplied by the estimated power  $P_c$ . Finally, it is multiplied by the BRDF  $\rho$  at  $v$ . When using an orthogonal wavelet basis, the double product integral term reduces to a simple dot product calculation [Ng et al., 2004].

Instead of keeping the coefficients of the irradiance maps as unknowns during optimization, we solve for the emitting powers of the clusters. This is very advantageous because the number of extra unknowns remains rather small. We can update the system of bilinear equations (Equation 6.3) used for optimization and add the new near-field lighting factor:

$$L_{x(y_p)} = \underbrace{\rho_{x(y_p)}^T T_{x(y_p)} L}_{\text{distant lighting}} + \underbrace{\sum_c \rho_{x(y_p)} I_{v,c}(y_p) P_c}_{\text{near-field lighting}} \quad (6.7)$$

We wish to avoid minimizing to a local minimum by alternating between distant and near-field lighting optimization. We achieve this by appending all the influences of the distant lighting and near-field lighting in one big influence matrix  $M \in \mathbb{R}^{\#\text{pixels} \times (\#\text{coeffs} + \#\text{clusters})}$ . The unknown coefficients for distant lighting and the powers for near-field lighting can be estimated at once with  $x \in \mathbb{R}^{(\#\text{coeffs} + \#\text{clusters}) \times 1}$ . This optimization step for illumination can be substituted in the full process of inverse rendering. Now the inverse rendering is alternated between material estimation and distant/near-field lighting estimation.

A result of inverse rendering with near-field lighting is shown in Figure 6.4. Here we captured a table dataset with 8 synchronized and calibrated cameras. The geometry is reconstructed



**Figure 6.4:** Inverse rendering using near-field light sources. A synchronized set of 8 photographs are used together with an active reconstruction of the scene as an input for the inverse rendering algorithm. The dataset clearly contains a near-field light source in the form of a desk lamp. Our adapted algorithm is able to take into account the near-field light source, making it possible to also apply relighting on the desk lamp. For example, the power, color or position of the desk lamp can be altered. In addition, new objects can be inserted with accurate lighting and shadows caused by the extracted near-field lighting conditions of the desk lamp.

using a structured light scanning approach based on gray codes. The main emitters are identified in the high dynamic range input images. In this case, there is only one visible emitter, i.e. the desk lamp. The irradiance maps are preprocessed for this emitter and the optimization process is executed. The results show that we can make a decent reconstruction of the original scene using the extracted parameters for geometry, materials, distant lighting and near-field lighting. We can apply near-field relighting by repositioning the desk lamp or change the color or power of the desk lamp. Notice how the shadows and radiance on other objects change accordingly.

Overall, adding near-field lighting in inverse rendering techniques shows promising results and adds an extra level of interactivity. However, we are still experiencing some minor problems. In some cases, there is still a residue of some old shadows. This is due to the fact that not all lighting effects are fully simulating the real life conditions. It is almost impossible to exactly simulate the effects of shadows. Furthermore, a bad reconstruction of geometry, noise and the texture-illumination ambiguity will amplify this problem. Nevertheless, apart from some artifacts and residues, we are able to remove almost all original lighting effects.

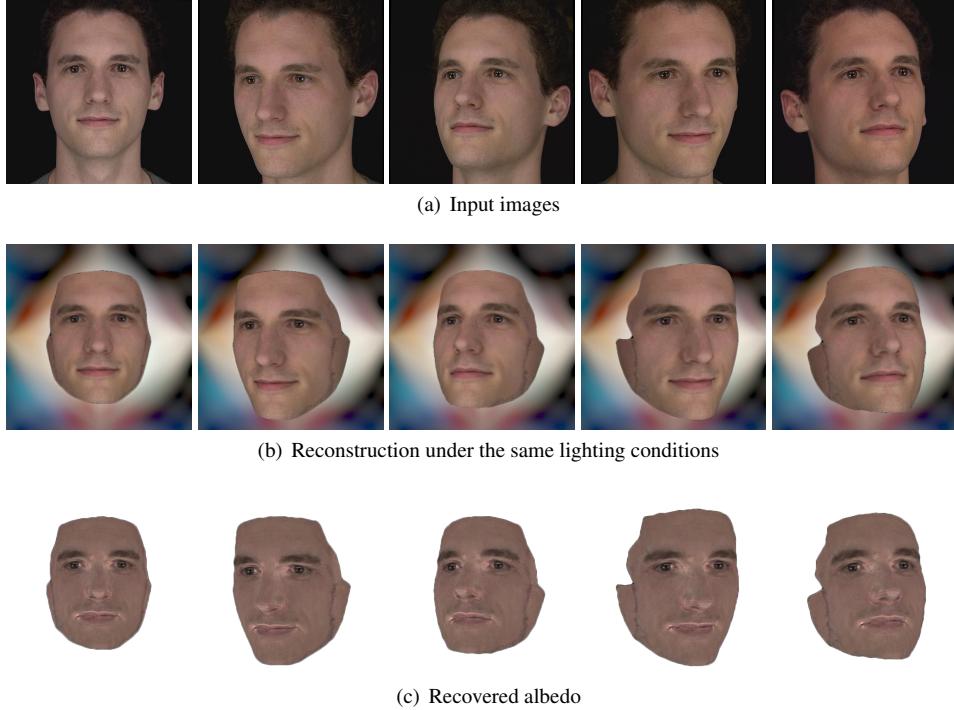
### 6.2.5 Results

The inverse rendering algorithm is integrated in our SRBF triple product rendering framework and is tested on an Intel Core™ i7 2.67 GHz processor with 12 GB of RAM and a GeForce 690 GTX.

We applied the inverse rendering algorithm on the SurreyFace dataset (courtesy of Klaudiny et al. [Klaudiny, 2013]), which consists of images of a frontal face captured with 5 synchronized cameras. The input images are downsampled to a resolution of  $960 \times 540$ . The enclosed mesh consists of 2689 vertices and 5248 faces. The results of the inverse rendering algorithm is shown in Figure 6.5. The input images of Figure 6.5(a) are used to build the bilinear system necessary for the inverse rendering optimizer (Section 6.2.3). The bilinear system is solved iteratively for both the lighting conditions in the form of an environment map and the material weights for each texel of the scene’s texture map. After optimization, the extracted scene properties can be used to reconstruct the scene, using the forward renderer. This is shown in Figure 6.5(b). The reconstruction is performed in real-time and can serve as an intermediate preview of the algorithm’s progress during optimization. The recovered material or albedo information is shown in Figure 6.5(c). Overall, it can be seen that most lighting effects are gone in the albedo renderings. However, some small lighting residue still remains noticeable, especially around the borders of facial features. This can be attributed to the inaccurate normals in these regions, as well as the fact that these regions have high-frequent transitions of lighting in the original image, which can result in an underconstrained optimization problem.

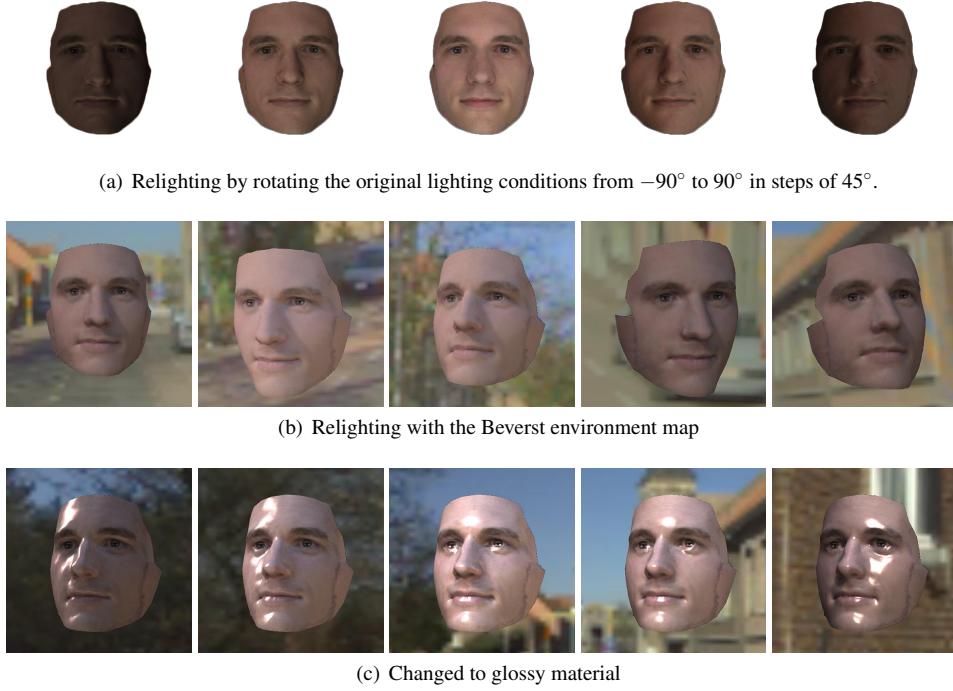
Relighting of the SurreyFace dataset is similar to the reconstruction approach, but instead of using the exact extracted lighting conditions, a rotated version or an entirely different environment map is used for rendering. Figure 6.6(a) shows different renderings of the recovered dataset where we rotate the estimated lighting conditions between an angle of  $-90^\circ$  to  $90^\circ$  in steps of  $45^\circ$ . One can observe that the main diffuse light source is rotated and the shadows are rotated from the right to the left accordingly. Figure 6.6(b), on the other hand, uses the Beverst environment map. The outside environment of Beverst contains a clear blue sky with bright sunlight. The face is now rendered with a blue outlook and the sun is causing self-occlusions onto the face. The shadows that are cast by the sun are especially noticeable in the regions of the nose. In addition to the change of lighting conditions, we can also change the material properties. For example, the materials can be altered to have a more plastic-like outlook. This is illustrated in Figure 6.6(c) where the faces are rendered with a mix of more specular BRDFs. The highlights of the sun are now clearly visible.

