User-Guided Scene Stylization using Efficient Rendering Techniques

---

Dissertation zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften der
Naturwissenschaftlich-Technischen Fakultäten der
Universität des Saarlandes

von

Oliver Klehm

August 2015

Betreuender Hochschullehrer – Supervisor
Prof. Dr. Hans-Peter Seidel, MPI Informatik, Saarbrücken, Germany

Gutachter – Reviewer
Prof. Dr. Elmar Eisemann, Delft University of Technology, Delft, The Netherlands
Prof. Dr. Hans-Peter Seidel, MPI Informatik, Saarbrücken, Germany
Prof. Dr. Diego Gutiérrez, University of Zaragoza, Zaragoza, Spain

Dekan – Dean
Prof. Dr. Markus Bläser, Universität des Saarlandes, Saarbrücken, Germany

Kolloquium – Examination

Datum – Date
12. Feb. 2016

Vorsitzender – Chair
Prof. Dr. Philipp Slusallek, Universität des Saarlandes, Saarbrücken, Germany

Prüfer – Examiners
Prof. Dr. Elmar Eisemann, Delft University of Technology, Delft, The Netherlands
Prof. Dr. Hans-Peter Seidel, MPI Informatik, Saarbrücken, Germany
Prof. Dr. Diego Gutiérrez, University of Zaragoza, Zaragoza, Spain

Protokoll – Reporter
Dr. Christian Richardt, MPI Informatik, Saarbrücken, Germany

Copyright
Oliver Klehm, 2015
All rights reserved.

Abstract

In this dissertation, we propose new techniques to display and manipulate virtual worlds to support visual design.

The real-time constraint of applications such as games limits the accuracy at which rendering algorithms can simulate light transport. Our first method focuses on the efficient rendering of surfaces under natural illumination, extending previous work that ignores directionally-dependent effects in the lighting. In a second work, we present an approach for the efficient computation of scattering in homogeneous participating media. The main challenge is the accumulation of visibility along view rays, which we solve using an efficient filtering scheme.

In the second part of the dissertation, we investigate methods that provide artists with approaches to stylize and manipulate the appearance of volumetric scattering. First, we focus on the effect of light shafts that one can typically observe on hazy days due to openings in the clouds. While the effect is often used in games and movies, it is difficult to manipulate. We propose tools to directly manipulate parameters of the rendering, effectively providing control over the creation, shape, and color of these light shafts. In another work, we abstract from direct parameter changes and propose a goal-based design approach to manipulate the appearance of heterogeneous media such as clouds. We use inverse rendering to infer volume parameters from user paintings to achieve the desired look. The problem is expressed as an optimization procedure for which we show an efficient execution on the GPU. We show in several examples that these novel methods enable intuitive, expressive, and effective control over the stylization of volumetric scattering.

Kurzzusammenfassung

In dieser Dissertation werden neue Verfahren zur realistischen Darstellung und künstlerischen Manipulation von virtuellen Welten beschrieben.

Für interaktive Anwendungen, wie zum Beispiel Computerspiele, kann der Lichttransport in virtuellen Szenen nur vereinfacht simuliert werden, da Bilder in kurzer Zeit berechnet werden müssen. Zuerst wird ein effizienter Algorithmus zur Beleuchtung von Oberflächen unter natürlichem Licht beschrieben. Hierbei wurde eine existierende Methode erweitert, um richtungsabhängige Effekte in der Beleuchtung genauer darzustellen. In einem anderen Verfahren wird gezeigt, wie die Streuung von Licht in homogenen Medien, wie z.B. Nebel, effizient berechnet werden kann. Hierzu muss die Sichtbarkeit eines jeden Punktes entlang eines Kamerastrahls zur Lichtquelle getestet und letztendlich integriert werden. Das Problem kann zu einer Filteroperation transformiert werden, wodurch viele Kamerastrahlen parallel bearbeiten werden und somit der Effekt praktisch sowie asymptotisch schneller als bisher berechnet wird.

Im zweiten Teil der Dissertation werden neue Methoden zur künstlerischen Gestaltung der Streuung von Licht beschrieben. Ein Verfahren ermöglicht die Bearbeitung von Strahlenbüscheln, einer Lichterscheinung, die in der Natur durch Öffnungen zwischen Wolken an diesigen Tagen entsteht und häufig in Filmen und Computerspielen eingesetzt wird. Die neuen Werkzeuge steuern spezielle Renderingtechniken, wodurch Strahlenbüscheln effektiv in Erscheinung, Form und Farbe verändert werden können. In einer zweiten Methode wird anstatt direkter Parametermanipulation eine zielorientierte Gestaltung von heterogenen Volumen wie Wolken betrachtet. Hierbei werden mittels invertierten Rendering aus benutzerdefinierten Eingabebildern volumetrische Parameter optimiert, um ein gewünschtes Erscheinungsbild zu erreichen. Es wird gezeigt, wie die Optimierung mit der Hilfe der GPU sowie der Wiederverwendung von Berechnung des Lichttransports beschleunigt werden kann. Anhand mehrerer Beispiele wird gezeigt, dass mit den neuen Verfahren eine intuitive, expressive und effektive Kontrolle zur Gestaltung von volumetrischen Effekten möglich ist.

Summary

In this dissertation, we are concerned with scene appearance to create believable and appealing worlds. Visual design can be constrained by the rendering algorithm that computes the image as well as the complexity of scenes that can be handled. We address both aspects, the rendering and manipulation of complex worlds, in order to depict detailed virtual scenes.

The proposed rendering algorithms target real-time scenarios, important for previsualizations of visual effects, where artists require instant feedback when designing novel worlds. Another application are computer games, where the player enters a virtual world and interacts with it. Immersion is a key aspect and often achieved with the help of realistic graphics, which builds upon physically-based rendering. The real-time constraint, however, often limits the accuracy at which light-transport effects can be displayed. We propose two novel rendering techniques: screen-space bent normals and cones (Chapter 3) for efficient surface shading and a filter-based single-scattering method (Chapter 4) for the rendering of homogeneous participating media.

Volumes not only enable rendering scenes more realistically, but are often used to emphasize the atmosphere in the scene, to highlight certain objects, or for purely artistic reasons. The second focus of this thesis is the stylization and appearance modification for scenes with participating media. While today's scene complexity can be considered rather high, most tools to create such scenes focus on the manipulation of surfaces and solid objects. We propose several tools to change the appearance of volumetric scattering. While rendering often achieves realism using physically-based rules, this restriction is not essential for many applications such as games and movies. Our tools explicitly allow for non-photorealistic manipulation to enable expressiveness in the depiction of virtual scenes. In our first work (Chapter 6), we propose a set of manipulation methods to influence the rendering of single-scattering homogeneous media. We then continue with the manipulation of actual volume parameters (Chapter 7) of a heterogeneous medium and environmental lighting to match a desired appearance. By deriving actual volumetric parameters, we are able to respect physical constraints, if desired by the user.

In the following, we summarize the proposed methods individually.

Screen-space Bent Normals and Cones Classic rendering focuses on the depiction of surfaces, as most objects which we interact with in real life, are solids. But even when limiting light transport to surfaces, it is highly complex and, in consequence, it is common to employ approximations such as ambient occlusion. We propose bent normals and cones that extend the very popular effect of ambient occlusion, which can be approximated in screen space in real time. Our method also performs in real time, but better accounts for the directional dependency of lighting, the main compromise of ambient occlusion. It hereby allows for a more accurate depiction of the scene under environmental lighting, a common approach to achieve realistic and natural illumination. Ambient occlusion reaches high performance by decoupling occlusion from shading, although both factors are inherently linked. Our approach estimates the statical properties mean and variance of unoccluded directions, which are used for a more accurate combination of average occlusion and shading without occlusion.

We show that the properties can be computed as a byproduct of screen-space ambient occlusion, imposing only a negligible performance overhead. Our method is easy to integrate into existing graphics pipelines and provides lighting effects with higher quality compared to previous work.

Filter-based Single Scattering Volumetric scattering, caused by participating media, is not only a highly appealing effect, but also adds realism and gives spatial cues regarding the scene's layout. Scattering, however, is very challenging to render as it can happen virtually anywhere. Interesting effects such as light shafts still occur with strongly simplifying assumptions of homogeneous media and single scattering, but a naive ray-marching approach is often prohibitive for real-time applications. Previous work improved performance by employing acceleration structures to speed-up or skip visibility checks when possible. However, the performance of the techniques is often scene-dependent, an undesirable property as the complexity of scenes increases. We introduced a filter-based single-scattering method that turns the commonly used ray marching into an image-filter operation. The core of our method is a linearization of the visibility function to allow for the filtering of shadow maps, an idea that was originally proposed for surface shadows. While the original work used filtering to counter shadow-map aliasing by averaging visibility in a local neighborhood of a surface point, we use it to compute an average visibility for entire view rays. To do so, we require a rectified shadow map, which is computed in a special light space, where light rays and view rays are orthogonal. To compute the map, we propose a novel projective transformation to enable the use of rasterization on the GPU. Our approach reduces the overall complexity of the scattering computations and – as most of the work is performed in image space – its cost is mostly scene-independent. It also scales very well with screen resolution as the per-pixel costs are small.

Expressive Stylization of Single Scattering We begin the exploration of methods for scattering manipulations by proposing a set of stylization tools targeting homogeneous media. Volumetric light scattering is often used as an artistic tool to achieve a certain mood in the scene or emphasize certain objects. We propose a novel solution to help artists change the appearance of single-scattering effects. Our approach offers techniques based on occluder manipulation to remove or add apparent complexity to the scattering. Furthermore, we propose easily modifiable transfer functions as a tool to control the colors of scattering. Our solution is easy to use, compatible with standard rendering pipelines, and can be executed interactively to provide the artist with quick feedback.

Property and Lighting Manipulations for Volume Stylization The previously-presented method only addresses homogeneous media. However, real-world media such as clouds or smoke are heterogeneous. The artistic control of the volume's appearance is challenging as it is the result of a complex interplay of potentially millions of parameters defining the medium. So far, either physically-based simulation results are used, or volume modifications are only high-level, e.g., by changing parameters of noise functions. Intuitive and detailed artistic control is not possible. We propose a solution to stylize single-scattering heterogeneous participating media using inverse rendering, enabling a goal-based editing experience. The main participating-media parameters of emission, scattering, and extinction coefficients become amenable to artistic control. Our approach lets the user define the appearance of the medium by specifying target images. An optimization procedure finds medium parameters such that the user indications are matched. Via an analysis of the volumetric rendering equation, we also show how to link this problem to tomographic reconstruction, enabling us to use efficient optimization strategies.

Acknowledgments

I wish to thank my girlfriend, Corinna, who supported me throughout the PhD. She gave me strength and were patient with me at stressful times. I also wish to thank my parents who support me in all my goals and whenever they can. A thanks also goes to my brother and my grandparents who certainly had a big influence on me.

A very special thanks goes to Elmar, who guided, helped, and taught me so much. His extraordinary knowledge and skills still amaze me and proofed to be a great source for me. I thank Elmar a lot for his cheerfulness, always believing in me and my work, which greatly motivated me. I really appreciate that I got such an amazing mentor, which was more than just a supervisor for my PhD.

I also want to thank Ivo with whom I had many interesting discussions. Thanks to him I was looking at some mathematical problems in more detail, which I am very grateful for. But he also opened up the world of computational photography for me, a very interesting topic.

Although we didn't see each other as much as we should have seen as office mates, I really appreciate the discussions I had with Tobias. His creativity and critical thinking inspired me a lot.

I also want to thank Hans-Peter, who manages to create an excellent environment at the graphics group that allows PhD students to focus on their research. I got a lot of freedom during my PhD, which I did not take for granted, but which I highly appreciated. Thanks to Hans-Peter, I could visit a number of conferences, something that I see as the best reward for a PhD student.

I would also like to thank the great pool of intelligent and fun people at the computer graphics group at MPI. In particular I would like to thank Bernhard, Chuong, Oskar, Timothy, Krzysztof, Petr, Piotr, Alkhazur, Ilya, Michael, and Karol for many discussions and making academia a fun place to work at.

I also had two visits during my PhD, which I very much enjoyed and where I also learned a lot. It was great fun to meet and work with the people of Disney Research Zurich and Technical University of Delft. The visits were definitely too short.

Working hard with so intelligent people clearly shaped me and changed the way I approach and perform tasks.