A comparison between inverse rendering with Haar wavelets and spherical radial basis functions for the SurreyFace dataset is tabulated in Table 6.1. Only half the preprocessing time is required when using SRBFs. The Haar wavelet approach requires the loading of a giant preprocessed rotation matrix. A second important bottleneck is the preprocessing of occlusion maps for the visibility function. For each vertex, a  $32 \times 32$  occlusion map is raytraced in



**Figure 6.5: Inverse rendering of the SurreyFace (courtesy of Klaudiny et al. [Klaudiny, 2013]) dataset using spherical radial basis functions instead of Haar wavelets. (a) Five input images are used to reconstruct the environment lighting as well as the material properties. (b) Once the lighting and material conditions are extracted, they can be used to reconstruct the scene under the same input conditions. (c) The recovered materials or albedo of the scene.**

the pixel domain and then converted to the Haar wavelet basis. For this rather small mesh, it takes about 15 seconds for raytracing all the occlusion maps. However, the amount of preprocessing grows linearly with the number of vertices. For example, a mesh with 100k vertices will take at least 10 minutes of preprocessing. SRBFs on the other hand are able to update the visibility in real-time and thus requires no preprocessing. The most time (14.5 s) is spent in the bin packing algorithm, necessary for sampling the image data in the texture domain of the scene. The projection of the images onto the pixel domain is done in real-time using GLSL shaders. For the optimization step, we show the execution time for different parts of one iteration. Optimization with Haar wavelets takes the most time when solving for the lighting coefficients. For each of the 1024 Haar wavelet coefficients, the optimizer requires all the influences for each pixel of the input images. These influences are extracted using an adapted forward rendering approach. Unfortunately, rendering with Haar wavelets is far from real-time, because often BRDF slices need to be sampled on-the-fly in the pixel domain before



**Figure 6.6: Relighting of the SurreyFace dataset using spherical radial basis functions instead of Haar wavelets. Relighting is applied by changing the lighting conditions of the reconstructed scene. (a) Rotation of the extracted lighting environment map. (b) Relighting using the Beverst outdoor environment map. The sun is now casting shadows onto the face. (c) Changing the material of the face to a more glossy BRDF under the Beverst lighting environment.**

they are converted to Haar wavelets. Furthermore, the rotation of wavelets is expensive. As a net result, the lighting coefficients are extracted in 1089 seconds. When using SRBFs, the forward rendering and building the influence matrix is entirely done on GPU and in real-time. The amount of time necessary to extract the 1020 lighting SRBFs is reduced to 10 seconds. One can observe that the total time for estimating the BRDF weights in the SRBF domain takes 180.8 seconds, which is considerably more compared to the 2.5 seconds in the wavelet domain. However, this is due to the fact that the SRBF approach applies the optimization per texel and not per vertex. In our approach we estimate BRDF weights for 906240 texels, where the wavelet approach only estimates for 2689 vertices. A more precise measure is the estimation of one single BRDF weight. We can observe that for one BRDF weight our approach takes only slightly more time with  $1.99 \times 10^{-4}$ s in contrast to  $1.36 \times 10^{-4}$ s when using the original Haar wavelet approach. Because our approach estimates the BRDF weights more densely, the result of the reconstruction will retain much more detail and will greatly

**Table 6.1:** Comparison of Haar wavelets [Haber et al., 2009] versus SRBFs (ours) in inverse rendering.

	Haar wavelets	SRBFs (ours)
General	input resolution	$960 \times 540$
	# input images	5
	# vertices	2689
	texture map resolution	n/a
Preprocess	$\Delta t$ update visibility	15.170 s
	$\Delta t$ update lighting	0.006 s
	$\Delta t$ update BRDF	<0.001 s
	$\Delta t$ loading rotation matrix	24.940 s
	$\Delta t$ overlapping calculation	n/a
	$\Delta t$ sampling image data	3.085 s (per vertex)
	$\Delta t$ clustering materials	0.004 s (per vertex)
	$\Delta t$ total preprocessing	48.875 s
		<0.001 s
Optimization	# estimated materials	3
	# estimated BRDF weights	2689 (per vertex)
	# estimated lighting coefficients	1024
	$\Delta t$ optimize lighting	1089.340 s
	$\Delta t$ optimize materials	2.516 s
	$\Delta t$ optimize BRDF weights	2.518 s
	$\Delta t$ optimize one BRDF weight	$1.36 \times 10^{-4}$ s
	$\Delta t$ one iteration	1099.680 s
	$\Delta t$ full optimization	184 min
Reconstruction		
		
	$\Delta t$ initialize reconstruction	30.426 s
	$\Delta t$ rendering reconstruction	13.403 s
		9.957 s
		0.022 s

improve the visual quality of the inverse rendering algorithm. This is clearly demonstrated in the rendered reconstruction. In total, a speedup factor of approximately 6 is achieved for this dataset. Lastly, the rendering of the reconstruction with our approach can be done at 45 fps, where the reconstruction using Haar wavelets is far from real-time with 0.07 fps.

### 6.3 Intrinsic Image Decomposition Application

An alternative approach to untangle the effects of joint shape, lighting and reflectance effects is to exploit the statistical characteristics of the three factors of the rendering equation. Some effects are more likely to occur than other effects. Such a probabilistic approach tries to find the most likely explanation for an input image. For example, reflectance tends to be piecewise constant, lighting tends to be natural and shape tends to be smooth. Each of these characteristics can be formulated as a prior that describes the likelihood for a certain effect to take place. Exploiting image statistics by defining priors has already been used in applications for natural image statistics [Field, 1987; Ruderman and Bialek, 1994]. In the case of intrinsic image decomposition, Barron and Malik [Barron and Malik, 2015] defined multiple priors for shape, illumination and reflectance.

In this section we will explain how our triple product renderer with spherical radial basis functions can be used in intrinsic image decomposition applications. We implemented the optimization framework of Barron and Malik [Barron and Malik, 2015] and show how SRBFs can be used to construct the different priors as well as how to apply them in the rendering pass.

#### 6.3.1 Background

Barron and Malik [Barron and Malik, 2015] tackle intrinsic image decomposition as an optimization problem where different priors are used as a cost function. The different priors for shape, illumination and reflectance are combined into one optimization problem, formulated as follows:

$$\begin{aligned} & \underset{G,R,L}{\text{maximize}} \quad P(G)P(R)P(L) \\ & \text{subject to} \quad I = R + S(G, L) \end{aligned} \tag{6.8}$$

where  $G$ ,  $R$  and  $L$  are the estimated intrinsic characteristics for respectively depth, reflectance and illumination. An image  $I$  is formed by adding together log-space reflectance  $R$  and log-space shading  $S(G, L)$ , where shading is a function of the estimated geometry and illumination. Shading can be seen as the actual forward rendering of the scene. No BRDF function is used in the original approach.  $P(G)$ ,  $P(R)$  and  $P(L)$  are the priors defined on  $G$ ,  $R$  and  $L$ . They define the likelihood of the observed intrinsics that is used to render the image. The goal is to maximize the combination of these priors.

The reflectance prior  $P(R)$  consists of three subpriors. First, the characteristic of piecewise constancy is integrated. The second assumption is parsimony of reflectance that states that most objects only consist of a limited number of colors. Third and last, there is an absolute prior that defines how likely a color belongs to the reflectance of a scene. The total reflectance cost is the sum of the three aforementioned cost functions. Each cost is assigned a weighting factor, which are obtained using cross-validation on an example dataset. The shape prior  $P(G)$  is threefold. First, there is the assumption of smoothness, where the variation of mean curvature should be minimized. Next, an assumption on isotropy stating that shapes are just as likely to be oriented in one direction as they are in another. Last, there is the boundary constraint, which states that shapes tend to face outwards at the occluding contour. Similar to the reflectance prior, the three costs are weighted where the weights are learned using cross-validation. The illumination prior  $P(L)$  is straightforward. The prior defines what kind of environment maps are likely to occur. Different natural environment maps are used to train a multivariate Gaussian, which is used as a cost function for the illumination estimation.

An optimizer wishes to find an optimal solution for the three intrinsics. For this, the problem is simplified to a minimization problem by rephrasing the priors to a cost function by taking the negative log likelihood of the priors  $P(G)P(R)P(L)$ . The cost function, alongside with the calculated derivatives, is passed to the L-BFGS optimization algorithm. After optimization, the least costly and thus the most likely solution for G, R and L is obtained.

### 6.3.2 SRBF Triple Product Gradients Rendering

Most solvers and in this case the L-BGFS solver require the derivatives of the function to minimize. These derivatives are calculated for all the different priors and are then propagated to the gradients of the proper unknowns. Instead of using spherical harmonics, as proposed in the original approach, we use spherical radial basis functions as a representation in the rendering framework. Spherical harmonics functions have a quite straightforward gradient calculation, which is just a simple linear operation. If we want to use spherical radial basis functions as an underlying basis function, the gradient calculation of the priors gets more complex. We need to be able to find correct derivatives of the full triple product integral. Using a finite differentiation method is far from accurate. On the other hand, automatic differentiation is too time consuming, since it requires multiple evaluations of the rendering integral, which we should avoid. An analytical formula for extracting gradients is the best solution for an efficient implementation. We used symbolic differentiation to extract the analytical formulas for the partial derivatives  $\frac{\partial B(x)}{\partial \text{unknown}}$  of the triple product rendering equation  $B$  with respect to the unknowns. For each SRBF we use symbolic differentiation to calculate

partial derivatives for the coefficients  $\mu_L$  and the SRBF centers  $\vec{p}_\rho$ :

$$\frac{\partial B}{\partial \mu_L} = \frac{\mu_\rho \mu_V e^{-(\lambda_\rho + \lambda_V + \lambda_L) + \|\lambda_\rho \vec{p}_\rho + \lambda_V \vec{p}_V + \lambda_L \vec{p}_L\|}}{\|\lambda_\rho \vec{p}_\rho + \lambda_V \vec{p}_V + \lambda_L \vec{p}_L\|} \quad (6.9)$$

$$\frac{\partial B}{\partial \vec{p}_\rho} = \frac{e^{-(\lambda_\rho + \lambda_V + \lambda_L) + \|r\|} \left( \frac{t}{2\|r\|} - 1 \right)}{2\|r\|} - \frac{e^{-(\lambda_\rho + \lambda_V + \lambda_L) + \|r\|} t}{\|r\|^3} \quad (6.10)$$

$$\text{with } t = 2\vec{p}_{\rho,x}(\lambda_\rho \vec{p}_\rho) + 2\vec{p}_{\rho,y}(\lambda_V \vec{p}_V) + 2\vec{p}_{\rho,z}(\lambda_L \vec{p}_L)$$

$$\begin{bmatrix} \frac{\partial B}{\partial \vec{p}_{\rho,x}} \\ \frac{\partial B}{\partial \vec{p}_{\rho,y}} \\ \frac{\partial B}{\partial \vec{p}_{\rho,z}} \end{bmatrix} = \begin{bmatrix} g(\lambda_\rho \vec{p}_{\rho,x} + \lambda_V * \vec{p}_{V,x} + \lambda_L \vec{p}_{L,x}) \lambda_\rho \\ g(\lambda_\rho \vec{p}_{\rho,y} + \lambda_V * \vec{p}_{V,y} + \lambda_L \vec{p}_{L,y}) \lambda_\rho \\ g(\lambda_\rho \vec{p}_{\rho,z} + \lambda_V * \vec{p}_{V,z} + \lambda_L \vec{p}_{L,z}) \lambda_\rho \end{bmatrix} \quad (6.11)$$

$$\text{with } g = \frac{e^{-(\lambda_\rho + \lambda_V + \lambda_L) + \|r\|}}{4\|r\|^2} - \frac{e^{-(\lambda_\rho + \lambda_V + \lambda_L) + \|r\|}}{2\|r\|^3}$$

These calculations are done entirely on GPU and are integrated with the forward rendering phase of the shading function. For each pixel, the required gradients are evaluated and stored in a texture buffer. Once the rendering step is executed, the gradients—together with the rendered shading—are used in the evaluation of the prior functions where the gradients of the different priors are being propagated to the gradients of the unknowns.

We will first take a closer look at the illumination prior. Here a multivariate Gaussian distribution is fitted on a training set of environment maps. The cost function is defined as the Mahalanobis distance of an estimate compared to the learned Gaussian distribution. The gradient of the cost function is directly calculated per SRBF coefficient and thus no gradient propagation is required.

This is not the case for the reflectance prior. The reflectance function itself is not an unknown in the optimization process. The gradients calculated for the cost function of the reflectance are in the space of the reflectance image and state the influence of a pixel in the reflectance image on the cost function:  $\frac{\partial \text{priorR}}{\partial R(x)}$ . However, the reflectance has an indirect connection with the unknowns for lighting and depth. The reflectance map is extracted by subtracting the shading image from the input image and the shading image itself is constructed using estimates for lighting and depth. If we want to solve the problem for the lighting coefficients and depth, we need to know the influences of the lighting and depth estimates on the total cost function of reflectance. The first step is to propagate the influence of a reflectance pixel to the influence of a shading pixel. This is fairly simple and is done by negating the gradients:  $\frac{\partial \text{priorR}}{\partial S(x)} = -\frac{\partial \text{priorR}}{\partial R(x)}$ . Now we can use the gradients we calculated during rendering which contain the influence of a lighting coefficient on a pixel to propagate the per pixel gradients to per lighting gradients. This is done by applying the product rule for derivatives:  $\frac{\partial \text{priorR}}{\partial L} = \frac{\partial \text{priorR}}{\partial S(x)} * \frac{\partial S(x)}{\partial L}$ .

### 6.3.3 Results

We have developed a full implementation of the intrinsic image decomposition problem where we imposed all priors on shape, reflectance and illumination. We integrated it with our interactive SRBF triple product rendering framework. The algorithm works on single input images only. We applied our adapted algorithm on the “Paper” dataset of Barron and Malik [Barron and Malik, 2015], as well as on a new “Shorts” dataset and the SurreyFace dataset [Klaudiny, 2013]. The results are shown in Figure 6.7. Results for extracted normals, albedo, shading and lighting conditions are given.

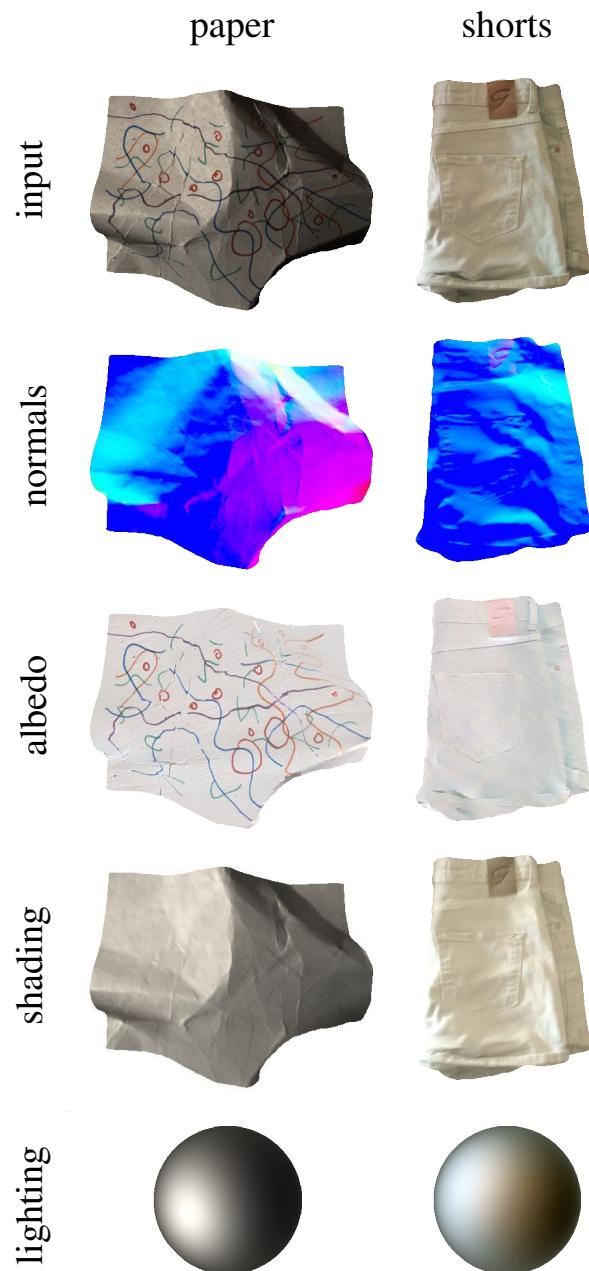
When the intrinsic decomposition is finished, we can use the estimates to reconstruct the scene or we can for example change the lighting conditions. Relighting of the “Paper” dataset is shown in Figure 6.8 and relighting of the “Shorts” dataset is shown in Figure 6.9. For both results we show rendering with different environment maps. We respectively show the Beverst, Bowling and Kitchen environment map. In addition, we can interactively change the material properties of the object. The original approach was limited to diffuse shading. The use of our SRBF rendering approach also allows for a more glossy or specular outlook by changing the BRDF parameters of the model.

## 6.4 Conclusions

In this chapter we showed how improved representations for the simulation of light transport affect current state-of-the-art appearance extraction algorithms. We have demonstrated the use of high-order wavelets and SRBFs in both an inverse rendering and intrinsic image decomposition problem.

Our SRBF based representation mainly has an impact on the quality and time performance of the appearance extraction algorithm. We explained how an efficient SRBF triple product rendering framework will nullify the main bottleneck of building the bilinear equation for the solver, which is essentially a forward rendering operation. Because our SRBF triple product rendering framework is able to dynamically change shape, illumination and materials and render at interactive framerates, the execution time of an inverse rendering algorithm can be reduced from hours to minutes. Furthermore, the system of bilinear equations is entirely built on GPU. Our algorithm is fully adapted to work in texture space instead of vertex space, allowing for more detailed albedo reconstructions. Overall, the relighting of a dataset will result in much more detailed renderings.

Unfortunately some of the inherent problems of the state-of-art inverse rendering algorithms are still present. For example, it is still possible that lighting conditions are wrongly assigned to the BRDF term, or the other way around, that colors of the materials are simulated with coefficients in the environment map. This is more an algorithmic problem instead of a representation problem. However, we demonstrated that inverse rendering algorithms can benefit from a smoother underlying basis representation like high-order wavelets or SRBFs. A smooth representation will allow for a better hierarchical refinement and thus a faster convergence to the correct estimates. In addition, we proposed a better factorization of the light transport by making it possible to include near-field lighting effects in the rendering phase. It is clear that a better simulation of the light conditions that better match with the real world will result in a better separation of lighting and material properties. Intrinsic image decomposition algorithms are able to cope with the lighting and texture ambiguity by applying extra priors on the data. We explained how an SRBF representation can be used for a more accurate simulation of light in a prior based optimization method.



**Figure 6.7: Intrinsic image decomposition on different input images using our SRBF render framework. The input images of the top row are decomposed into normals, albedo, shading and environment lighting.**



**Figure 6.8: Relighting on the “Paper” dataset.** Three different environment maps are applied: Beverst, Bowling and Kitchen. These are shown in the top row. During rendering, the user can change the material properties to diffuse (second row), glossy (third row) or a mix of BRDFs (bottom row).



**Figure 6.9:** Relighting on the “Shorts” dataset. Three different environment maps are applied: Beverst, Bowling and Kitchen. These are shown in the top row. During rendering, the user can change the material properties to diffuse (second row), glossy (third row) or a mix of BRDFs (bottom row).

# Chapter 7

---

## Conclusions and Future Work

7.1	General Wavelet Triple Product Rendering .....	130
7.2	Spherical Radial Basis Triple Product Rendering .....	131
7.3	Relighting Use Cases .....	132
7.4	Future Work .....	133

This dissertation has presented new methods for efficient triple product rendering with different underlying basis representations. We have learned that an improvement of the representation in the rendering equation, which can be a small step in the research in computer graphics, can have a great influence in the development of image processing or computer vision algorithms. The way shape, materials and lighting data are represented and used to simulate the light propagation with precomputed radiance transfer has a great influence on the quality and efficiency of relighting applications. We have argued that an improvement of these basis representations are essential to bridge the gap between the virtual and the real world and to achieve more interactive and realistic renderings of virtual objects in the real world, as well as realistic renderings of real objects under different lighting conditions.

## 7.1 General Wavelet Triple Product Rendering

We devised an efficient algorithm to calculate the product integral binding coefficients for a heterogeneous mix of wavelet bases. This allows us to use any wavelet basis function in the triple product rendering integral and we are no longer limited to Haar wavelets only. The binding coefficients can no longer be manually identified compared to the state-of-the-art Haar wavelet algorithms, because of the complex overlapping characteristics of high-order wavelets. In this dissertation, we have developed different algorithms to efficiently evaluate the non-zero elements in the tensor of binding coefficients. Although, the tensor is denser compared to its Haar wavelet counterpart, it still remains very sparse. We exploited the characteristics of the high-order wavelets in order to identify other overlapping wavelet functions. By using the support, symmetry and vanishing moments of wavelets, we have drastically reduced the complexity of calculating the tensor of binding coefficients. For example, each wavelet has a fixed support and only a small number of other wavelet functions will have an overlap with that specific wavelet. Furthermore, we have observed that the branch of binding coefficients of wavelets overlapping a certain other wavelet is similar to all the other wavelets of the same level. We preprocessed the entire branch of binding coefficients and duplicated them to the other branches. Last, when the mix of wavelet bases in the product integral are homogeneous, symmetry can be observed. This halves the number of calculations, because the values of the tensor can be mirrored. We mainly focused on double and triple coefficients, but the approach can be extended to quadruple or higher product integrals. We created a robust solution method that is not only useful for triple product rendering applications, but can be deployed to any other signal processing algorithm that requires product integrals in combination with high-order wavelets.