Finally, I wish to thank Thomas and Timothy for proofreading parts of my thesis and the members of the PhD committee for their time and effort.

Contents

Abstract V

Kurzzusammenfassung VII

Summary IX

Acknowledgments XI

Table of Contents XV

1 Introduction 1

1.1 Topics of the Dissertation . . . 2
1.2 Original Contributions . . . 4
1.3 Structure of the Dissertation . . . 10

I Real-time Rendering for Natural Illumination and Volumetric Single Scattering 11

2 Background on Rendering and Light Transport 13

2.1 Assumptions on Light Behavior . . . 13
2.2 Determining Visibility . . . 14
2.2.1 Ray Tracing . . . 15
2.2.2 Rasterization . . . 15
2.2.3 Applications of Rasterization . . . 15
2.3 Interaction of Light with Surfaces . . . 18
2.3.1 The Rendering Equation . . . 19
2.3.2 Global Illumination . . . 19
2.3.3 Environmental Illumination . . . 21
2.3.4 Ambient Occlusion and Bent Normals . . . 22
2.4 Light Transport in Participating Media . . . 24
2.4.1 Assumptions on Participating Media and Their Properties . . . 24
2.4.2 The Radiative Transfer Equation . . . 27
2.4.3 Volume-Rendering Approaches . . . 28
2.4.4 Single Scattering for Homogeneous Media . . . 30

3 Bent Normals and Cones in Screen Space 33

3.1 Contributions . . . 34
3.2 Screen-space Ambient Occlusion . . . 34
3.3 Our Technique . . . 35

3.3.1 Bent Normals . . . 35
3.3.2 Bent Cones . . . 36
3.3.3 Preconvolved lighting . . . 36
3.3.4 Geometric Term . . . 37
3.4 Implementation . . . 38
3.5 Results . . . 40
3.6 Discussion and Conclusion . . . 43
3.6.1 Future Work . . . 44

4 Real-time Single-Scattering Rendering of Homogeneous Media 45
4.1 Contributions . . . 46
4.2 Background . . . 46
4.2.1 Scattering Model . . . 46
4.2.2 Rectified Shadow Map and Epipolar Geometry . . . 47
4.3 Single-Scattering Method . . . 48
4.3.1 Average Visibility . . . 48
4.3.2 Extensions . . . 50
4.3.3 Implementation . . . 53
4.4 Rectification . . . 57
4.4.1 Epipolar Geometry . . . 57
4.4.2 Our Projective Rectification . . . 58
4.5 Results . . . 61
4.6 Conclusion . . . 64
4.6.1 Future Work . . . 65

II Stylization and Appearance Manipulation of Volumetric Scattering 67

5 Related Work on Appearance Manipulation and Reconstruction 69
5.1 Light Transport . . . 69
5.2 Artistic Manipulation of Volumetric Scattering . . . 71
5.3 Reconstruction of Volumetric Effects . . . 72
5.4 Fabrication . . . 72

6 Stylization of Volumetric Scattering 73
6.1 Contributions . . . 74
6.2 Stylized Single Scattering . . . 74
6.2.1 Occluder Modification . . . 75
6.2.2 Transfer Functions . . . 79
6.3 Results and Discussion . . . 80
6.4 Conclusion . . . 85
6.4.1 Future Work . . . 85

7 Property and Lighting Manipulations for Volume Stylization 87
7.1 Contributions . . . 88
7.2 Scattering Model . . . 88

7.3 Volume & Light Reconstruction . . . 89
7.3.1 Volume Reconstruction . . . 89
7.3.2 Property Reconstruction . . . 90
7.3.3 Discussion of Volume Reconstruction . . . 92
7.3.4 Light Reconstruction . . . 93
7.4 Implementation . . . 93
7.4.1 Volume Reconstruction . . . 94
7.4.2 Optimizations and Extensions . . . 94
7.4.3 Light Reconstruction . . . 97
7.5 Results and Discussion . . . 98
7.5.1 Animation . . . 105
7.6 Conclusion . . . 106
7.6.1 Future Work . . . 106

8 Conclusion **107**
8.1 Future Work . . . 109

Bibliography **111**

1
Introduction



Figure 1.1: Visual richness in today's movies ranging from photorealistic visual effects to artistically-driven animation films. Both exemplary types feature highly detailed worlds, allowing for immersion with believable (non-realistic) worlds. Image credits and copyrights: Warner Bros. (Man of Steel), Warner Bros. (Harry Potter and the Deathly Hallows), Disney (Frozen), Disney (Big Hero 6).

The vast increase of model complexity in computer graphics leads to highly detailed scenes. Coupled with realistic rendering solutions, almost photorealistic images can be produced. But we can even go further and enhance photo-realism to emphasize important aspects of a scene and artistically design the look of scene elements. The appearance of a scene – solid objects or volumes – is inherently influenced by shape, material properties, as well as lighting, which plays a critical role. Complex appearance models, natural illumination, geometrically-detailed objects, and realistic camera models enable the visual richness that we are used to by today's high-quality renderings. Unfortunately, creating detailed worlds as shown in Figure 1.1 is extremely time consuming and requires many skilled artists to perform laborious work as it is very common for the creation of AAA-games and movie blockbusters. What artists and common users can build is ultimately limited by the tools they work with.

Existing artistic tools to manipulate virtual worlds focus on surface modeling: the creation and manipulation of geometry, animation, and material design in conjunction with illumination. The control over volumetric scattering caused by participating media is mostly ignored by these tools. Physically-based simulations of volumetric elements such as smoke and fluids yield highly detailed results, but the appearance of those volumes cannot be predicted. Control over volumes is important as it is an element that is used increasingly often as a supporting element in feature movies as well as games. Volumes not only enable rendering scenes more realistically, but are often used to emphasize the atmosphere in the scene, to highlight certain objects, or for purely artistic reasons. Unfortunately, creating and manipulating participating media is tremendously more difficult than editing surfaces as they have a continuous volumetric extent. A medium is often represented by high-dimensional models, commonly regular grids with millions of parameters (voxels). The difficulty in controlling its appearance directly lies in the image formation model: many voxels influence the final rendering in a non-trivial manner. While this is also the case for indirect illumination on surfaces, the effect is much more severe for volumes as even the traversal of a single straight ray through the scene can intersect many voxels. Editing operations are therefore challenging as the outcome of a parameter change is difficult to predict, especially due to the sheer number of parameters that need to be controlled.

The goal of this dissertation is to develop efficient and effective methods for users to create and display scene content as well as changing its appearance. We limit ourselves to methods that relate to the effect of volumetric scattering. To meet the goal of effectivity we will look at different levels of complexity of scene content and its manipulation. We will investigate artistic tools that can be applied to simplified volumetric scattering such as that in homogeneous media, i.e., media with constant density as is commonly used for interactive applications. We will also explore methods that allow for manipulating large, heterogeneous volume sets. Finally, we will go beyond volume editing and manipulate the scene illumination such that the entire scene exhibits a desired mood. To ensure efficiency, we focus on interactive rendering for all our techniques. Rendering itself poses a limiting factor in the creation of detailed worlds as it constrains which light-transport effects are supported, the grade of complexity of the scene such that render time stays reasonable, and – in conjunction with scene manipulations – how quickly users are provided with feedback. We will investigate novel algorithms to advance the state of the art in real-time rendering of volumetric scattering as well as surface shading under natural illumination. In this context rendering is more than a method to turn the virtual scene description into images, it is also an essential part of the scene manipulation process. This can be as simple as providing the user with immediate feedback on editing actions. But we will also expose more control over the rendering as a way to manipulate the appearance of the scene. Lastly, we use rendering in a reconstruction method to derive volume parameters from user paintings.

1.1 Topics of the Dissertation

In the following, we discuss the aforementioned topics in slightly more detail, before presenting on overview of our contributions in the next section.

Interactive Rendering The main task of a rendering algorithm is the simulation of light transport from the light emitters towards a virtual camera sensor. Although this process is well-defined mathematically, the computational problem cannot be considered as solved. Even for offline rendering, as used for movies that do not have strict time constraints, scene complexity is easily increased such that the movie is not renderable in a reasonable time. Volumetric scattering as mentioned before is such a phenomenon that is still very difficult to



Figure 1.2: Visual richness achieved with interactive rendering methods. The constraint of real-time rendering does not allow for the same degree of details as offline rendering (cf. Figure 1.1). Yet, a clever usage of the available scene elements and methods such as lighting, approximated global illumination, and volumetric scattering also allows for the creation of plausible, believable, and appealing worlds. In both cases volumetric scattering is simplified: while the rendering on the left features fake heterogeneity by a non-uniform scaling of the scattering contribution, the rendering on the right only uses homogeneous media. Both renderings only use a single-scattering model. The image on the left is courtesy of Ubisoft.

render efficiently as light can scatter virtually anywhere in space. However, the rendering problem is even more challenging for interactive applications such as computer games or scene editing software. The computational demands of photorealistic rendering are still far too high for interactive applications. To deliver high-quality results within fixed time budgets, a composition of specialized techniques is often applied. These techniques typically focus on the display of specific scene elements or approximate complex light-transport phenomena in a simplified, potentially non-physically-based manner (see Figure 1.2). A very basic, but essential problem is the determination of visibility and its integration over solid angles and rays for the computation of light reflection and scattering. We will investigate advances to the integration computations related to the specific rendering technique of *ambient occlusion* as well as single scattering in homogeneous media. As such, improving rendering algorithms increases visual richness, which in turn helps us to better design scenes and convey moods. However, it also supports the modeling process as artists require immediate visual feedback on the performed actions to successfully manipulate the scene. This feedback enables them to explore the design space, or quickly react to unwanted results. Effectively, feedback is supplied by re-rendering the scene, object, or element that is modified.

Rendering Control Rendering may also constrain the modeling of virtual worlds, which is especially the case for interactive applications that use specialized rendering techniques. As such, artists cannot freely design a world, but need to make use of the capabilities and techniques offered by the rendering engine wisely. The challenge for novel design tools is that they need to go hand in hand with the final rendering pipeline as the content is displayed by the latter. We can observe this link in today's game engines, such as the CryEngine, Unreal Engine, or Unity. They provide integrated design and editing tools besides their rendering capabilities. However, the specialized rendering techniques also offer an indirect way of scene manipulation as not necessarily the scene itself but the rendering process can be altered to provide artistic control. It needs to be explored which parts of the pipeline or parameters should be modified. We will investigate this for the case of volumetric scattering.

Manipulating the rendering process with artistic tools potentially contradicts physically-based rendering, which is used to achieve photorealistic results. However, violations may be justified by two reasons. As indicated

before, real-time rendering is not yet able to accurately simulate light transport and requires artistic help to generate appealing results. More importantly, physical correctness is not a strict requirement for the visual design used in games or movies. Ideally, we are interested in tools that allow us to go beyond physical constraints and to guide the attention of the observer to important parts of the scene, support a desired atmosphere, convey a mood, while ensuring plausible results.

Inverse Rendering Finally, we can use rendering itself as a method to manipulate the scene. The idea of inverse rendering is that the user or artist directly indicates what they want to see, e.g., by painting strokes or an entire image. The system needs to reconstruct scene parameters such that the rendering matches the user's input. Such an approach is useful as it gives intuitive control to the user, even for highly complex parameter sets such as those of participating media. This is desired as the influence of parameters on the overall scene appearance is difficult to predict due to multiple reasons. First, as indicated before, participating media are often represented by voxel grids with potentially millions of individual elements, whereas many of them affect a pixel's color. Second, non-linear models (such as transmittance attenuation in participating media) lead to a non-intuitive impact of parameter edits on the final rendering. These factors make it challenging to control the parameters such as voxels manually. With inverse rendering, the complex interaction of parameters is abstracted away and the user is left with a goal-based scene editing experience. The approach shifts the work to the algorithm, which usually implements the parameter reconstruction as an optimization procedure. However, the reconstruction problem is usually ill-posed. As such, many parameter configurations may map to a similar rendering and finding the right is far from trivial. It is therefore required to add additional constraints to the optimization, such as the number of parameters to estimate. The speed of convergence depends on the optimization algorithm, but also on the implementation of its individual steps such as the evaluation routine to test the current set of parameters. The latter closes the loop to efficient rendering techniques as evaluation means rendering an image using the modified scene parameters.

As a side effect, inverse rendering also tackles another essential problem of volume modification: the way how a user should interact with the volume. Volumes have a 3D extent, but most of our input devices to the computer are limited to 2D surfaces such as the mouse, fingers, or a pen. Inverse rendering solves this issue, as the user indicates the desired appearance equally to the output rendering in the 2D image space. We will see how inverse rendering can be applied to derive volume parameters at a per-voxel level. We will express elemental operations of the optimization as rendering problems, allowing us to leverage efficient algorithms and the graphics hardware.

1.2 Original Contributions

This thesis builds upon work previously published at peer-reviewed conferences, journals as well as book chapters. In the following we will highlight the contributions that we made in those works and which will then be discussed in the following chapters. Our contributions target the problems listed before: displaying virtual worlds efficiently, going beyond physically-based rendering to support the visual design process, and, finally, handling and creating complex worlds.

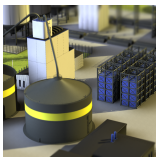
Real-time Surface Rendering for Environmental Illumination (Chapter 3)

Natural illumination is a key ingredient to display virtual worlds realistically. It is often achieved using environment maps that encode a position-independent irradiance value for all directions. As such, the illumination

emitter is assumed to be far away from the scene content. This model works well for outdoor scenes, where most of the light comes from the sky. In our approach we extend the well-known method of ambient occlusion to handle natural illumination efficiently. Ambient occlusion decouples shading from occlusion to gain efficiency. This decoupling results in an average occlusion caused by nearby objects that modulates the surface shading. It is a crude, but very efficient approximation of global illumination, which works well for cloudy skies [Landis, 2002]. The main compromise is the lack of directional information in the illumination. We look at statistical properties of occlusion to incorporate directional information efficiently. Bent normals [Landis, 2002], the mean of unoccluded directions, were already proposed as an amelioration that addresses this issue for offline rendering. In our work, we describe how to compute bent normals as a cheap by-product of real-time methods of screen-space ambient occlusion. In addition, we also show that it is possible to compute the variance of unoccluded directions, which we combine with the mean direction to bent cones. Hence, they have a direction and an angular extent, which can be used to query sub-regions of unoccluded environmental illumination. These extensions combine the speed and simplicity of screen-space ambient occlusion with physically more plausible lighting. In short, our contributions are:

- Extending existing screen-space ambient occlusion techniques to compute bent normals;
- Computing bent cones that combine mean and variance of unoccluded directions.

The proposed method was published in:



Bent Normals and Cones in Screen Space
Oliver Klehm, Tobias Ritschel, Elmar Eisemann, and Hans-Peter Seidel
16th International Workshop on Vision, Modeling and Visualization (VMV)
2011, pp. 177–182, Berlin, Germany, DOI: 10.2312/PE/VMV/VMV11/177-182



Screen-space Bent Cones: A Practical Approach
Oliver Klehm, Tobias Ritschel, Elmar Eisemann, and Hans-Peter Seidel
GPU Pro 3, editor: W. Engel
2012, part 3, ch. 2, pp. 191–207, A K Peter/CRC Press, DOI: 10.1201/b11642-15

Real-time Single-Scattering Evaluation for Homogeneous Media (Chapter 4)

Our modeling methods will focus on editing volumetric light-scattering effects. Scattering is a complex phenomenon that is difficult to simulate in real time as light can be scattered towards the camera from anywhere in space. To allow for interactive frame rates, the participating medium is assumed to be homogeneous. Combining this assumption with a single-scattering model, the main computational work reduces to visibility checks at scatter points on camera rays towards the light source. We replace the usually-employed ray-marching procedure with an efficient ray-independent filtering process that reduces the complexity of the scattering computation. To do so, we linearize the visibility function as proposed by Annen et al. [2007] and compute visibility for entire epipolar slices, i.e., 1D lines in screen space. The filtering is an efficient one-dimensional operation that assumes a rectified shadow map as input for which we propose a novel rectification scheme. Our method is fast and almost screen resolution independent, while producing near-reference results. In summary, our contributions are:

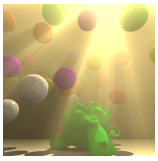
- A single-scattering algorithm with reduced complexity based on prefiltering of a rectified shadow map;
- A matrix transform for single-step shadow-map rectification;

- Efficient strategies for a GPU execution.

The proposed method was published in:



Prefiltered Single Scattering
Oliver Klehm, Hans-Peter Seidel, and Elmar Eisemann
ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)
2014, pp. 71–78, San Francisco, CA, USA, DOI: 10.1145/2556700.2556704



Filter-based Real-time Single Scattering using Rectified Shadow Maps
Oliver Klehm, Hans-Peter Seidel, and Elmar Eisemann
Journal of Computer Graphics Techniques (JCGT)
2014, 3(3), pp. 7–34

Expressive Stylization of Volumetric Scattering (Chapter 6)

We begin the exploration of methods for scattering manipulations by proposing a set of stylization tools targeting homogeneous media. Volumetric light scattering is often used as an artistic tool to achieve a certain mood in the scene or emphasize certain objects. Thus far, however, little research has focused on artistically influencing the scattering process, which will be reviewed in Chapter 2. We propose a novel solution to help artists change the appearance of single-scattering effects. Specifically, we target the stylization of crepuscular rays (also called light shafts), which are caused when light rays illuminate the medium, while their neighboring rays are blocked. Our approach offers techniques based on occluder manipulation to remove or add apparent complexity to light shafts. Furthermore, we propose easily modifiable transfer functions as a tool to control the colors of light shafts. Our solution is easy to use, compatible with standard rendering pipelines, and can be executed interactively to provide the artist with quick feedback. Specifically, our work makes the following contributions:

- Addition, removal, and enhancement of light shafts using image-based occluder manipulation;
- Color modifications of light shafts via user-editable transfer functions based on view-ray properties;
- Animation support by dynamic occluder manipulation, and key-framed transfer functions.

The proposed method was published in:



Stylized Scattering via Transfer Functions and Occluder Manipulation
Oliver Klehm, Timothy R. Kol, Hans-Peter Seidel, and Elmar Eisemann
Graphics Interfaces (GI)
2015, pp. 115–121, Halifax, Canada

Appearance Modifications of Heterogeneous Media and Environmental Illumination Using a Painting Metaphor (Chapter 7)

The previously presented method only addresses homogeneous media. However, real-world media such as clouds or smoke are heterogeneous. The artistic control of the volume's appearance is challenging as it is the result of a complex interplay of potentially millions of parameters defining the medium. So far, either physically-based simulation results are used, or volume modifications are only high-level, e.g., by changing parameters

of noise functions. Intuitive artistic control is not possible. We propose a solution to stylize single-scattering heterogeneous participating media using inverse rendering, enabling a goal-based editing experience. The main participating media parameters of emission, scattering, and extinction coefficients become amenable to artistic control. Our approach lets the user define the appearance of the medium by specifying target images. An optimization procedure finds medium parameters such that the user paintings are matched. Via an analysis of the volumetric rendering equation, we can show how to link this problem to tomographic reconstruction, enabling us to use efficient optimization strategies.

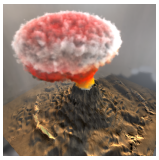
A consistent stylization of the entire scene is achieved by also manipulating the illumination. This illumination manipulation gives effective control over the perceived atmosphere. We extend the medium stylization by optimizing environmental illumination parameters instead of volume parameters. The user interaction is the same: an artist specifies constraints by painting the heterogeneous medium from desired points of view. Finally, we show how to combine volume and lighting manipulation. Precisely, our contributions are:

- An inverse rendering approach to intuitively stylize heterogeneous participating media;
- An inverse rendering approach to stylize environmental illumination for scenes with heterogeneous participating media;
- A fast linear optimization of millions of parameters assuming a single-scattering model;
- An efficient implementation of the optimization procedure using light-transport caching to keep execution time and memory cost practical.

The proposed method was published in:



Volume Stylizer: Tomography-based Volume Painting
Oliver Klehm, Ivo Ihrke, Hans-Peter Seidel, and Elmar Eisemann
ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)
2013, pp. 161–168, Orlando, FL, USA, DOI: 10.1145/2448196.2448222



Property and Lighting Manipulations for Static Volume Stylization Using a Painting Metaphor
Oliver Klehm, Ivo Ihrke, Hans-Peter Seidel, and Elmar Eisemann
IEEE Transactions on Visualization and Computer Graphics (TVCG)
2014, 20(7), pp. 983–995, DOI: 10.1109/TVCG.2014.13

Virtual Reproduction of Real-World Objects

The topics described so far focus on appearance manipulation and efficient rendering of virtual sceneries to create appealing and believable virtual worlds. An apparently straightforward way to reach this goal of plausible worlds is the digital cloning of real-world objects and elements. While we also looked at this digitalization process, this thesis only focuses on the aforementioned parts of purely virtual worlds with participating media. We will only give an outline on the research performed on the virtual reproduction of the real world.

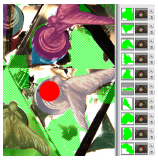
Using references from the real world is an option to create believable and realistically-looking scene elements that are difficult to achieve otherwise. For example, manually recreating the 3D face including its appearance of an actor is extremely challenging as humans are very sensitive to the slightest inconsistency, but digitalization of real-world objects offers a solution. Nonetheless, the process is complicated. Three main challenges can be identified: the data acquisition, model reconstruction, and specialized rendering. The details of data acquisition strongly depend on the final purpose of the reproduction, yet, a common way is to use optical sensing with

cameras, e.g., plain photographs of an actor's face. The data can then be used to reconstruct high-level elements such as a geometric representation of the object, combined with an estimation of the object's material. For the example of virtual actors, one reconstructs the geometry, combined with its normals as well as surface reflections and sub-surfaces scattering characteristics. Once such a parametric model is obtained, it can be further processed, e.g., by applying artistic modifications. Eventually, the obtained model is used in a standard rendering pipeline to generate novel images such as the actor in a virtual environment. Often, only parts of the scene can be reconstructed, e.g., the geometry, but not the material properties, which requires tailored rendering techniques. It is possible to use the original photographic data and treat parts of the scene as a black box. The resulting process is generally called image-based rendering and samples the input images according to the properties of the capture setup and potentially reconstructed information (such as scene geometry). In the following, we will outline instances of: data acquisition, model reconstruction, as well as image-based rendering.

Kaleidoscopic Imaging and Geometry Reconstruction Mirror systems have recently emerged as an alternative low-cost multi-view imaging solution. The use of these systems critically depends on the ability to compute the background of a multiple-mirrored object. The images taken in such systems show a fractured, patterned view, making edge-guided segmentation difficult. Standard segmentation techniques fail on such input due to global illumination and light attenuation caused by the mirrors. We propose a system that assists a user performing the segmentation manually. We provide convenient tools that enable an interactive segmentation of kaleidoscopic images containing three-dimensional objects. Throughout the process, we preview the user's stroke actions in 2D as well as in 3D in conjunction with the object reconstructed based on the current segmentation. The system employs volumetric ray-marching to reconstruct a voxel-based representation of the object based on virtual ray tracing through the mirror setup. We obtain a visual hull of the object and estimate surface normals. Finally, to display the reconstructed object, we use the captured data in an image-based rendering setup to maintain complex view-dependent shading effects. To achieve interactivity, we employ the GPU in all stages of the application, such as 2D/3D rendering as well as segmentation. In summary:

- We present an application that allows for manual foreground/background segmentation and provides feedback to the user in the 2D kaleidoscope domain as well as the 3D domain of the visual hull.
- We describe an on-line technique for estimating the visual hull of an object inside a kaleidoscope system of arbitrarily positioned planar mirrors.
- We explore suitable interaction schemes and visualization techniques that guide the user in rapidly creating an accurate segmentation.
- We handle the heavy computations of drawing, rendering, visual hull derivation, and labeling by exploiting the GPU.

The proposed method was published in:



Interactive Geometry-Aware Segmentation for the Decomposition of Kaleidoscopic Images
Oliver Klehm, Ilya Reshetouski, Hans-Peter Seidel, Elmar Eisemann, and Ivo Ihrke
17th International Workshop on Vision, Modeling and Visualization (VMV)
2012, pp. 9–14, Magdeburg, Germany, DOI: 10.2312/PE/VMV/VMV12/009-014

Light-Field Capture and Rendering The full spectrum of acquisition, reconstruction, and image-based rendering needs to be considered for light-field imaging. Light fields are four-dimensional and sample the scene

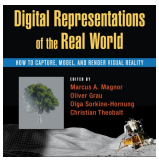
not just spatially (as usual images), but also measure directional information of surface points. This directional sampling is achieved by imaging the scene from different points of view. The most prominent application is virtual refocusing, which works by resampling the light field and requires only very small changes in the input viewpoints. In an overview article, we cover specialized camera setups and their principles to capture 4D light fields. We also show how refocusing is performed as well as other applications such as a virtual change in the camera's field of view.