A second conclusion is that the underlying representation should be tailored to the signal. It is optimal to use the characteristics of visibility, materials and lighting to assign a proper wavelet basis. Haar wavelets are optimal to represent the piecewise constant visibility functions, where the BRDF and lighting functions are generally more smooth and are better ap-

proximated using high-order wavelet functions, for example the Daubechies wavelet. By assigning an optimal wavelet basis to the functions, we reduced the number of coefficients by a factor of up to 20%. We have demonstrated this in a rendering application where coarser Haar wavelets introduce much more blocky artifacts in the lower levels of the environment map compared to their smoother high-order counterparts.

Although we have developed an efficient calculation for the sparse tensor, we sacrificed some performance in the triple product calculation. Ng et al. [Ng et al., 2004] is able to evaluate a triple product integral in linear time because they can iterate over all non-zero wavelet coefficients and evaluate only the exact cases where the binding coefficients result in zero. On the other hand, in our more general approach we can either iterate over all non-zero binding coefficients of the tensor or all non-zero wavelet coefficients of the signals, but no longer both at the same time. However, high-order wavelets have much better compression performance. The number of coefficients to represent each signal will be much lower. We argue that the loss in time performance for triple product rendering is earned back because we need to evaluate fewer triple product evaluations, and thus fewer binding coefficients.

## 7.2 Spherical Radial Basis Triple Product Rendering

Wavelets have an optimal compression performance, but they still lack flexibility during rendering. There is no efficient rotation operator and they require quite a lot of preprocessing for transforming the lighting, shape and material functions to the proper wavelet basis. As a result, the forward rendering is often limited to static scenes with fixed environment lighting. In addition, the slow preprocessing stage of wavelets has a great impact on the time performance of relighting applications. To overcome some of the inherent problems of wavelet triple product rendering, we proposed a novel combination of techniques to evaluate the triple product rendering integral using spherical radial basis functions (SRBFs). Here, we did not only focus on a fast evaluation of the rendering integral, but also on the interactivity of the data itself. We argued that interactivity is a key factor in relighting applications. It is essential that all three factors can be changed dynamically during the rendering itself. We showed different techniques for transforming the environment lighting, materials and visibility data into a spherical radial basis representation.

First, we proposed a fast hierarchical approach on the GPU for transforming environment lighting, which is generally expressed in the pixel domain, into a multiscale grid of SRBFs. Each scale consists of a grid of partly overlapping SRBFs with equal lobe sizes. The amplitudes or coefficients of the SRBFs are extracted by measuring the influences of the pixels onto the SRBF. Furthermore, we extracted overlapping information for different bins of BRDF lobe sizes and positions. This overlapping information is exploited during rendering, where we can skip the SRBFs that have no influence on the rendered pixels and we only iterate over the important overlapping SRBFs.

Secondly, we are supporting dynamic shape by approximating the visibility SRBFs with cone tracing. The geometry is represented in a voxel structure which can be used for cone tracing. We defined a mapping of visibility SRBFs to the proper cones with a correct direction and radius. We observed that visibility SRBFs are only required in the areas that are reflected to the eye, more specifically inside the BRDF lobe. By efficiently subsampling the number of cones depending on the glossiness of the BRDF, we greatly reduced the number of visibility cones to trace. The more subsampling used, the more accurate hard shadows are rendered, but it also has a linear impact on the time complexity, since each visibility SRBF will require an extra triple product to evaluate. This time-quality trade-off needs to be considered and depends on the target application.

To allow for the fact that very bright light sources are not overlooked in the subdivision scheme or to avoid too much subdivision, we implemented a peak-detection algorithm. The main bright light sources in the environment map are detected and treated as a special case. For each special case, we trace more detailed—free form SRBF—cones to allow for sharp shadows. The approximation of the peaks are rather straightforward. We used a connected components approach to search for islands of bright light sources. Then, each connected component is approximated with only one or two free form SRBFs. In the future, more detailed approximations are required to better match the shape of the light source.

### 7.3 Relighting Use Cases

We have demonstrated the strengths of our general wavelet product and spherical radial basis framework in different relighting use cases. Both representations are useful in a variety of applications. The use of a general wavelets product is applicable when the application wants to exploit specific characteristics of the signals and to do so a specific wavelet function is required. On the other hand, when time complexity and interactivity is the main requirement of an application, it is beneficial to use a per pixel real-time GPU implementation of our spherical radial basis functions framework.

In our first demonstrator, we explained how our algorithms can be used for the relighting of virtual objects. We augmented omnidirectional video with realistic relit virtual objects. We have reconstructed the trajectory of an omnidirectional camera and inserted new virtual objects in the video. We showed that the omnidirectional video can be used as environment lighting for the rendering of the virtual objects. This requires our real-time transformation from video frames to SRBF lighting data.

In the second demonstrator we applied our algorithms to different relighting algorithms for real objects. We showed a practical use case of our general wavelet triple product renderer in an inverse rendering application. To allow for a hierarchical optimization algorithm, where

the lower level coefficients are estimated first and then only more detailed coefficients are inserted based on the well-posedness of the system, it is essential that the lower level coefficients are good approximations of the signal to estimate. Since the environment lighting is profoundly more smooth and high-order wavelets are ideal at representing smooth signals, we proposed to use our general triple product renderer instead of the piecewise constant Haar wavelet alternative. Besides a better refinement method, we also showed how an integration with our SRBF triple product renderer will reduce the execution time of the optimization process from hours to minutes. As a last application, we experimented with the intrinsic image decomposition problem, where we used our SRBF renderer in combination with a prior based optimization method. We adapted the rendering framework to also export the proper gradients for the optimization step.

## 7.4 Future Work

Some aspects of this dissertation are rather theoretical and require further effort to become application-ready. For example, more research is required to identify different application domains for the general triple product theorem of wavelets. We have shown a solution method where the theorem is applied to the computer graphics domain and more specifically to solve the rendering equation. In the future, the theorem could be useful in other application domains where triple product integrals are required.

More research is required to improve the time performance of the general triple product wavelet evaluation. The current sparse tensor does not allow for a linear time evaluation of the triple product integral. To accomplish this in the future, the tensor data needs to be structured in a tree format to better match the tree of wavelets that represent the three factors of the integral.

We have provided tools to allow for triple product rendering with spherical radial basis functions and developed proof of concepts for augmented reality applications and different inverse rendering applications. However, further development is required to see how they perform in other, more realistic, applications. In the future, the dynamic and interactive triple product renderer should be integrated in various applications and on different hardware platforms, e.g. the Oculus Rift [VR, 2012] or Microsoft Hololens [Microsoft, 2016]. A proof of concept of near-field lighting in the wavelet domain is presented in this dissertation. This approach can be further optimized with a SRBF representation. Near-field lighting can be applied with an adapted cone tracing algorithm. Concepts of this approach are presented by Crassin et al. [Crassin et al., 2011b,a]. Instead of just encoding occlusion information in the voxel volume, color information can be inserted as well. This color information is obtained by prerendering the entire scene in the voxel volume, including all direct lighting effects. Each traced cone throughout the volume contains additional information on the color of the scene

and possibly on emitting light sources. In addition, to reduce the total memory consumption of the voxel volumes, partially resident textures [Sellers, 2013] or lossless compression of 3D textures [Guthe and Goesele, 2016] can be used.

Although we have improved the underlying representation for both time performance and quality performance, the ambiguities that arise between the illumination and shading are far from solved completely. We have learned that there is no clear-cut solution for each scene or object. Each object requires specific user management to push the optimization process into a more plausible direction. Our speedup factor already allows for more iterations in the optimization stage and thus allows for the minimization process to invest more time in finding the correct solution rather than wasting time on the evaluation of the rendering equation. In the future, the optimization process should become much more flexible, allowing the user to have real-time previews of the current estimates and, if necessary, allowing the user to steer the optimization process whenever necessary. Unfortunately, to fully separate the shading from the lighting, more information and constraints are required. The field of machine learning is a promising lead. For example, convolutional neural network can be used to extract light sources [Augusto Dorta Marques and Walter Gonzalez Clua, 2016]. Furthermore, convolutional neural network can be used to automatically build shape, illumination and material priors, rather than building them manually.

## Bijlage A

---

### Nederlandse Samenvatting (Dutch Summary)

---

In de laatste decennia is het mixen van computer gegenereerde beelden en echte cameragebaseerde beelden steeds belangrijker aan het worden. Digitale animaties zijn niet meer weg te denken uit filmproducties en het aanpassen van filmopnames door het toevoegen van hyperrealistische virtuele objecten is een vaste waarde geworden in de filmindustrie. Daarnaast worden digitale animatietechnieken ook meer en meer toegepast op bestaande objecten in de scène. Met behulp van foto's kunnen bestaande objecten volledig worden ingescand, waarna ze digitaal worden bewerkt en gevisualiseerd onder verschillende nieuwe omstandigheden. Om dit realisme te bekomen, is het belangrijk dat de belichtings- en materiaaleffecten accuraat worden weergegeven. Het toevoegen van een computer gegenereerd virtueel object in een echte foto vereist ook dat dezelfde belichtingseigenschappen van de foto worden toegepast op het virtuele object. Dit noemt men het “herbelichten” van virtuele objecten. Met het opkomen van nieuwe apparaten zoals bijvoorbeeld de Hololens [Microsoft, 2016] zal dit alleen nog maar belangrijker worden. De Hololens is een “head-mounted display” dat een virtuele laag toevoegt aan de echte wereld die de gebruiker rondom zich waarnemt. Een goede inleving van de gebruiker is enkel mogelijk als er voldoende realistische belichting gebruikt wordt om de virtuele laag te visualiseren. Het herbelichten van echte, bestaande objecten is nog uitdagender. Hiervoor is informatie over de materiaaleigenschappen van het echte object onmisbaar. Reflectieve objecten, bijvoorbeeld spiegels, moeten de correcte omgevingsbelichting reflecteren. Het schatten van materiaaleigenschappen, samen met belichtingseigenschappen van echte objecten, wordt in de onderzoekswereld “appearance extraction” of “inverse rendering” genoemd.

Herbelichtingsalgoritmes stellen twee grote uitdagingen. Ten eerste zit alle noodzakelijke informatie—zoals materiaaleigenschappen, omgevingsbelichting en structuur—van het ob-

ject samen in een verzameling van pixels. Gezien deze informatie gebundeld zit, is het moeilijk de textuur en de belichting te onderscheiden. Het is bijvoorbeeld moeilijk te achterhalen of een “rood” object zijn rode kleur krijgt doordat het object zelf rood is, of doordat het onder rode belichting staat. Er is onvoldoende informatie om de factoren volledig te kunnen reconstrueren. Dit wordt vaak aangepakt door zoveel mogelijk informatie toe te voegen, zoals bijvoorbeeld het toevoegen van verschillende standpunten rondom het object of door het automatisch aanleren van eigenschappen van de verschillende factoren.

Een tweede grote uitdaging is het realistisch simuleren van het licht in de scène. Huidige technieken zijn hierin vaak heel beperkt. Het is cruciaal dat zo veel mogelijk globale belichtingseffecten kunnen gesimuleerd worden. Een efficiënte methode hiervoor is het gebruik van “precomputed radiance transfer”, waar de verschillende factoren, die belichting, materiaal en structuur voorstellen, vooraf worden gecomprimeerd met behulp van sferische harmonische functies of Haar wavelets. We zien echter dat het uitrekenen van de simulatie van licht nog steeds bijna 90% van de tijd inneemt. Daarnaast worden nog niet alle belichtingseffecten—zoals lichtbronnen die niet op oneindig staan—in rekening genomen. Om herbelichtingstechnieken efficiënt te kunnen inzetten, is het van primordiaal belang dat de gekozen representatie onmiddellijk geëvalueerd kan worden. In tegenstelling tot bestaande technieken zal de “inverse rendering” van een aantal uren naar een aantal minuten gereduceerd worden en zal ook het wijzigen van de parameters interactief kunnen gebeuren.

In dit doctoraat hebben we ons vooral toegespitst op de tweede uitdaging. We hebben nieuwe technieken ontwikkeld om de achterliggende representaties van herbelichtingsalgoritmes te verbeteren en op een efficiënte manier uit te rekenen. Het versnellen van de simulatie van het licht zal interactie met de gebruiker mogelijk maken. De gebruiker kan interactief virtuele objecten toevoegen aan bestaande scènes onder correcte belichting of veel sneller bestaande objecten kunnen inscannen. We focussen vooral op de tweede uitdaging, maar het verbeteren van de representaties zal ook invloed hebben op de eerste uitdaging. Een sneller werkend algoritme laat meer iteraties toe en het algoritme kan makkelijker gestuurd worden door de gebruiker, waardoor het algoritme meer tijd en rekenkracht over heeft om een betere scheiding van belichting- en materiaaleigenschappen te bekomen.

Een eerste concrete verbetering die we hebben toegepast in deze thesis is het uitbreiden van het wavelet triple product theorema naar een algemeen triple product theorema, waar naast Haar wavelets ook andere hogere order wavelets kunnen gebruikt worden die meer afgestemd zijn op het onderliggend signaal. Het gebruik van wavelets heeft de kwaliteit van herbelichtingsalgoritmes sterk verbeterd, omdat het hoog frequente belichtingseffecten toelaat. Jammer genoeg zijn de huidige technieken enkel beperkt tot het gebruik van de trapvormige Haar wavelets. Haar wavelets zijn ideaal voor het voorstellen van de structuur van het object, maar de omgevingsbelichting en de materiaaleigenschappen hebben over het algemeen meer

“gladde” eigenschappen, waardoor ze ook beter kunnen worden voorgesteld met gladde hogere order wavelets (Daubechies, Coiflets enz.). We hebben een algoritme ontwikkeld om de “bindingscoëfficiënten” van de integraal efficiënt te berekenen. Hierbij zijn we niet beperkt tot het gebruik van Haar wavelets, maar kunnen we ook de meer complexe hogere order wavelets gebruiken. In tegenstelling tot het manueel bepalen van deze coëfficiënten—zoals dat gebeurt bij Haar wavelets—is dit niet langer mogelijk bij hogere order wavelets, dankzij hun complexe overlap kenmerken. We buiten de verschillende eigenschappen van de hogere order wavelets uit om de tensor van bindingscoëfficiënten automatisch te berekenen. Zo identificeren we de overlappende wavelet functies en kunnen we de complexiteit terugdringen door gebruik te maken van de breedte, symmetrie en nulpunten van de wavelet. We hebben een robuuste oplossingsmethode ontwikkeld, die niet alleen nuttig is voor het gebruik in een herbelichtingstoepassing, maar ook kan gebruikt worden in andere domeinen die signaalverwerking toepassen.

In het tweede deel hebben we ons toegelegd op de interactiviteit en de snelheid van huidige herbelichtingsalgoritmes. Het gebruik van wavelets is ideaal voor applicaties waar een goede compressie noodzakelijk is. Desalniettemin is het uitrekenen van de rendering integraal voor de simulatie van licht niet flexibel genoeg om interactief beelden te genereren. Zo bestaat er geen efficiënte rotatie operator en zijn de drie factoren statisch, omdat ze op voorhand moeten worden uitgerekend. Om tegemoet te komen aan deze problemen stellen we het gebruik van sferische radiale basis functies (SRBFs) voor. We presenteren een unieke combinatie van technieken om de triple product integraal uit te rekenen met behulp van sferische radiale basis functies. De literatuur heeft al bewezen dat een snelle evaluatie mogelijk is, maar wij besteden ook aandacht aan de interactiviteit van de data zelf. We beargumenteren dat het essentieel is dat de gebruiker interactief de belichtings-, materiaal-, en structuureigenschappen kan aanpassen. We hebben hiervoor een hiërarchisch algoritme ontwikkeld op de GPU om de 360 graden omgevingsbelichting, voorgesteld in het pixeldomein, om te zetten naar een hiërarchisch rooster van SRBFs. De overlap informatie van het hiërarchisch rooster wordt uitgebuit om een snelle rendering mogelijk te maken. Animaties van het object, oftewel het aanpassen van de structuur, wordt toegelaten door de zichtbaarheidsfactor van de integraal te bemonsteren met behulp van “voxel cone tracing”. De structuur van het object wordt omgezet in een voxel volume waar in functie van de materiaaleigenschappen correcte kegels worden in getraceerd. In het geval van een heldere lichtbron is het noodzakelijk om de zichtbaarheidsfactor nauwkeuriger te bemonsteren. Heldere lichtbronnen worden geïdentificeerd met behulp van een “piek detectie” algoritme.

Om aan te tonen hoe bovenstaande representaties huidige herbelichtingsalgoritmes verbeteren, hebben we twee gebruikstoepassingen ontwikkeld. In een eerste toepassing tonen we een “augmented reality” applicatie, waar we onze SRBF representatie gebruiken om virtuele objecten toe te voegen aan een 360 graden video. Het virtuele object wordt uitgelijnd met de video en herbelicht volgens de belichtingsomstandigheden van de video zelf. We tonen

aan hoe de beelden van de 360 graden video gebruikt worden als belichtingsfactor van de triple product rendering integraal. Naast het herbelichten van virtuele objecten, hebben we ook onderzocht hoe onze nieuwe representaties kunnen geïntegreerd worden in bestaande “inverse rendering” algoritmes. In de tweede toepassing hebben we twee bestaande herbelichtingsalgoritmes geïmplementeerd en aangepast aan onze representatie. De eerste techniek die we hebben getest is de “inverse rendering” techniek van Haber et al. [Haber et al., 2009]. De gladde hogere order wavelets helpen bij een betere regularisatie van het optimalisatieprobleem. Daarnaast zorgt het gebruik van SRBFs voor een veel snellere evaluatie van de onbekenden en laat het toe om de schatting terug te dringen naar een aantal minuten, daar waar het eerst makkelijk een paar uur in beslag nam. In een tweede voorbeeld gebruiken we onze SRBF representatie in het “intrinsic image decomposition” algoritme van Barron en Malik [Barron and Malik, 2015]. We vergelijken hoe onze representatie presteert ten opzichte van hun originele laag-frequente sferische harmonische functies.

We kunnen besluiten dat we in dit proefschrift hebben aangetoond dat de manier waarop de belichting, materialen en structuur worden voorgesteld een grote invloed heeft op de kwaliteit en snelheid van herbelichtingsalgoritmes. We hebben geleerd dat een verbetering van de onderliggende representaties essentieel is om de kloof tussen de virtuele en de echte wereld te helpen dichten.

## Appendix B

---

### Scientific Contributions and Publications

---

The following list of publications, presented at scientific international conferences, contains work that is part of this dissertation:

- [**Michiels et al., 2013**] Michiels, N., Put, J., Haber, T., Klaudiny, M., and Bekaert, P. (2013). High-order wavelets for hierarchical refinement in inverse rendering. In *ACM SIGGRAPH 2013 Posters*, SIGGRAPH '13, pages 99:1–99:1, New York, NY, USA. ACM
- [**Put et al., 2014**] Put, J., Michiels, N., and Bekaert, P. (2014). Exploiting material properties to select a suitable wavelet basis for efficient rendering. In *GRAPP 2014 - Proceedings of the 9th International Conference on Computer Graphics Theory and Applications, Lisbon, Portugal, 5-8 January, 2014.*, pages 218–224, Lisbon, Portugal. INSTICC
- [**Michiels et al., 2014b**] Michiels, N., Put, J., and Bekaert, P. (2014b). Product integral binding coefficients for high-order wavelets. In *SIGMAP 2014 - Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications, Vienna, Austria, 28-30 August, 2014*, pages 17–24, Vienna, Austria
- [**Michiels et al., 2014a**] Michiels, N., Jorissen, L., Put, J., and Bekaert, P. (2014a). Interactive augmented omnidirectional video with realistic lighting. In *Augmented and Virtual Reality - First International Conference, AVR 2014, Lecce, Italy, September 17-20, 2014, Revised Selected Papers*, pages 247–263, Lecce, Italy
- [**Michiels et al., 2015**] Michiels, N., Put, J., and Bekaert, P. (2015). Interactive relighting of virtual objects under environment lighting. In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications (GRAPP 2015), Berlin, Germany, 11-14 March, 2015*, pages 220–228. SciTePress

**[Put et al., 2015]** Put, J., Michiels, N., and Bekaert, P. (2015). Using near-field light sources to separate illumination from brdf. In Xie, X., Jones, M. W., and K. L. Tam, G., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 16.1–16.13. BMVA Press

**[Put et al., 2016]** Put, J., Michiels, N., and Bekaert, P. (2016). Material specific chromaticity priors. In Wilson, R. C., Hancock, E. R., and A. P. Smith, W., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 25.1–25.10. BMVA Press