In another work, we propose a removable add-on for standard cameras that enables aperture sub-sampling and, hence, light-field imaging. Our design is based on an optical copying mechanism using mirrors to create multiple virtual cameras conceptually identical to the multi-view kaleidoscopic setup. With the help of the mirrors, the virtual cameras, each with a slightly different point of view, are spatially multiplexed on the physical sensor. A minor modification of the design also allows for plenoptic imaging, i.e., the capture of other light information such as its polarization state. To this extent, we use a diffuser to get several identical copies of an image instead of multiple viewpoints. The copies can then be optically filtered to get samples of the plenoptic information of interest. As the filters in our design are exchangeable, a reconfiguration for different imaging purposes is possible. We show in a prototype setup that light-field, high dynamic range, multispectral, and polarization imaging can be achieved with our design.

Our design features several advantages. In particular, our contributions are an optical design for a removable camera add-on for snapshot plenoptic imaging, that:

- enables a light-field camera design which allows for the control of depth-of-field vs. light throughput, and a spatial/angular resolution trade-off that facilitates high spatial resolution (full HD) recording and refocusing,
- features identical sub-images without parallax for filter-based imaging by employing a diffuser architecture, enabling scene-independent, precalibrated acquisition,
- allows for an increased photographic control of plenoptic imagery as compared to existing approaches, by avoiding the use of sub-apertures for HDR, multispectral, and polarization imaging.

The overview article and the proposed method were published in:



Plenoptic Cameras

Bastian Goldlücke, Oliver Klehm, Sven Wanner, and Elmar Eisemann

Digital Representation of the Real World: How to Capture, Model, and Render Visual Reality, editors: M. Magnor, C. Theobalt, O. Grau, O. Sorkine-Hornung

2015, ch. 5, pp. 65–78, A K Peter/CRC Press, DOI: 10.1201/b18154-7



A Reconfigurable Camera Add-On for High Dynamic Range, Multispectral, Polarization, and Light-Field Imaging

Alkhazur Manakov, John F. Restrepo, Oliver Klehm, Ramon Hegedus, Elmar Eisemann, Hans-Peter Seidel, and Ivo Ihrke

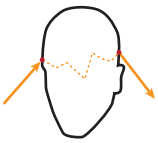
ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)

2013, 32(4), pp. 47:1–47:14, Anaheim, CA, USA, DOI: 10.1145/2461912.2461937

Facial Appearance Capture As we already indicated, a very important application of digital cloning is the capture of actors and, in particular, their faces. Faces are extremely detailed and probably the most important visual cue for humans to recognize each other. Placing an actor into new (virtual) environments is often needed for movie or game production and requires a virtual model of the actor. While a lot of previous work focused

on the estimation of geometry and its change during performances, we investigated the state of the art in facial appearance capture. We found that most approaches can be categorized into image-based and parametric methods with a current trend towards the latter. The main future challenges will be to improve the underlying mathematical models as most are not yet fully accurate, and fitting their parameters from observations is highly complex.

The report was published in:



Recent Advances in Facial Appearance Capture

Oliver Klehm, Fabrice Rousselle, Marios Papas, Derek Bradley, Christophe Hery, Bernd Bickel, Wojciech Jarosz, and Thabo Beeler

Computer Graphics Forum (Proc. of Eurographics)

2015, 32(2), pp. 709–733, Zurich, Switzerland, DOI: 10.1111/cgf.12594

1.3 Structure of the Dissertation

This dissertation is divided into two parts and a total of eight chapters. In Part I we will begin with a background on rendering and light transport (Chapter 2), which introduces the mathematical foundation for the proposed rendering techniques as well as the editing methods. We also review related work for physically-based image synthesis, in particular interactive rendering of surface shading under natural illumination as well as volumetric single scattering of homogeneous media before detailing our contributions to both of these areas. We then show in Chapter 3 how the screen-space ambient occlusion technique can be extended to better account for natural illumination with our screen-space bent cones. We then move on to filter-based single scattering in Chapter 4, where we present how the computational complexity of scattering can be reduced. This technique consistently achieves the performance that is needed to build interactive editing techniques.

We continue in Part II with novel stylization and manipulation techniques that focus on the effect of volumetric scattering. We begin with reviewing related work of stylization, scene editing, and inverse rendering in Chapter 5. We then introduce our novel artistic tools in Chapter 6 that can be easily integrated into common rendering pipelines for real-time volumetric scattering such as – but not limited to – the proposed one in Chapter 4. We continue with the approach of inverse rendering in Chapter 7 to allow for intuitive volume manipulations of heterogeneous media with their complex parameter interaction. There, we also show how this can be extended to derive natural illumination such that the media and the rest of the scene appear in a desired atmosphere. We conclude this dissertation in Chapter 8.

Part I

Real-time Rendering for Natural Illumination and Volumetric Single Scattering

2

Background on Rendering and Light Transport



Figure 2.1: Examples of physically-based rendering used in movies and games. Image credits and copyrights: Warner Bros. (Godzilla), Bioware (Dragon Age: Inquisition).

Photorealistic or plausible image synthesis such as shown in Figure 2.1 is often achieved using so-called physically-based rendering. As the name suggests, the physical behavior of light is mimicked to compute how light that is emitted by light sources scatters in the scene until it reaches the virtual sensor of the camera.

In the following, we will first cover the basics of the nature of light as well as how the basic principle of visibility is resolved. We then continue with the general mathematical formulations for physically-based rendering. We begin the discussion with the effect of light interaction with surfaces (Section 2.3) and focus on the problem of environmental illumination as a form of natural illumination. We then continue with participating media in Section 2.4 and the background for efficient single-scatter rendering. Hereby, we will also give an overview over techniques that have been proposed in the computer-graphics literature. Please note, that we will not discuss the fundamentals of light transport in detail and instead refer the interested reader to excellent previous work such as that of Veach [1997] for the interaction of light with surfaces and Jarosz [2008] for light transport in participating media. We list our symbols in Table 2.1.

2.1 Assumptions on Light Behavior

Computer graphics typically relies on the relatively simple model of ray optics, which assumes that light travels in straight lines at infinite speed. Physics has a more advanced understanding of light with models that extend ray optics: wave, electromagnetic, or quantum optics. Consequently, complicated effects such as diffraction

Symbol	Description
$\mathbf{x}, \mathbf{x}_\mathrm{s}$	Position, point on a surface
$\vec{\mathbf{n}}$	Surface normal at $\mathbf{x}$, i.e., a normalized vector ($\|\vec{\mathbf{n}}\| = 1$)
$\vec{\omega}, \vec{\omega}_\mathrm{i}, \vec{\omega}_\mathrm{o}$	Normalized direction: incoming and outgoing from a point
$\Omega, \Omega^+, \Omega^{4\pi}$	Space of directions: upper hemisphere (with respect to $\vec{\mathbf{n}}$), full sphere
$\langle \vec{\omega} \cdot \vec{\omega}' \rangle$	Dot product between $\vec{\omega}, \vec{\omega}'$; cosine of angle between unit vectors
$L(\mathbf{x}, \vec{\omega})$	Radiance at $\mathbf{x}$ in direction $\vec{\omega}$
$L_\mathrm{i}, L_\mathrm{o}, L_\mathrm{e}, L_\mathrm{r}$	Incoming, outgoing, emitted, and reflected radiance
$L_\mathrm{env}(\vec{\omega})$	Spatially-constant environmental radiance
$f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}')$	Bidirectional distribution function (BRDF) at $\mathbf{x}$ for directions $\vec{\omega}, \vec{\omega}'$
$V(\mathbf{x}, \vec{\omega})$	Visibility at $\mathbf{x}$ in direction $\vec{\omega}$ (to infinity or a predefined range)
$V(\mathbf{x})$	Visibility at $\mathbf{x}$ (with respect to a predefined light source)
$z_\mathbf{x}$	The z-component (i.e., depth) of $\mathbf{x}$ in light space; we omit the transformation for clarity
$\mathcal{V}_{z_i}(z_\mathbf{x})$	Visibility of $\mathbf{x}$ using a shadow map; z_i is the depth at texel i, implicitly given by $\mathbf{x}$
Additional symbols for light transport in participating media	
$\sigma_\mathrm{s}(\mathbf{x})$	Scattering coefficient at $\mathbf{x}$
$\sigma_\mathrm{a}(\mathbf{x})$	Absorption coefficient at $\mathbf{x}$
$\sigma_\mathrm{t}(\mathbf{x})$	Extinction coefficient at $\mathbf{x}$ ($\sigma_\mathrm{t} = \sigma_\mathrm{s} + \sigma_\mathrm{a}$)
$\rho(\mathbf{x})$	Single-scattering albedo ($\rho = \sigma_\mathrm{s}/\sigma_\mathrm{t}$)
$\tau(\mathbf{x}, \mathbf{x}')$	Optical thickness between $\mathbf{x}, \mathbf{x}'$ ($\tau(\mathbf{x}, \mathbf{x}') = \int_\mathbf{x}^{\mathbf{x}'} \sigma_\mathrm{t}(\mathbf{x}_t)\, \mathrm{d}t$)
$T_\mathrm{r}(\mathbf{x}, \mathbf{x}')$	Transmittance between $\mathbf{x}, \mathbf{x}'$ ($T_\mathrm{r}(\mathbf{x}, \mathbf{x}') = e^{-\tau(\mathbf{x}, \mathbf{x}')}$) – likelihood that a photon travels from $\mathbf{x}$ to $\mathbf{x}'$ without interaction with the medium
$T_\mathrm{r}(\mathbf{x}, \vec{\omega})$	Transmittance from $\mathbf{x}$ to infinity into direction $\vec{\omega}$
$p(\mathbf{x}, \vec{\omega}, \vec{\omega}')$	Phase function at $\mathbf{x}$ for directions $\vec{\omega}, \vec{\omega}'$
$L_\mathrm{m}, L_\mathrm{emit}, L_\mathrm{scat}$	Radiance due to emission and in-scattering in the medium ($L_\mathrm{m} = L_\mathrm{emit} + L_\mathrm{scat}$)

Table 2.1: Symbols used for light transport

and interference (wave optics) are ignored. Nonetheless, the essential effects of emission, reflectance, and transmittance that dominate light transport can be simulated.

An important consequence of the ray-optics model is the linearity of light transport (interference as part of wave optics would result in non-linear behavior). In practice, the contribution of individual light sources can be computed separately and the results are simply summed. Moreover, one can essentially treat any lighting effect separately and derive special models for solving each of them. A very common and important application is the handling of light paths of different length (with length being defined by the number of bounces). Interactive rendering usually focuses on direct illumination (one bounce) and only optionally paths of higher order.

2.2 Determining Visibility

As light travels in straight lines, the most basic problem that a light-transport algorithm needs to solve is that of visibility, which comes in two similar forms. First, one needs to determine where a light ray will intersect scene geometry. Second, one needs to compute if two points in space *see* each other or whether they are occluded by

some scene geometry.

2.2.1 Ray Tracing

Ray tracing (also called ray casting) [Appel, 1968; Whitted, 1980] is the direct implementation of the model of ray optics, and analytically computes the intersection of a ray with some simple geometric primitive, most commonly triangles. Visibility between points is easily solved by casting a ray from one point into the direction of the other and checking if the resulting first intersection is at the target position. Ray tracing poses no restrictions on the rays, which are cast into the scene and a light-transport algorithm can query visibility between any points. However, intersections tests are done against simple geometric primitives, effectively requiring many tests as today's models are highly detailed. Ray tracing builds acceleration structures over the primitives to speed up the process. The more serious performance problem is caused by the sheer number of rays that need to be cast, e.g., simply casting a single ray for each pixel of a full HD image requires at least two million individual intersection tests (assuming only a single test per ray).