The following work is not part of this dissertation:

**[Jorissen et al., 2014]** Jorissen, L., Goorts, P., Bex, B., Michiels, N., Rogmans, S., Bekaert, P., and Lafruit, G. (2014). A qualitative comparison of mpeg view synthesis and light field rendering. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2014*, pages 1–4, Budapest, Hungary

**[Chenchu et al., 2016]** Chenchu, R., Michiels, N., Rogmans, S., and Bekaert, P. (2016). Simplification of moving 3d scene data on gpu. In *SIGMAP 2016 - Proceedings of the 13th International Conference on Signal Processing and Multimedia Applications, Lisbon, Portugal, July, 2016*, pages 95–98, Lisbon, Portugal

Related student theses:

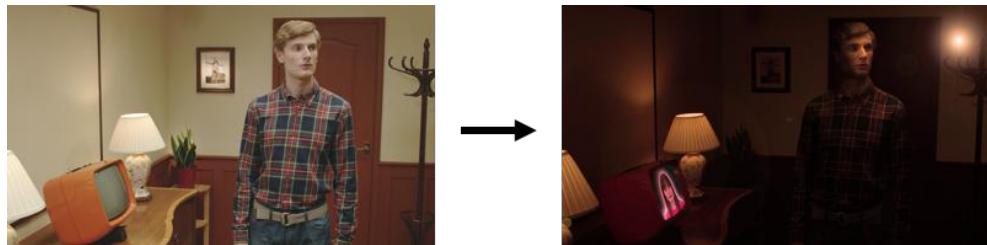
**[Metzmacher et al., 2012]** Metzmacher, H., Bloembergen, D., Alers, S., Michiels, N., Maessen, S., and Tuylsland, K. (2012). Semi-Automatic 3D Object Extraction from Image Collections. Master's thesis, Maastricht University, Maastricht, The Netherlands

**[Van Nerum et al., 2014]** Van Nerum, K., Put, J., Michiels, N., and Bekaert, P. (2014). Real-Time GPU Ray Tracing. Bachelor's thesis, Hasselt University, Hasselt, Belgium

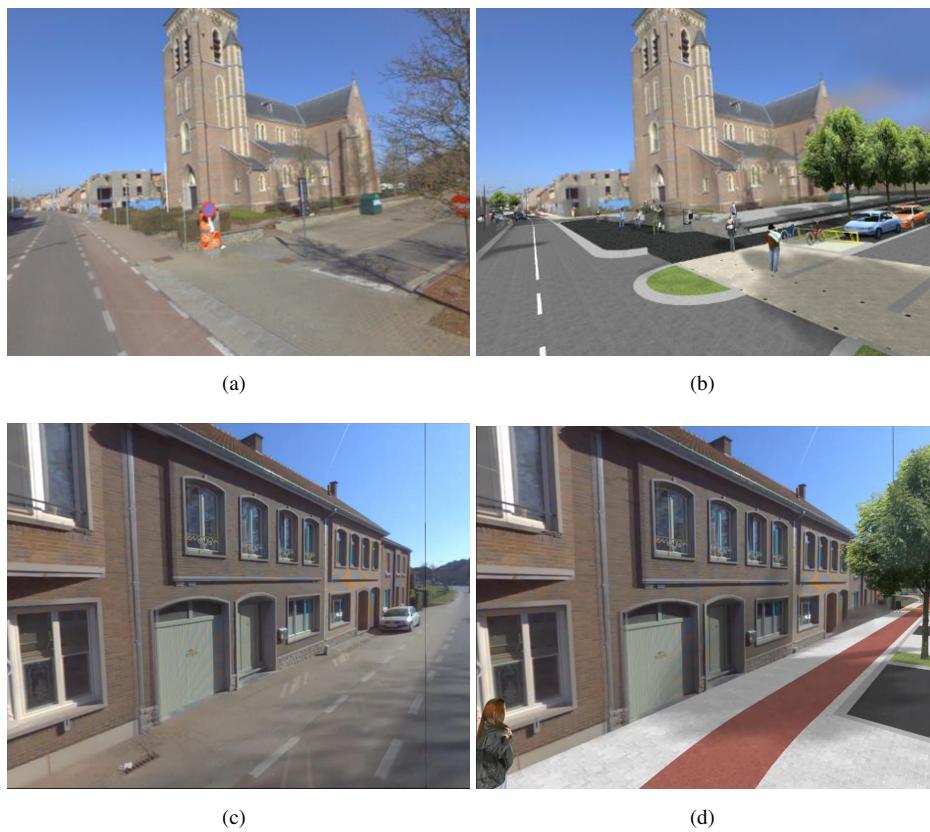
**[Scarlino et al., 2015]** Scarlino, D., Michiels, N., Put, J., and Bekaert, P. (2015). Real-time Ambient Occlusion. Master's thesis, Hasselt University, Hasselt, Belgium

Here is a list of other public dissemination:

- Demonstrator for the SCENE consortium (FP7-288238 SCENE) in 2014. We demonstrated relighting of a temporal scene in the context of a movie post production. A result is shown in Figure B.1.
- Demonstrator for the AVIE consortium in 2014. The demonstrator paired a virtual scene with real omnidirectional video. Our real-time renderer is used to allow virtual and real objects to cast shadows on each other. This is shown in Figure B.2.
- Demonstrator for the HiViz Lab Opening in 2016. This demonstrator showed our real-time SRBF framework where real and virtual objects were being relit with a live environment map, captured by a omnidirectional camera. This is shown in Figure B.3.



**Figure B.1:** Demonstrator of our relighting technique in the context of a movie post production. Demonstration for the SCENE project, 2014.



**Figure B.2:** Public demonstrator for merging virtual and real content in the context of the re-design of public roads. A basis mesh reconstruction is used to accurately cast shadows from virtual objects onto real objects and vice versa.



**Figure B.3:** Live demonstrator at HiViz 2016 Lab opening

---

## Bibliography

---

- Adelson, E. H. and Pentland, A. P. (1996). The perception of shading and reflectance. In Knill, D. C. and Richards, W., editors, *Perception As Bayesian Inference*, chapter The Perception of Shading and Reflectance, pages 409–423. Cambridge University Press, New York, NY, USA.
- Agusanto, K., Li, L., Chuangui, Z., and Sing, N. W. (2003). Photorealistic rendering for augmented reality using environment illumination. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 208–216.
- Arief, I., McCallum, S., and Hardeberg, J. Y. (2012). Realtime estimation of illumination direction for augmented reality on mobile devices. In *Color and Imaging Conference*, pages 111–116, Los Angeles, CA, USA. IS&T and SID.
- Artusi, A., Banterle, F., and Chetverikov, D. (2011). A survey of specularity removal methods. *Computer Graphics Forum*, 30(8):2208–2230.
- Augusto Dorta Marques, B. and Walter Gonzalez Clua, E. (2016). Gpu accelerated method for estimation of light-sources. In *The GPU Technology Conference 2016 (GTC 2013)*, GTC 2016.
- Barron, J. T. and Malik, J. (2011). High-frequency shape and albedo from shading using natural image statistics. *CVPR*.
- Barron, J. T. and Malik, J. (2012). Shape, albedo, and illumination from a single image of an unknown object. *CVPR*.
- Barron, J. T. and Malik, J. (2015). Shape, illumination, and reflectance from shading. *TPAMI*.
- Barrow, H. (1978). Recovering intrinsic scene characteristics from images. *Computer Vision Systems*, pages 3–26.
- Belhumeur, P. N., Kriegman, D. J., and Yuille, A. L. (1999). The bas-relief ambiguity. *Int. J. Comput. Vision*, 35(1):33–44.

- Berman, D. F., Bartell, J. T., and Salesin, D. H. (1994). Multiresolution painting and compositing. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 85–90. ACM.
- Beylkin, G., Coifman, R., and Rokhlin, V. (1991). Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44(2):141–183.
- Blinn, J. F. (1977). 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. ACM.
- Bonneau, G.-P. (1999). Optimal triangular haar bases for spherical data. *LMC - CRNS*.
- Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M. (2001). Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 425–432, New York, NY, USA. ACM.
- Chen, Q. and Koltun, V. (2013). A simple model for intrinsic image decomposition with depth cues. In *2013 IEEE International Conference on Computer Vision*, pages 241–248.
- Chenchu, R., Michiels, N., Rogmans, S., and Bekaert, P. (2016). Simplification of moving 3d scene data on gpu. In *SIGMAP 2016 - Proceedings of the 13th International Conference on Signal Processing and Multimedia Applications, Lisbon, Portugal, July, 2016*, pages 95–98, Lisbon, Portugal.
- Christensen, P., Stollnitz, E., Salesin, D., and DeRose, T. (1995). Wavelet radiance. In *Photorealistic Rendering Techniques*, pages 295–309. Springer.
- Chui, C. K. (1992). *An Introduction to Wavelets*. Academic Press Professional, Inc., San Diego, CA, USA.
- Clapham, C. and Nicholson, J. (2009). *The Concise Oxford Dictionary of Mathematics*. Oxford Paperback Reference. OUP Oxford.
- Clarberg, P., Jarosz, W., Akenine-Möller, T., and Jensen, H. W. (2005). Wavelet importance sampling: efficiently evaluating products of complex functions. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1166–1175, New York, NY, USA. ACM.
- Cook, R. L. and Torrance, K. E. (1981). A reflectance model for computer graphics. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81, pages 307–316, New York, NY, USA. ACM.
- Cooley, J. and Tukey, J. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

- Crassin, C. and Green, S. (2012). Octree-based sparse voxelization using the gpu hardware rasterizer. In Cozzi, P. and Riccio, C., editors, *OpenGL Insights*, pages 303–318. CRC Press.
- Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E. (2011a). Interactive indirect illumination using voxel-based cone tracing: An insight. In *ACM SIGGRAPH 2011 Talks*, SIGGRAPH ’11, pages 20:1–20:1, New York, NY, USA. ACM.
- Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E. (2011b). Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proc. of Pacific Graphics 2011)*.
- Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996.
- Daubechies, I. (1992). *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
- Debevec, P. (1998). 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, NY, USA. ACM.
- Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., and Sagar, M. (2000). Acquiring the reflectance field of a human face. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’00, pages 145–156, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- Debevec, P., Tchou, C., Gardner, A., Hawkins, T., Poullis, C., Stumpfel, J., Jones, A., Yun, N., Einarsson, P., Lundgren, T., and Fajardo, M. (2004). Estimating surface reflectance properties of a complex scene under captured natural illumination. conditionally accepted to. *ACM Transactions on Graphics*.
- DeVore, R. A., Jawerth, B., and Lucier, B. J. (1992). Image compression through wavelet transform coding. *Information Theory, IEEE Transactions on*, 38(2):719–746.
- Duchêne, S., Riant, C., Chaurasia, G., Lopez-Moreno, J., Laffont, P.-Y., Popov, S., Bousseau, A., and Drettakis, G. (2015). Multi-view intrinsic images of outdoors scenes with an application to relighting. *ACM Transactions on Graphics*.
- Dumont, M., Maesen, S., Frederix, K., Raymaekers, C., Bekaert, P., and Reeth, F. (2010). Immersive collaboration environment. In Nitto, E. and Yahyapour, R., editors, *Towards a Service-Based Internet: Third European Conference, ServiceWave 2010, Ghent, Belgium, December 13–15, 2010. Proceedings*, pages 187–188. Springer Berlin Heidelberg.

- Dumont, M., Rogmans, S., Maesen, S., Frederix, K., Taelman, J., and Bekaert, P. (2011). A spatial immersive office environment for computer-supported collaborative work - moving towards the office of the future. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGMAP 2011, pages 212–216.
- Durou, J.-D., Falcone, M., and Sagona, M. (2008). Numerical methods for shape-from-shading: A new survey with benchmarks. *Comput. Vis. Image Underst.*, 109(1):22–43.
- Dutre, P., Bala, K., Bekaert, P., and Shirley, P. (2006). *Advanced Global Illumination*. AK Peters Ltd.
- Dutre, P., Heckbert, P., Ma, V., Pellacini, F., Porschka, R., Ramasubramanian, M., Soler, C., and Ward, G. (2001). Global illumination compendium.
- Ferrari, S., Maggioni, M., and Borghese, N. A. (2004). Multiscale approximation with hierarchical radial basis functions networks. *IEEE Transactions on Neural Networks*, 15(1):178–188.
- Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12):2379–2394.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Fraundorfer, F. and Scaramuzza, D. (2012). Visual odometry : Part ii: Matching, robustness, optimization, and applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90.
- Gertz, E. M. and Wright, S. J. (2003). Object-oriented software for quadratic programming. *ACM Trans. Math. Softw.*, 29(1):58–81.
- Gierlinger, T., Danch, D., and Stork, A. (2010). Rendering techniques for mixed reality. *Journal of Real-Time Image Processing*, 5(2):109–120.
- Gijsenij, A., Gevers, T., and van de Weijer, J. (2011). Computational color constancy: Survey and experiments. *Image Processing, IEEE Transactions on*, 20(9):2475–2489.
- Gorski, K., Hivon, E., Banday, A., Wandelt, B., Hansen, F., et al. (2005). HEALPix - A Framework for high resolution discretization, and fast analysis of data distributed on the sphere. *Astrophys.J.*, 622:759–771.
- Gortler, S. J., Schröder, P., Cohen, M. F., and Hanrahan, P. (1993). Wavelet radiosity. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 221–230, New York, NY, USA. ACM.

- Grosch, T. (2005). PanoAR: Interactive augmentation of omni-directional images with consistent lighting. In *Mirage 2005, Computer Vision / Computer Graphics Collaboration Techniques and Applications*, pages 25–34.
- Grosch, T., Eble, T., and Mueller, S. (2007). 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. ACM.
- Gupta, M. and Narasimhan, S. G. (2007). Legendre polynomials triple product integral and lower-degree approximation of polynomials using chebyshev polynomials. Technical Report CMU-RI-TR-07-22, Robotics Institute, Pittsburgh, PA.
- Guthe, S. and Goesele, M. (2016). Gpu-based lossless volume data compression. In *2016 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4.
- Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 71(1):38–53.
- Haber, T. (2015). *Acquiring the World through Photographs*. PhD thesis, Hasselt University, Hasselt, Belgium.
- Haber, T., Fuchs, C., Bekaer, P., Seidel, H.-P., Goesele, M., and Lensch, H. (2009). Relighting objects from image collections. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 627–634.
- Hachama, M., Ghanem, B., and Wonka, P. (2015). Intrinsic scene decomposition from rgbd images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 810–818.
- Haller, M. (2004). Photorealism or/and non-photorealism in augmented reality. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry, VRCAI ’04*, pages 189–196, New York, NY, USA. ACM.
- Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition.
- Holografika (2016). Holografika - holographic display technology. <http://www.holografika.com/>. Accessed February 25, 2016.
- Horn, B. K. (1970). Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

- Horn, B. K. P. (1989). Shape from shading. In Horn, B. K. P. and Brooks, M. J., editors, *Shape from Shading*, chapter Obtaining Shape from Shading Information, pages 123–171. MIT Press, Cambridge, MA, USA.
- Imagine Engine (2016 (accessed May 2, 2016)). *Jurassic World Breakdown*. <http://image-engine.com/film/jurassic-world/>.
- Imber, J., Guillemaut, J.-Y., and Hilton, A. (2014). *Intrinsic Textures for Relightable Free-Viewpoint Video*, pages 392–407. Springer International Publishing, Cham.
- Immel, D. S., Cohen, M. F., and Greenberg, D. P. (1986). 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. ACM.
- Inger, Y., Farbman, Z., and Lischinski, D. (2013). Locally adaptive products for all-frequency relighting. *Computer Graphics Forum (Proceedings of Eurographics 2013)*, 32(2pt1):73–82.
- Iwasaki, K., Furuya, W., Dobashi, Y., and Nishita, T. (2012). Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. *Comp. Graph. Forum*, 31(2pt4):727–734.
- Jorissen, L., Goorts, P., Bex, B., Michiels, N., Rogmans, S., Bekaert, P., and Lafruit, G. (2014). A qualitative comparison of mpeg view synthesis and light field rendering. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2014*, pages 1–4, Budapest, Hungary.
- Jylänki, J. (2010). A thousand ways to pack the bin - a practical approach to two-dimensional rectangle bin packing. Retrieved from <http://clb.demon.fi/files/RectangleBinPack.pdf>.
- Kajiya, J. T. (1986). 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. ACM.
- Kanbara, M. and Yokoya, N. (2004). Real-time estimation of light source environment for photorealistic augmented reality. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 911–914 Vol.2.
- Kautz, J., Sloan, P.-P., and Snyder, J. (2002). Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *Proceedings of the 13th Eurographics workshop on Rendering*, EGRW '02, pages 291–296, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- Khronos Group (2016). *OpenCL*. <https://www.khronos.org/opencl/>.

- Kingsbury, N. and Magarey, J. (1998). Wavelet transforms in image processing. In *Signal analysis and prediction*, pages 27–46. Springer.
- Klaudiny, M. (2013). *High-detail temporally consistent 3D capture of facial performance*. PhD thesis, Centre for Vision, Speech and Signal Processing, Faculty of Engineering and Physical Sciences, University of Surrey, Guildford, Surrey, U.K.
- Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. (2015). Deep convolutional inverse graphics network. *CoRR*, abs/1503.03167.
- Laffont, P., Bousseau, A., and Drettakis, G. (2013). Rich intrinsic image decomposition of outdoor scenes from multiple views. *Visualization and Computer Graphics, IEEE Transactions on*, 19(2):210–224.
- Laffont, P.-Y., Bousseau, A., Paris, S., Durand, F., and Drettakis, G. (2012). Coherent intrinsic images from photo collections. *ACM Trans. Graph.*, 31(6):202:1–202:11.
- Lam, P.-M., Ho, T.-Y., Leung, C.-S., and Wong, T.-T. (2010). All-frequency lighting with multiscale spherical radial basis functions. *Visualization and Computer Graphics, IEEE Transactions on*, 16(1):43–56.
- Land, E. H., John, and Mccann, J. (1971). Lightness and retinex theory. *Journal of the Optical Society of America*, pages 1–11.
- Lee, K. J., Zhao, Q., Tong, X., Gong, M., Izadi, S., Lee, S. U., Tan, P., and Lin, S. (2012). Estimation of intrinsic image sequences from image+depth video. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV’12, pages 327–340, Berlin, Heidelberg. Springer-Verlag.
- Lensch, H. P. A., Kautz, J., Goesele, M., Heidrich, W., and Seidel, H.-P. (2003). Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.*, 22(2):234–257.
- Lessig, C. and Fiume, E. (2008). SOHO: Orthogonal and Symmetric Haar Wavelets on the Sphere. *ACM Trans. Graph.*, 27(1).
- Levoy, M. and Hanrahan, P. (1996). Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’96, pages 31–42, New York, NY, USA. ACM.
- Liu, X., Sloan, P.-P., Shum, H.-Y., and Snyder, J. (2004). All-frequency precomputed radiance transfer for glossy objects. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*, EGSR’04, pages 337–344, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

- Lourakis, M. I. A. and Argyros, A. A. (2009). Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Softw.*, 36(1):2:1–2:30.
- Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA. IEEE Computer Society.
- Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Ma, W.-C., Hsiao, C.-T., Lee, K.-Y., Chuang, Y.-Y., and Chen, B.-Y. (2006). Real-time triple product relighting using spherical local-frame parameterization. *The Visual Computer*, 22(9):682–692.
- Marschner, S. R. (1998). *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University, Ithaca, NY, USA. AAI9839924.
- Marschner, S. R. and Greenberg, D. P. (1997). Inverse lighting for photography. In *IN FIFTH COLOR IMAGING CONFERENCE*, pages 262–265.
- Marschner, S. R., Westin, S. H., Lafortune, E. P. F., Torrance, K. E., and Greenberg, D. P. (1999). Image-based brdf measurement including human skin. In *Proceedings of the 10th Eurographics Conference on Rendering*, EGWR'99, pages 131–144, Aire-la-Ville, Switzerland. Eurographics Association.
- Matusik, W., Loper, M., and Pfister, H. (2004). Progressively-refined reflectance functions from natural illumination. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*, EGSR'04, pages 299–308, Aire-la-Ville, Switzerland. Eurographics Association.
- Matusik, W., Pfister, H., Brand, M., and McMillan, L. (2003). A data-driven reflectance model. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 759–769, New York, NY, USA. ACM.
- Mei, X., Ling, H., and Jacobs, D. (2009). Sparse representation of cast shadows via 11-regularized least squares. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 583–590.
- Meka, A., Zollhöfer, M., Richardt, C., and Theobalt, C. (2016). Live intrinsic video. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 35(4).
- Metzmacher, H., Bloembergen, D., Alers, S., Michiels, N., Maesen, S., and Tuylsland, K. (2012). Semi-Automatic 3D Object Extraction from Image Collections. Master's thesis, Maastricht University, Maastricht, The Netherlands.