2.2.2 Rasterization

Rasterization addresses this issue of computing the first visible surface for many rays by leveraging the inherent grid structure of images. Many rays are highly coherent, in particular the view rays of projective cameras (such as a pinhole camera model) that undergo the same projection. Rasterization inverts the ray casting operation and instead projects the scene's geometry onto the image plane. This effectively solves visibility for many lines of sight in parallel. The subsequent pixel-coverage tests, i.e. the actual *rasterization* of the geometric primitives onto pixels, are hardware-accelerated by GPUs. To compute the closest geometry for each pixel, z-buffering [Straßer, 1974; Catmull, 1974] is used, which stores the distance to the closest projected geometry and tests the depth of newly projected geometry against the stored value. As this process is highly optimized with today's GPUs, rasterization is the preferred method for interactive applications. The downside of rasterization is the assumption that a single transformation is used for many rays. This global transformation scheme directly limits the rendering algorithm as visibility between arbitrary points cannot be queried efficiently.

2.2.3 Applications of Rasterization

The purpose of rasterization in combination with z-buffering is the determination of the first visible surface. In the following, we present the two most common scenarios where rasterization is used in a modern interactive rendering engine: deferred shading [Deering et al., 1988] and shadow mapping [Williams, 1978]. Our techniques will be based on both of these methods.

Deferred Shading We already mentioned the main application of rasterization: determining visibility for a projective camera. The result of this process is a camera depth buffer (or z-buffer), which is used to test if a fragment should be kept or not. However, to perform lighting calculation also other information such as the surface normal are required, which are only available during the processing of the geometry. Hence, classical forward rendering performs shading (lighting calculations) as soon as the current fragment *survives* the depth comparison against the z-buffer. The disadvantage is that there is no ordering in the projection of the geometry, i.e., geometry that gets processed later may overwrite previous results. While this is intended for the z-buffer, the shading operations can become expensive and unnecessary calculations should be avoided.

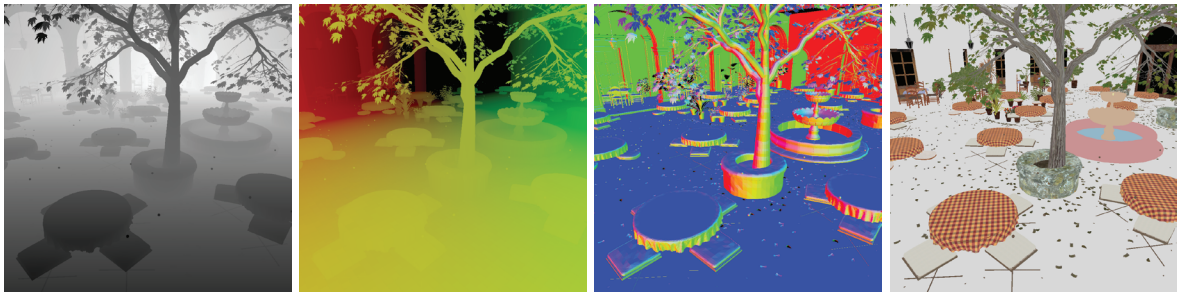


Figure 2.2: Example G-buffer containing (left to right): linear depth, world-space position, surface normal, diffuse color.

Deferred shading [Deering et al., 1988] was proposed as an alternative and performs shading computations after solving visibility. To this extent, all required information for the subsequent shading (i.e., surface normal, texture coordinates, potentially material parameters) are simply written to buffers, the so-called G-buffer (geometric buffer, see Figure 2.2) [Saito and Takahashi, 1990] when projecting the geometry. The shading calculations are performed in a separate pass purely operating on the image buffers, hence, the name *deferred*. While this requires to store a lot of intermediate data (the G-buffer), it reduces the shading costs by avoiding computations for occluded surfaces. The technique became practical for interactive rendering with the advent of multiple render targets and an increase in GPU memory [Hargreaves, 2004].

As an important side effect, the G-buffer can also be used for other techniques, so-called screen-space methods that operate on the buffer data. The algorithm on screen-space bent normals and cones that we introduce in Chapter 3 belongs to this class.

Shadow Mapping Another important and well-used application of rasterization is shadow mapping [Williams, 1978]. It allows for testing direct visibility of a point towards the light source. The construction is as follows. Instead of computing the first visible surface from the camera, shadow mapping computes what the *light* sees, i.e., the first geometry hit by emitted light (see Figure 2.3). Similar to the requirement for the camera, also the light source must have a projective transformation, which is the case for the basic types of spot, and directional light sources. The projective mapping is easily calculated given the position and properties of the light source. Having computed a per-pixel depth map – the shadow map – one can test visibility of any point $\mathbf{x}$ against it. First the point is transformed into the light space of the light source. The resulting x, y coordinates determine the texel index i where the stored depth z_i is located. The actual visibility test is a simple depth comparison, just as the one used for the z-buffer computation:

$$\mathcal{V}_{z_i}(z_\mathbf{x}) = \begin{cases} 1 & \text{if } z_\mathbf{x} \leq z_i \\ 0 & \text{else} \end{cases}. \tag{2.1}$$

We denote $z_\mathbf{x}$ as the depth of $\mathbf{x}$ in light space, not world space. Further, z_i is implicitly given by the position of $\mathbf{x}$ in light space, therefore, we do not use it as a parameter.

Unfortunately, shadow mapping may not return correct visibility, which we denote as V. The algorithm implicitly discretizes the emission domain of the light source (the light space) to a regular grid when computing the shadow map. Hence, shadow mapping is only exact at texel centers. The function above turns this sparse sampling at the texel centers into a piecewise constant function over entire texels. This happens when computing the texel index i from the x, y coordinates of the projected point $\mathbf{x}$. The resulting artifacts are shadow aliasing, i.e., jaggy shadow edges, and shadow acne, i.e., false self-shadowing. A lot of research addresses these issues using



Figure 2.3: Surface shading using shadow maps for the same scene as shown in Figure 2.2. A shadow map (left) encodes the distance of the light source to the closest surface and enables the computation of visibility (middle). A combination of it with the light color (i.e., emitted radiance) and local reflection behavior, defined by the BRDF, yields the shading result (right) for direct lighting.

shadow-map filtering [Reeves et al., 1987; Donnelly and Lauritzen, 2006; Lauritzen and McCool, 2008; Annen et al., 2007, 2008; Salvi, 2008], light-space warping [Stamminger and Drettakis, 2002; Wimmer et al., 2004; Martin and Tan, 2004], storing auxiliary geometry data [Sen et al., 2003; Lecocq et al., 2014; Dou et al., 2014], or using adaptive sampling [Fernando et al., 2001; Aila and Laine, 2004; Sintorn et al., 2008]. We will shortly investigate light space warping and filtering as we will use both approaches for our prefiltered single-scattering technique, which will be explained in Chapter 4.

Light-space warping techniques change the sampling distribution of visibility within the emitter domain. Standard light-space calculation uses a uniform (or close to uniform) distribution. For example, computing a shadow map using an orthogonal transformation for a directional light source causes the light rays that correspond to the texel centers to be equally spaced. However, the distribution of the shading points, for which visibility is queried, is usually not uniform. Shadow map aliasing is typically visible for shadows on surfaces close to the camera [Stamminger and Drettakis, 2002] as neighboring shading points are very close to each other, requiring a denser sampling of the visibility. The shadow-map sampling can be changed by using a different transformation of the scene when performing the projection operation. However, to leverage rasterization, the transformation has to be projective, which ensures that lines are still lines after the transformation [Heckbert, 1989]. Stamminger and Drettakis [2002] propose to first apply the camera transformation and then apply the standard *uniform* light space transformation, resulting in an overall non-uniform sampling. The camera transformation – assuming a perspective camera – causes points close to the camera in world-space to become more distant to each other in post-perspective space. Consequently, a uniform sampling of this post-perspective space causes a higher sampling density in regions close to the camera in world space. While this technique strongly reduces under-sampling artifacts close to the camera, visibility for surfaces further away may now suffer from under-sampling. Later techniques try to find a better tradeoff [Wimmer et al., 2004; Martin and Tan, 2004]. We will also investigate a custom projective transformation (Chapter 4), not to counter under-sampling, but to ensure special properties of the camera rays in light space.

Filtering is used to increase the quality of the reconstruction of the visibility function, which is only sparsely sampled by shadow maps. In image processing low-band filters (such as a Gaussian filter) are used to remove high frequencies from an image. However, two reasons prevent such an approach for visibility filtering: First, shadow maps do not represent visibility itself, but store the distance to the closest surface. Second, visibility is a non-linear function, it returns either one or zero. Hence, a filtering of the input values (depth value) does not result in a properly-filtered visibility. Percentage-closer filtering (PCF) [Reeves et al., 1987] presents the

simplest filtering mechanism and accounts for both of these aspects. It averages multiple visibility queries performed against the shadow map values in a small window $\mathcal{P}(\mathbf{x})$ around the projected position of the position $\mathbf{x}$:

$$\mathcal{V}_{\text{PCF}}(z_{\mathbf{x}}) = \frac{1}{||\mathcal{P}(\mathbf{x})||} \sum_{j \in \mathcal{P}(\mathbf{x})} \mathcal{V}(z_j, z_{\mathbf{x}}),$$
$$\mathcal{V}(z_j, z_{\mathbf{x}}) := \begin{cases} 1 & \text{if } z_{\mathbf{x}} \leq z_j \\ 0 & \text{else} \end{cases}. \tag{2.2}$$

Compared to Equation 2.1 the texel index z_j is not implicit, but given by the texel neighborhood $\mathcal{P}(\mathbf{x})$, which is why we pass z_j as an argument to $\mathcal{V}$. The neighborhood $\mathcal{P}(\mathbf{x})$ is usually a fixed window of at least 3×3 texels around the texel of the projection of $\mathbf{x}$ into the shadow map (i in Equation 2.1). Effectively, PCF blurs shadow boundaries, which is often more appealing but requires many shadow-map queries for each shading point $\mathbf{x}$.

The technique of convolution shadow maps (CSM) [Annen et al., 2007] addresses the latter issue by applying a filtering on data that is computed from the shadow-map depth values, i.e., it filters the visibility function. $\mathcal{V}$ is linearized and it is exploited that for a linear function: $f(ax+by) = af(x)+bf(y)$. Annen et al. observe that the visibility function can be modeled by a Heaviside step function. Applying trigonometric identity operations on the Fourier series of the Heaviside step function leads to an expression of the form $\mathcal{V}_{\text{CSM}}(z_j, z_{\mathbf{x}}) = \sum_i^{\infty} a_i(z_{\mathbf{x}}) B_i(z_j)$. $\mathcal{V}_{\text{CSM}}$ has the form of a linear combination of basis functions, which allows for a filtering of the coefficients and basis functions. In particular the basis functions $B_i(z_j)$ are of interest as they only depend on the values stored in the shadow map. For a correct reformulation of $\mathcal{V}$ via $\mathcal{V}_{\text{CSM}}$ an infinite number of bases is required, which is simply truncated in practice (e.g., using 16 coefficients and basis functions). The last issue is that of the pixel neighborhood $\mathcal{P}$ that defines the z_j and depends on $\mathbf{x}$. CSM, however, apply the filtering on the basis functions $B_i(z_j)$ without any knowledge regarding $\mathbf{x}$ (hereby filtering visibility for all possible $\mathbf{x}$). This theoretical issue is easily solved by simply using multiple fixed filter-kernel sizes to filter the maps of basis functions. Follow-up work [Annen et al., 2008] improves performance by using a different linearization with less data to filter. Statistical methods [Donnelly and Lauritzen, 2006; Lauritzen and McCool, 2008] work similarly but the theory of filtering the input values to get a filtered response is based on statistical rules. In Chapter 4 we will give a more detailed derivation of filtered visibility with the form of basis functions and coefficients. We will use this linearization of visibility not to reduce shadow-mapping artifacts, but to turn single-scattering computations into a filter operation.

For a more in-depth exploration of shadowing techniques, we refer to the excellent book by Eisemann et al. [2011], which covers shadow mapping, but also other interactive techniques such as shadow volumes [Crow, 1977] exhaustively. The speed at which shadow maps can be computed and the simplicity of it make it the de-facto standard in rasterization-based graphics pipelines. We will use shadow mapping as a way to resolve visibility for single-scattering in homogeneous as well as heterogeneous participating media.

2.3 Interaction of Light with Surfaces

Having covered visibility computations, we can look at the actual light transport to perform physically-based image synthesis. Surface rendering is only concerned with the interaction of light with solid objects. It assumes that the rest of the scene is filled with a vacuum, which does not influence the behavior of light. Hence, light can only travel in straight lines (assuming ray optics) from surface to surface. Light interacting with a surface can hereby result in light absorption, reflection, or refraction.