- Meunier, S., Perrot, R., Aveneau, L., Meneveaux, D., and Ghazanfarpour, D. (2010). Cosine lobes for interactive direct lighting in dynamic scenes. *Computers & Graphics*, 34(6):767–778.
- Meyer, Y. (1993). *Wavelets - Algorithms and applications*. Society for Industrial and Applied Mathematics.
- Michiels, N., Jorissen, L., Put, J., and Bekaert, P. (2014a). Interactive augmented omnidirectional video with realistic lighting. In *Augmented and Virtual Reality - First International Conference, AVR 2014, Lecce, Italy, September 17-20, 2014, Revised Selected Papers*, pages 247–263, Lecce, Italy.
- Michiels, N., Put, J., and Bekaert, P. (2014b). Product integral binding coefficients for high-order wavelets. In *SIGMAP 2014 - Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications, Vienna, Austria, 28-30 August, 2014*, pages 17–24, Vienna, Austria.
- Michiels, N., Put, J., and Bekaert, P. (2015). Interactive relighting of virtual objects under environment lighting. In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications (GRAPP 2015), Berlin, Germany, 11-14 March, 2015*, pages 220–228. SciTePress.
- Michiels, N., Put, J., Haber, T., Klaudiny, M., and Bekaert, P. (2013). High-order wavelets for hierarchical refinement in inverse rendering. In *ACM SIGGRAPH 2013 Posters, SIGGRAPH '13*, pages 99:1–99:1, New York, NY, USA. ACM.
- Microsoft (2016). Microsoft hololens. <https://www.microsoft.com/microsoft-hololens/en-us>. Accessed February 25, 2016.
- Miller, G. and C., H. (1984). Illumination and reflection maps: Simulated objects in simulated and real environments. *Advanced Computer Graphics Animation seminar notes*.
- Mohlenkamp, M. J. (1997). *A Fast Transform for Spherical Harmonics*. PhD thesis, Yale University, New Haven, CT, USA. UMI Order No. GAX97-33952.
- Mortensen, J. (2011). *Virtual light fields for global illumination in computer graphics*. PhD thesis, UCL (University College London).
- Narihira, T., Maire, M., and Yu, S. X. (2015). Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2992–2992.
- Ng, R., Ramamoorthi, R., and Hanrahan, P. (2003). All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph.*, 22(3):376–381.

- Ng, R., Ramamoorthi, R., and Hanrahan, P. (2004). Triple product wavelet integrals for all-frequency relighting. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 477–487, New York, NY, USA. ACM.
- Ngan, A., Durand, F., and Matusik, W. (2005). Experimental analysis of brdf models. In *Proceedings of the Sixteenth Eurographics conference on Rendering Techniques*, EGSR'05, pages 117–126, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- Nielson, G. M., Jung, I.-H., and Sung, J. (1997). Haar wavelets over triangular domains with applications to multiresolution models for flow over a sphere. In *Proceedings of the 8th Conference on Visualization '97*, VIS '97, pages 143–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.
- Nimeroff, J., Simoncelli, E., and Dorsey, J. (1995). Efficient re-rendering of naturally illuminated environments. In Sakas, G., Müller, S., and Shirley, P., editors, *Photorealistic Rendering Techniques*, Focus on Computer Graphics, pages 373–388. Springer Berlin Heidelberg.
- Nowrouzezahrai, D., Simari, P., Kalogerakis, E., and Fiume, E. (2007). Eigentransport for efficient and accurate all-frequency relighting. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 163–169, New York, NY, USA. ACM.
- NVIDIA (2016). *CUDA Best Practices Guide*. <http://docs.nvidia.com/cuda/>.
- OpenGL (2016). *OpenGL Shading Language*. <https://www.opengl.org/documentation/glsl/>.
- Papagiannakis, G. and Foni, R. (2005). N.: Practical precomputed radiance transfer for mixed reality. In *In: Proceedings of Virtual Systems and Multimedia 2005*, pages 189–199. VSMM Society.
- Peers, P. and Dutré, P. (2003). Wavelet environment matting. In *Proceedings of the Eurographics Symposium on Rendering*, pages 157–166, Belgium.
- Peers, P. and Dutré, P. (2005). Inferring reflectance functions from wavelet noise. In *Proceedings of the Sixteenth Eurographics conference on Rendering Techniques*, pages 173–182, Konstanz, Germany. Eurographics Association.
- Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317.
- PointGrey (2014). Ladybug3. Web page. <http://www.ptgrey.com/>. Accessed October 13 , 2014.

- Pont, S. C. and te Pas, S. F. (2006). Material-illumination ambiguities and the perception of solid objects. *Perception*, 35(10):1331–1350.
- Praun, E. and Hoppe, H. (2003). Spherical parametrization and remeshing. *ACM TOG*, 22(3):340–349.
- Put, J., Michiels, N., and Bekaert, P. (2014). Exploiting material properties to select a suitable wavelet basis for efficient rendering. In *GRAPP 2014 - Proceedings of the 9th International Conference on Computer Graphics Theory and Applications, Lisbon, Portugal, 5-8 January, 2014.*, pages 218–224, Lisbon, Portugal. INSTICC.
- Put, J., Michiels, N., and Bekaert, P. (2015). Using near-field light sources to separate illumination from brdf. In Xie, X., Jones, M. W., and K. L. Tam, G., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 16.1–16.13. BMVA Press.
- Put, J., Michiels, N., and Bekaert, P. (2016). Material specific chromaticity priors. In Wilson, R. C., Hancock, E. R., and A. P. Smith, W., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 25.1–25.10. BMVA Press.
- Ramamoorthi, R. and Hanrahan, P. (2001). 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. ACM.
- Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. (1998). The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’98, pages 179–188, New York, NY, USA. ACM.
- Romeiro, F., Vasilyev, Y., and Zickler, T. (2008). Passive reflectometry. In *Proceedings of the 10th European Conference on Computer Vision: Part IV*, ECCV ’08, pages 859–872, Berlin, Heidelberg. Springer-Verlag.
- Ruderman, D. L. and Bialek, W. (1994). Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814.
- Salvi, J., Pagès, J., and Batlle, J. (2004). Pattern codification strategies in structured light systems. *PATTERN RECOGNITION*, 37:827–849.
- Samsung (2016). Samsung gear vr. <http://www.samsung.com/global/galaxy/wearables/gear-vr/>. Accessed February 25, 2016.
- Sato, I., Sato, Y., and Ikeuchi, K. (2003). Illumination from shadows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(3):290–300.
- Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry : Part i - the first 30 years and fundamentals. *Robotics Automation Magazine, IEEE*, 18(4).

- Scarlino, D., Michiels, N., Put, J., and Bekaert, P. (2015). Real-time Ambient Occlusion. Master's thesis, Hasselt University, Hasselt, Belgium.
- Schoeneman, C., Dorsey, J., Smits, B., Arvo, J., and Greenberg, D. (1993). Painting with light. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 143–146, New York, NY, USA. ACM.
- Schröder, P. and Sweldens, W. (1995). Spherical wavelets: Efficiently representing functions on the sphere. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 161–172, New York, NY, USA. ACM.
- Sellers, G. (2013). Opengl extension specification: Arb\_sparse\_texture. Web page. [https://www.opengl.org/registry/specs/ARB/sparse\\_texture.txt](https://www.opengl.org/registry/specs/ARB/sparse_texture.txt). Accessed October 11, 2014.
- Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600.
- Sloan, P.-P., Hall, J., Hart, J., and Snyder, J. (2003). Clustered principal components for precomputed radiance transfer. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 382–391, New York, NY, USA. ACM.
- Sloan, P.-P., Kautz, J., and Snyder, J. (2002). 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. ACM.
- Stollnitz, E. J., DeRose, A. D., and Salesin, D. H. (1995). Wavelets for computer graphics: a primer.1. *IEEE Computer Graphics and Applications*, 15(3):76–84.
- Sun, B. and Ramamoorthi, R. (2009). Affine double- and triple-product wavelet integrals for rendering. *ACM Trans. Graph.*, 28(2):14:1–14:17.
- Sun, W. and Mukherjee, A. (2006). Generalized wavelet product integral for rendering dynamic glossy objects. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 955–966, New York, NY, USA. ACM.
- Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186 – 200.
- Sweldens, W. (1998). The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546.

- Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, UK. Springer-Verlag.
- Tsai, Y.-T. and Shih, Z.-C. (2006). All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 967–976, New York, NY, USA. ACM.
- Van Nerum, K., Put, J., Michiels, N., and Bekaert, P. (2014). Real-Time GPU Ray Tracing. Bachelor's thesis, Hasselt University, Hasselt, Belgium.
- VR, O. (2012). Oculus rift. <https://www.oculus.com/en-us/rift/>. Accessed February 25, 2016.
- Wang, J., Ren, P., Gong, M., Snyder, J., and Guo, B. (2009). All-frequency rendering of dynamic, spatially-varying reflectance. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 133:1–133:10, New York, NY, USA. ACM.
- Wang, R., Ng, R., Luebke, D., and Humphreys, G. (2006). Efficient wavelet rotation for environment map rendering. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, pages 173–182, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- Ward, G. J. (1992). 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. ACM.
- Weistroffer, R. P., Walcott, K. R., Humphreys, G., and Lawrence, J. (2007). Efficient basis decomposition for scattered reflectance data. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 207–218, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- Ye, G., Garces, E., Liu, Y., Dai, Q., and Gutierrez, D. (2014). Intrinsic video and applications. *ACM Trans. Graph.*, 33(4):80:1–80:11.
- Yu, T., Wang, H., Ahuja, N., and Chen, W.-C. (2006a). Sparse lumigraph relighting by illumination and reflectance estimation from multi-view images. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR '06, pages 41–50, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- Yu, T., Wang, H., Ahuja, N., and Chen, W.-C. (2006b). Sparse lumigraph relighting by illumination and reflectance estimation from multi-view images. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA. ACM.
- Yu, Y., Debevec, P., Malik, J., and Hawkins, T. (1999). Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual*

- conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 215–224, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- Yu, Y. and Malik, J. (1998). Recovering photometric properties of architectural scenes from photographs. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 207–217, New York, NY, USA. ACM.
- Zhang, R., Tsai, P.-S., Cryer, J., and Shah, M. (1999). Shape-from-shading: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):690–706.
- Zickler, T., Ramamoorthi, R., Enrique, S., and Belhumeur, P. N. (2006). Reflectance sharing: Predicting appearance from a sparse set of images of a known shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1287–1302.