In the following, we will first introduce the complete problem definition of surface rendering and briefly introduce solutions to global illumination for completeness. We then focus on the simplified problem of environmental illumination and its derivative of ambient occlusion. In our own work, we will reintroduce important elements for a real-time approximation of ambient occlusion (Chapter 3).

2.3.1 The Rendering Equation

Light transport for surfaces is mathematically described by the rendering equation [Kajiya, 1986]:

$$\underbrace{L_{\text{surface}}(\mathbf{x}_\text{s}, \vec{\omega}_\text{o})}_{\text{outgoing}} = \underbrace{L_\text{e}(\mathbf{x}_\text{s}, \vec{\omega}_\text{o})}_{\text{emitted}} + \underbrace{\int_{\Omega^+} f_r(\mathbf{x}_\text{s}, \vec{\omega}_\text{i}, \vec{\omega}_\text{o})\, L_\text{i}(\mathbf{x}_\text{s}, \vec{\omega}_\text{i})\, \langle \vec{\mathbf{n}} \cdot \vec{\omega}_\text{i} \rangle \, \mathrm{d}\, \vec{\omega}_\text{i}}_{\text{reflected}}. \tag{2.3}$$

The intuition behind this equation is as follows: the amount of light that leaves the surface point $\mathbf{x}_\text{s}$ into a direction $\vec{\omega}_\text{o}$ is the sum of the locally emitted L_e and reflected light. Light that is potentially reflected can arrive from any possible direction, for solid objects this is limited to the upper hemisphere Ω^+ around the surface normal $\vec{\mathbf{n}}$. The reflection behavior is defined by the material properties at $\mathbf{x}_\text{s}$, mathematically represented by the bidirectional reflectance distribution function f_r (BRDF) [Nicodemus et al., 1977]. Refractions are supported by the bidirectional scattering distribution function (BSDF), which requires an integration over the full sphere. For both functions, the BRDF and BSDF, it is assumed that light is reflected or refracted at the same point as it is incident. More sophisticated models such as the bidirectional surface scattering reflectance distribution function (BSSRDF) lift this restriction. Finally, to synthesize an actual image, one computes what surface the camera sees (per pixel) and solves the rendering equation for that point and the direction towards the camera. Optionally, a sensor model (i.e., camera response function) can be applied. For clarity reasons, we will ignore the emission term in all following equations for surface rendering as it is only non-zero for light sources, which are usually modeled explicitly (e.g., as an environment map, see Section 2.3.3).

Difficulties in Solving the Rendering Equation The rendering equation cannot be solved analytically for non-trivial scenes and we do not want to impose any restrictions on the geometry, material, or lighting of the scene. There are the two theoretical problems of global illumination and high-frequency functions, but also the practical problem of leveraging the GPU (and its rasterization) for interactive applications. Global illumination requires us to solve the rendering equation again for the incoming light L_i at the surface point at which the incoming light originates. This recursiveness is not bounded as light may bounce infinitely many times. But even with a simplified model of L_i such as environmental illumination without global illumination, f_r and L_i may be high-frequency functions, which requires many samples to numerically estimate the integral.

2.3.2 Global Illumination

For completeness, we briefly give an overview of important methods that try to solve the full rendering equation, also addressing global illumination. In the next section, we will discuss the problem of environmental illumination that can be seen as an approximation of global illumination.

General Solutions using Random Sampling Monte Carlo estimation is a general way to solve definite integrals by using a random variable as a representative to evaluate the integrand. Repeating this sampling with uncorrelated random numbers and averaging the computed values results in a convergence towards the solution of the integral. Applied directly to the rendering equation, one samples incoming light directions as representatives

for all other directions. Path tracing [Whitted, 1980; Cook et al., 1984; Cook, 1986; Kajiya, 1986], as well as its bidirectional extension [Lafortune and Willems, 1993; Veach and Guibas, 1994], directly employ such a scheme to estimate the rendering equation. Other algorithms such as photon mapping [Jensen and Christensen, 1995] and Metropolis light transport [Veach and Guibas, 1997] rely on random sampling of parts of the light paths or path perturbations. While a Monte-Carlo approach allows us to simulate all effects described by the rendering equation, it results in variance in the estimates. This variance shows up as high-frequency noise in the image for non-converged results. Variance can be reduced by intelligently placing samples (importance sampling), which requires a weighted average of the sample contributions. While all practical methods importance sample parts of the light transport, it is not possible to importance sample the entire path from the light source to the camera (as this would already be a solution to the rendering equation up to a constant factor). Variance can further be reduced using multiple importance sampling (MIS) [Veach and Guibas, 1995], which combines different methods of sampling light transport (i.e., different sampling strategies) to yield an improved overall sampling of the light transport. The approach recently gained attention in research [Georgiev et al., 2012; Hachisuka et al., 2012; Křivánek et al., 2014]. However, even with all those advances, many rays (typically hundreds per pixel) are needed for the estimation to converge. Another downside for interactive applications is that Monte-Carlo methods rely on the ability to trace arbitrary rays, which prevents them from using rasterization efficiently. Due to these reasons, Monte Carlo (or generally ray tracing)-based methods are typically used for offline rendering only.

Interactive Rasterization-based Approaches Time-critical applications such as games or artist tools that need to provide feedback ofter, have to deliver images instantaneously. This can be achieved by simplifying light transport and using GPUs with their hardware-accelerated rasterization, i.e., the projection of geometry on an image. As mentioned before, shadow mapping presents an efficient method that uses rasterization. It is, however, limited to one-bounce paths, as only direct visibility can be tested. Instant radiosity [Keller, 1997] is a global illumination algorithm that effectively turns global illumination due to many light bounces into direct illumination only. It does so by shooting photons that are turned into virtual point lights (VPLs) at the intersection with surfaces. Such a reformulation of the rendering problem can also be shown mathematically by transforming the integral of the rendering equation into a Neumann series (infinite sum over all path lengths). Illumination due to the virtual lights is then treated as direct illumination and can be solved efficiently using shadow maps. However, there are multiple problems. First, the initial creation of the VPLs requires the tracing of photons that bounce in the scene. While the original algorithm used ray tracing with unlimited bounces (hence, supporting full global illumination), real-time approaches often limit this to one-bounce indirect illumination by sampling a reflective shadow map [Dachsbacher and Stamminger, 2005], i.e., a shadow map storing also surface normal and light flux. Second, from a sampling point of view, VPLs represent highly correlated light paths, which cause splotch artifacts (high energy areas) if an insufficient amount of VPLs is used. While the splotches can be easily detected and avoided by clamping, the clamping results in bias and dark corners (where most of the splotches are), which can be addressed by approximate compensation techniques [Novák et al., 2011]. Finally, many VPLs are required to accurately represent global illumination, a problem considered for offline rendering [Walter et al., 2005; Hašan et al., 2007], and more recently also for interactive rendering [Ritschel et al., 2011]. As creating many shadow maps is costly, visibility is either omitted [Dachsbacher and Stamminger, 2005, 2006], or a point-based representation of the scene can be used to compute imperfect shadow maps [Ritschel et al., 2008]. Such a representation is efficient to rasterize, but leads to holes, which are filled in an image-based approach.



Figure 2.4: High dynamic range light probes in latitude longitude format, storing the illumination in the Grace Cathedral and Uffizi Gallery. Probes captured by Debevec [1998].

The idea of using simple geometric proxies such as spheres or disks are used in so-called point-based global illumination techniques [Christensen, 2008; Ritschel et al., 2009a; Holländer et al., 2011; Kontkanen et al., 2011; Buchholz and Boubekeur, 2012; Wang et al., 2013], which rely on final gathering. Final gathering applies another indirection when computing the shading for a point, i.e., VPLs are not required for one-bounce indirect illumination. The interactive method of Ritschel et al. [2009a] renders micro-buffers for final gathering at sparse image locations. Those micro-buffers can be seen as a rendering of the scene with direct illumination from that surface point (the shading point) and therefore represent the incoming light L_i. To avoid the repeated rasterization of the full scene for each micro-buffer, a hierarchical sphere representation is rasterized, very similar to the one used for imperfect shadow maps. Summing the micro-buffer (i.e., computing the integral over all incoming directions) solves indirect illumination for the shading point.

Another common simplification is the use of diffuse materials as the BRDF can then simply be removed from the integral in Equation 2.3. However, the final gathering approach even allows for some specularity. Finally, a more comprehensive overview of interactive global-illumination techniques is given by Ritschel et al. [2012].

2.3.3 Environmental Illumination

Global illumination is difficult to solve due to the recursiveness in the rendering equation. Environmental illumination [Blinn and Newell, 1976] (also called image-based lighting) tries to find a tradeoff between rendering complexity and light-transport realism. It does so by assuming that far away scene objects can be approximated by a scene-surrounding imposter that acts as a light source. This approximation not only avoids the modeling and simulation of light transport in those regions, but also enables the use of natural illumination. In practice, real-world photographs are used to producing natural illumination in a virtual scene [Debevec, 1998]. The novel light source is distant by definition, i.e., it is spatially invariant: $L_\mathrm{env}(\mathbf{x},\vec{\omega}) = L_\mathrm{env}(\vec{\omega})$. To meet the requirement of distant illumination, the scene extent should be small compared to the distance of the source of illumination. Therefore, the model works well for outdoor scenes, where most of the incoming illumination is caused by the sky.

A further simplification is to completely abandon global illumination in the scene and only rely on the direct illumination due to the image-based environment. While simplified, the rendering with environmental illumination is still difficult though. Ignoring global illumination, the expression of the incoming light simplifies to direct illumination from the environment map $L_\mathrm{i}(\mathbf{x}_\mathrm{s},\vec{\omega}_\mathrm{i}) := L_\mathrm{env}(\vec{\omega}_\mathrm{i})\, V(\mathbf{x}_\mathrm{s},\vec{\omega}_\mathrm{i})$. Nonetheless, it is still required to integrate over the hemisphere to solve for the outgoing radiance at a surface point:

$$L_\mathrm{o}(\mathbf{x}_\mathrm{s},\vec{\omega}_\mathrm{o}) = L_\mathrm{e}(\mathbf{x}_\mathrm{s},\vec{\omega}_\mathrm{o}) + \int_{\Omega^+} f_r(\mathbf{x}_\mathrm{s},\vec{\omega}_\mathrm{i},\vec{\omega}_\mathrm{o})\, L_\mathrm{env}(\vec{\omega}_\mathrm{i})\, V(\mathbf{x}_\mathrm{s},\vec{\omega}_\mathrm{i})\ \langle\vec{\mathbf{n}}\cdot\vec{\omega}_\mathrm{i}\rangle\ \mathrm{d}\,\vec{\omega}_\mathrm{i}. \tag{2.4}$$

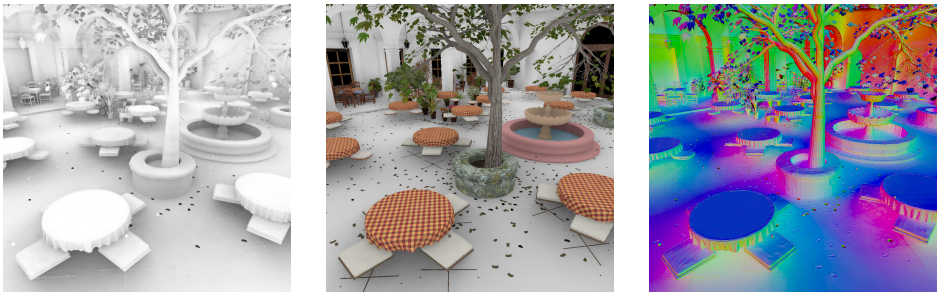


Figure 2.5: Example of ambient occlusion. From left to right: ambient occlusion (here computed in screen space), multiplied with a diffuse material, and bent normals. Although not physically based, ambient occlusion gives an impression of global illumination. A key aspect is the object-object relationship that becomes visible due to the distance of objects to each other in world space.

The difficulty in solving the equation lies in the high-frequency nature of the functions. Visibility V is high-frequency by definition as light is either blocked or not, but also the environment map can contain high frequencies as is shown in Figure 2.4. Hence, still many samples are required to estimate the integral, even when using a diffuse material that parameterized by an albedo ρ ($f_r(\mathbf{x}_s, \vec{\omega}_i, \vec{\omega}_o) = \rho/\pi$). To counter the high-frequency noise, a number of specialized environment-map-sampling techniques have been proposed [Secord et al., 2002; Kollig and Keller, 2003; Agarwal et al., 2003; Clarberg et al., 2005; Burke et al., 2005; Ghosh and Heidrich, 2006]. Alternatively, high quality can be achieved by projecting the illumination and/or BRDF into finite bases such as spherical harmonics [Ramamoorthi and Hanrahan, 2001, 2002] or wavelets [Ng et al., 2003]. Combined with precomputations, these bases allow for high-quality environment illumination even at interactive rates [Sloan et al., 2002]. In the following we will continue the simplification of the rendering equation for even more efficient solutions.

2.3.4 Ambient Occlusion and Bent Normals

The computational problem of environmental illumination is the product of the illumination L_{env} and the visibility term V, as both can be of high frequency and their product is difficult to sample. However, both terms are important. As said before, the illumination acts as a relatively cheap approximation of global illumination and simplification of scene modeling. The remarkable importance of occlusion as a visual cue was first described by Langer and Zucker [1994]. They found that the human ability to understand shape from shading combines common directional illumination and shadows, as well as soft ambient lighting, and *ambient occlusion* (AO) found on a cloudy day.

Miller [1994] introduced accessibility shading that was modified by Zhukov et al. [1998] to AO. The main assumption of rendering with ambient occlusion is that V can be moved outside of the integral, as

$$L_o(\mathbf{x}, \vec{\omega}_o) \approx AO(\mathbf{x}) \frac{\rho}{\pi} \int_{\Omega^+} L_{env}(\vec{\omega}_i) \langle \vec{\mathbf{n}} \cdot \vec{\omega}_i \rangle \, \mathrm{d}\vec{\omega}_i, \tag{2.5}$$

where

$$AO(\mathbf{x}) := \frac{1}{2\pi} \int_{\Omega^+} V(\mathbf{x}, \vec{\omega}_i) \, \mathrm{d}\vec{\omega}_i. \tag{2.6}$$

Basically, AO decouples shading from visibility: light from all directions is equally attenuated by the average visibility over all directions (Figure 2.5). Whether this assumption applies, highly depends on the lighting and materials [Yu et al., 2009]. The ambient-occlusion term can also be seen as special environmental illumination with constant irradiance, i.e., the incoming radiance $L_{env}(\vec{\omega}_i) = 1$ (whether or not the geometric term $\langle \vec{\mathbf{n}} \cdot \vec{\omega}_i \rangle$ is

included in AO differs in the literature). Rendering performance can be increased as both integrals (illumination and occlusion) can be treated separately. Often, illumination can be preconvolved, i.e., the illumination integral is solved for many different outgoing directions $\vec{\omega}_o$ in a preprocess.

In the production settings considered by Landis [2002], ray tracing was used to compute the hemispherical integral of AO using Monte Carlo integration: a set of randomly oriented rays R is cast into the hemisphere to evaluate V and the result is averaged:

$$AO_{\mathrm{MC}}(\mathbf{x}) := \frac{1}{|R|} \sum_{\vec{\omega} \in R} V(\mathbf{x}, \vec{\omega}) \approx AO(\mathbf{x}). \tag{2.7}$$

Even with all these simplifications, this estimation still requires many samples if the visibility function is of high frequency.

Bent Normals

The decoupling of visibility and lighting causes a loss in the variation of shading, as AO ignores the directionality of the lighting. The idea of *bent normals* dates back to Landis [2002] where it was proposed as a way of adding this directionality with the same level of simplicity as AO. Bent normals are the mean free direction scaled by the mean occlusion and are used for shading instead of the surface normals. Different from AO, their definition includes the direction $\vec{\omega}$ inside the integral:

$$\vec{N}(\mathbf{x}) := \frac{1}{\pi} \int_{\Omega^+} V(\mathbf{x}, \vec{\omega})\, \vec{\omega}\, \mathrm{d}\, \vec{\omega}. \tag{2.8}$$

For lighting computations, bent normals simply replace the surface normal and the visibility term:

$$L_o(\mathbf{x}) \approx \frac{1}{\pi} \int_{\Omega^+} L_{\mathrm{env}}(\vec{\omega}) \left\langle \vec{N}(\mathbf{x}) \cdot \vec{\omega} \right\rangle \mathrm{d}\, \vec{\omega}.$$

The Monte-Carlo computation of bent normals $\vec{N}_{\mathrm{MC}}(\mathbf{x})$ requires to multiply visibility with the direction, which is as computationally simple and efficient as AO_{MC} alone. Different to common normals, bent normals are not normalized and their length represents the ambient occlusion value.

Interactive Approaches

Due to its success in offline production, considerable effort was made to support AO in interactive rendering. Pharr and Green [2004] used several shadow maps to compute ambient occlusion for a single static object. Bunnell [2005] compute AO in dynamic scenes using a finite-element approach that approximates the surfaces as a hierarchy of discs. AO is gathered at vertices or pixels in a fast GPU multi-pole approach. The incorrect linear summation of visibility of different occluders can be improved by a multi-pass approach. The method can compute bent normals and even full indirect illumination. Rigidly transformed static objects can benefit from a precomputation that stores AO in a discrete 3D grid [Kontkanen and Laine, 2005; Malmer et al., 2007; Kroes et al., 2015].

A breakthrough in terms of performance was made with the advent of screen-space ambient occlusion (SSAO) [Mittring, 2007; Shanmugam and Arikan, 2007]. Input to the method is solely a rendering of the scene into a deferred shading buffer [Saito and Takahashi, 1990]. Occlusion of nearby geometry is then determined by sampling and evaluating the depth buffer. We will give a more detailed description of the SSAO computation in Chapter 3. SSAO has many desirable properties: no assumptions are made about the geometry, dynamic scenes are trivially supported, it works output-sensitive (AO is computed for visible pixels only), avoids intermediate data structures, and is easy to integrate into deferred-shading pipelines, the state of the art in interactive rendering.

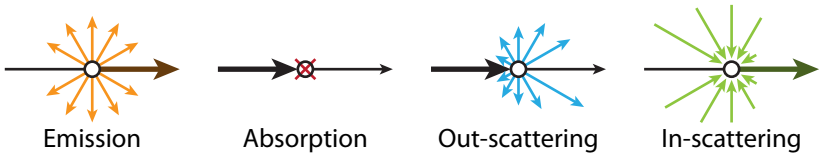


Figure 2.6: Events that can happen in a participating media.

Further research has focused on increasing sampling efficiency to improve performance and avoid artifacts [Bavoil et al., 2008; Loos and Sloan, 2010; Huang et al., 2011; McGuire et al., 2011; Timonen, 2013; Kroes et al., 2015; Hendrickx et al., 2015]. The main artifact of any screen-space technique is incorrect occlusion as it only uses incomplete information about the scene: geometry that is occluded or outside the viewport cannot be accounted for. It has been proposed to compute depth buffers for multiple view directions [Ritschel et al., 2009b; Vardis et al., 2013] but this strongly contradicts with the goal of high performance. The recent approach of Nalbach et al. [2014a] proposes to operate in a *deep screen space* to account for hidden geometry. Instead of applying the normal rasterization of the scene, they use the programmable hardware to create an on-the-fly point-based representation of the geometry within the camera frustum. While they introduce intermediate data, they keep all of the other mentioned advantages of SSAO. Due to the custom geometry processing some performance is lost, but quality can be improved as hidden geometry can be accounted for. The commonly observed popping artifacts are avoided and temporal stability of real-time ambient occlusion is increased. Working in (deep) screen space is only approximate, but its efficiency made SSAO a core component in most modern real-time rendering engines.

In Chapter 3 we will show how SSAO can be extended to compute bent normals as well as our novel bent cones (variance of unoccluded directions) in screen space. Both techniques address the directionality issue and improve rendering accuracy with environmental illumination. As this topic was already handled in previous work [Ritschel et al., 2009b], we also show the differences and provide comparisons. Later work proposes the use of spherical harmonics [Herholz et al., 2012]. It is similar in the spirit that the mean and variance that we compute are captured by the first three orders of spherical harmonics, but higher orders can be added if desired.

2.4 Light Transport in Participating Media

Light transport in participating media is similar to the one for surfaces but scattering can happen virtually anywhere in space. As light travels through a participating medium, the following types of interaction can occur: emission, absorption, out-scattering, and in-scattering (see Figure 2.6). Finally, for scenes with solid objects and participating media, we also need to consider the interaction of light with surfaces as described previously.

We will shortly introduce common assumptions on participating media, which define how they are modeled. We then present the general equations to describe light transport in participating media before looking at the problem of computing single scattering for homogeneous media. As mentioned before, we refer to previous work for a more in-depth derivation and explanation [Jarosz, 2008].

2.4.1 Assumptions on Participating Media and Their Properties

Most applications in computer graphics treat participating media as a collection of microscopic particles that are not modeled explicitly. The particles are randomly positioned as well as oriented, which allows for a statistical description of scattering in a medium. For the validity of such a probabilistic description, it is assumed that the

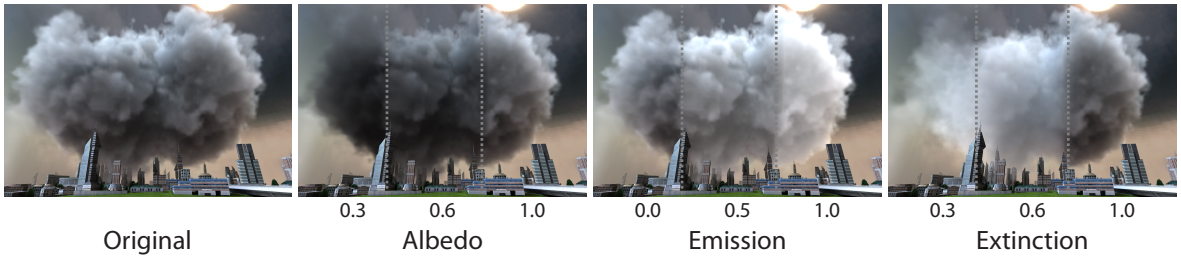


Figure 2.7: Parameters of a participating medium. The parameters are scaled locally, showing their effect on the appearance of a heterogeneous participating medium. The phase function is assumed to be isotropic. For the rendering, we used the approximation described in Equation 2.17, i.e., we only consider single scattering under environmental illumination.

interaction of light with one particle is statistically independent of the interaction with other particles. This is valid if the average particle size is much smaller than the spacing to neighboring particles.

Properties

In the following, we list the parameters that define a participating medium. We visualize the effect of a local scaling of each of them in Figure 2.7.

Extinction Coefficient The extinction coefficient σ_t is the quantity that is related to the probability that a photon interacts with the medium. The quantity depends on the density of the volume (i.e., the number of particles per volume) as well as the average 2D cross section of the particles. The latter is due to the random orientation of the particles as well as the assumption that light travels in straight lines. The probability of a photon *not* interacting with the medium for a line segment is given by the transmittance:

$$T_r(\mathbf{x},\mathbf{x}') = e^{-\tau(\mathbf{x},\mathbf{x}')}, \tag{2.9}$$

with τ being the optical thickness, which is computed as the line integral over the extinction coefficient:

$$\tau(\mathbf{x},\mathbf{x}') = \int_0^d \sigma_t(\mathbf{x}+t\vec{\omega})\,\mathrm{d}t, \tag{2.10}$$

where $\mathbf{x}+t\vec{\omega}$ parametrizes the line segment between the points $\mathbf{x},\mathbf{x}'$ and d denotes the distance between the two points.

Scattering and Absorption Coefficient In the case that a photon hits a particle, it may either by absorbed or scattered into another direction. Those events are described by the absorption coefficient, σ_a, and the scattering coefficient, σ_s, respectively. As the extinction coefficient relates to any interaction of a photon with the medium, it holds that: $\sigma_t = \sigma_a + \sigma_s$. The probability of scattering in the case of interaction in the medium can be described by the albedo $\rho = \sigma_s/\sigma_t$, which is the volumetric equivalent of the albedo for diffuse surface materials.

Emission Finally, the participating medium may emit light on its own by turning some energy into visible light. An example of such a medium is fire. This conversion process is not further described as we are only interested in the resulting emitted radiance $L_e(\mathbf{x},\vec{\omega})$. While some literature introduces a new set of particles, described by the coefficient σ_e, computer graphics often just links the phenomena to only the particles that absorb light σ_a or all particles σ_t. This linking imposes no constraints on L_e, and we use the latter.

Phase Function Finally, in the case of scattering, light may change its direction, which is defined by the phase function $p(\mathbf{x}, \vec{\omega}_\mathrm{i}, \vec{\omega}_\mathrm{o})$. It is simply the volumetric equivalent to the BRDF for surfaces. While many more phase functions have been proposed in the literature, the two most common phase functions are the isotropic phase function, $p(\mathbf{x}, \vec{\omega}_\mathrm{i}, \vec{\omega}_\mathrm{o}) = \frac{1}{4\pi}$, and the Henyey-Greenstein phase function. The latter is anisotropic and has only a single parameter g to describe the anisotropy:

$$p_\mathrm{HG}(\mathbf{x}, \theta, g) = \frac{1-g^2}{4\pi(1+g^2-2g\cos\theta)^{1.5}}, \tag{2.11}$$

with θ being the angle between $\vec{\omega}_\mathrm{i}$ and $\vec{\omega}_\mathrm{o}$.

Evaluating Transmittance

A participating medium is called homogeneous if $\sigma_\mathrm{t}(\mathbf{x})$ is spatially constant, otherwise it is called heterogeneous. The computation of transmittance in homogeneous media simplifies to $T_\mathrm{r}(\mathbf{x}, \mathbf{x}') = e^{-\sigma_\mathrm{t} d}$, with d being the distance between the points $\mathbf{x}, \mathbf{x}'$. As this can be evaluated analytically, homogeneous media are generally much easier to render and therefore more accessible for interactive applications.

For heterogeneous media, the difficulty lies in evaluating the integral of the optical thickness $\tau(\mathbf{x}, \mathbf{x}') = \int_0^d \sigma_\mathrm{t}(\mathbf{x} + t\vec{\omega})\,\mathrm{d}t$. As we do not want to impose any restrictions on the extinction coefficient $\sigma_\mathrm{t}(\mathbf{x})$ or its spatial variation, the integral cannot be solved analytically. The common way of estimating integrals in computer graphics is Monte-Carlo estimation, as it is unbiased and allows for an easy trade-off between accuracy and performance. As mentioned before, it uses random samples to estimate the solution and relies on the fact that the expected value equals the true solution of the integral. Unfortunately, Monte-Carlo integration cannot be used to compute transmittance. While it is possible to estimate optical thickness (as it is a plain integral), transmittance requires exponentiation and $E[\exp(X)] \neq \exp(E[X])$, with X denoting a random variable and E the expected value of it. Hence, one has to numerically estimate the transmittance as accurately as possible since averaging different samples does not converge to the correct solution. The probably most common numerical integration method uses the quadrature rule. To apply it, the extinction needs to be sampled at regular intervals of the line segment between $\mathbf{x}$ and $\mathbf{x}'$. Practically, this is achieved by marching along a ray (from $\mathbf{x}$ to $\mathbf{x}'$) and aggregating the optical thickness.

An alternative is Woodcock tracking [Woodcock et al., 1965], which has been introduced for volume rendering by Raab et al. [2006]. It has the desirable property of allowing for averaging the results of probabilistic samples. It is not a method to estimate transmittance per se, but provides an unbiased estimate of the location of the next scattering event. It does so by using a seemingly simple trick: the heterogeneous medium is filled up with virtual particles such that it becomes homogeneous and the next scattering event can be computed analytically. Whether an interaction happens with a real or virtual particle is decided randomly based on the local extinction coefficient. To achieve the desired scattering behavior of heterogeneous media, virtual particles do not influence the light, i.e., a photon is scattered in the same direction as it was traveling before. Transmittance can now be estimated by repeatedly testing if a photon interacts with any real particle on its way from $\mathbf{x}$ to $\mathbf{x}'$. Woodcock tracking therefore acts as a binary estimator. While this enables us to compute an unbiased estimate of transmittance, it is very inefficient, a fact addressed by the recent work of Novák et al. [2014].

Finally, efficiency in repeated transmittance computations can be increased by caching. Shadow maps have been introduced as an efficient image-based way to test visibility regarding solid objects (see Section 2.2.3). Deep shadow maps [Lokovic and Veach, 2000] as well as opacity shadow maps [Kim and Neumann, 2001] have been proposed as extensions to participating media. However, the motivation for shadow maps was that they can leverage the efficiency of hardware rasterization. Rasterization cannot be used for participating media as

they cannot be efficiently projected onto an image plane. However, such maps also act as an efficient cache: visibility and transmittance towards the light source can be evaluated for any point. This issue is even more important for transmittance as its computation is more involved compared to visibility. Texels of the deep and opacity shadow maps store information about the transmittance function, in case of deep shadow maps a piecewise-linear approximation is used. This 1D function describes the transmittance from the light source to any point along the light ray. Hence, the maps can be constructed by ray marching away from the light source through the medium for each texel and constantly updating the optical thickness to compute the transmittance for different distances to the light. Hence, the resulting data has a 3D structure. Similar to shadow mapping, transmittance toward the light source can be evaluated for any point in two steps: first, the point is projected into light space to select the right 2D texel position, i.e., transmittance function. Then, given the z-coordinate, transmittance is either reconstructed from the function (deep shadow maps), or can be looked up directly (opacity shadow maps). Besides the 2D resolution of shadow maps, the volumetric equivalent has to make a compromise regarding the resolution at which the transmittance function is sampled and stored. We will use opacity shadow maps to efficiently evaluate transmittance for directional light sources that we sample from an environment map (Chapter 7).

2.4.2 The Radiative Transfer Equation

Having introduced the description of participating media, we can look at how radiance changes along a direction $\vec{\omega}$ based on a medium's properties. Four different events can happen at a point $\mathbf{x}$: light may be spontaneously *emitted* by the medium; light traveling into $\vec{\omega}$ may be *absorbed*, or *out-scattered* in a new direction; finally, light traveling in some other direction $\vec{\omega}'$ may be *in-scattered*, i.e., its direction is changed to $\vec{\omega}$. The latter three happen only due to an interaction event of light with the medium. As such, energy can be gained (emission), lost (absorption), or redistributed (scattering) at any point in the medium. Mathematically, the change of radiance at $\mathbf{x}$ in a direction $\vec{\omega}$ is described by the integro-differential form of the radiative transfer equation (RTE) [Chandrasekar, 1960]:

$$(\vec{\omega}\cdot\nabla)L(\mathbf{x},\vec{\omega}) = -\underbrace{\underbrace{\sigma_\mathrm{a}(\mathbf{x})L(\mathbf{x},\vec{\omega})}_{\text{absorption}} - \underbrace{\sigma_\mathrm{s}(\mathbf{x})L(\mathbf{x},\vec{\omega})}_{\text{out-scattering}}}_{\text{extinction}} + \underbrace{\sigma_\mathrm{t}(\mathbf{x})L_\mathrm{e}(\mathbf{x},\vec{\omega})}_{\text{emission}} + \underbrace{\sigma_\mathrm{s}(\mathbf{x})L_\mathrm{i}(\mathbf{x},\vec{\omega})}_{\text{in-scattering}}. \tag{2.12}$$

To solve for L, we need to integrate both sides, producing an integral equation just as the rendering equation (Equation 2.3). This also introduces the rendering equation into volume rendering as a boundary condition:

$$\begin{aligned} L(\mathbf{x},\vec{\omega}) = {} & T_\mathrm{r}(\mathbf{x},\mathbf{x}_\mathrm{s})\, L_\mathrm{s}(\mathbf{x}_\mathrm{s},\vec{\omega}) \\ & + \int_0^s T_\mathrm{r}(\mathbf{x},\mathbf{x}_t)\, \sigma_\mathrm{t}(\mathbf{x}_t)\, L_\mathrm{e}(\mathbf{x}_t,\vec{\omega})\, \mathrm{d}t \\ & + \int_0^s T_\mathrm{r}(\mathbf{x},\mathbf{x}_t)\, \sigma_\mathrm{s}(\mathbf{x}_t)\, L_\mathrm{i}(\mathbf{x}_t,\vec{\omega})\, \mathrm{d}t. \end{aligned} \tag{2.13}$$

Here, we are integrating over a ray up to the first surface $\mathbf{x}_\mathrm{s}$ at which we need to evaluate the rendering equation to compute L_s (but using Equation 2.13 for the incoming radiance to the surface). The first summand corresponds to surface radiance that is reduced due to out-scattering and absorption (transmittance term $T_\mathrm{r}(\mathbf{x},\mathbf{x}_\mathrm{s})$). The second is the accumulated emitted radiance, which also gets out-scattered and absorbed on its way from $\mathbf{x}_t$ to $\mathbf{x}$. The last is the accumulated in-scattered radiance. We reformulate the equation by combining the latter two:

$$L(\mathbf{x},\vec{\omega}) = T_\mathrm{r}(\mathbf{x},\mathbf{x}_\mathrm{s})\, L_\mathrm{s}(\mathbf{x}_\mathrm{s},\vec{\omega}) + \int_0^s T_\mathrm{r}(\mathbf{x},\mathbf{x}_t)\, \sigma_\mathrm{t}(\mathbf{x}_t)\, L_\mathrm{m}(\mathbf{x}_t,\vec{\omega})\, \mathrm{d}t. \tag{2.14}$$

We express radiance in the medium at a single point as medium radiance, which is a sum of emitted and in-scattered radiance:

$$L_{\mathrm{m}}(\mathbf{x}_t, \vec{\omega}) = L_{\mathrm{e}}(\mathbf{x}_t, \vec{\omega}) + \rho(\mathbf{x}_t) L_{\mathrm{i}}(\mathbf{x}_t, \vec{\omega}). \tag{2.15}$$

The albedo ρ limits in-scattering to only the fraction of particles that actually scatter light. Finally, we describe the in-scattering, through which medium radiance gets a similar form as the rendering equation (cf. Equation 2.3):

$$L_{\mathrm{m}}(\mathbf{x}_t, \vec{\omega}) = L_{\mathrm{e}}(\mathbf{x}_t, \vec{\omega}) + \rho(\mathbf{x}_t) \int_{\Omega^{4\pi}} p(\mathbf{x}_t, \vec{\omega}_{\mathrm{i}}, \vec{\omega})\, L(\mathbf{x}_t, \vec{\omega}_{\mathrm{i}})\, \mathrm{d}\,\vec{\omega}_{\mathrm{i}}. \tag{2.16}$$

Different from surface rendering, we need to integrate over the full sphere of directions as light coming from any direction may be in-scattered. Again, the equation needs to be solves recursively as the incoming light is computed by solving Equation 2.13. Similar to global illumination for surface rendering, the recursive evaluation of scattering (multiple scattering) makes volume rendering difficult.

2.4.3 Volume-Rendering Approaches

While the fundamental mathematical models and even some current methods were introduced studying radiative heat transfer and neutron transport [Chandrasekar, 1960], volume rendering was eventually introduced to computer graphics by Blinn [1982]. Early solutions used finite-elements to discretize the volume into *zones* between which energy was exchanged [Rushmeier and Torrance, 1987]. This early radiosity-based method was combined with Monte-Carlo estimation to handle anisotropic scattering [Rushmeier, 1988]. The first true adoption of path tracing to handle volumetric scattering was then proposed by Hanrahan and Krueger [1993]. We will not explore early previous work in detail and instead refer to a comprehensive overview covering methods until the mid 2000s [Cerezo et al., 2005].

Recently, more surface light-transport algorithms have been successfully generalized to efficiently handle the case of multiple scattering in participating media. Photon mapping has been extended to a beam estimate as well as photon beams [Jarosz et al., 2008, 2011a,b], and virtual point lights to virtual ray lights [Novák et al., 2012a,b]. Very recently, improvements have also been proposed for Monte Carlo importance sampling in participating media [Kulla and Fajardo, 2012; Georgiev et al., 2013], as well as the efficient combination of photon mapping techniques with classical Monte-Carlo integration techniques [Křivánek et al., 2014] – based on the multiple importance sampling technique of Veach and Guibas [1995]. The generality of these methods also enables the handling of surface-medium interactions, but comes at the cost of long computation times, far from interactivity.

Higher performance can be achieved by focusing on the scattering in media only and representing many scatter events by a mathematical model that can be solved efficiently. Diffusion is such a model, and allows for analytic solutions by assuming constant medium properties and a low directional variance in radiance as a result of scattering. It was introduced as a solution for volume rendering by the seminal work of Stam [1995], after earlier work on spherical harmonics by Kajiya and Von Herzen [1984]. The assumption of low variation is usually met by high-order scattering media, i.e., the medium must be very dense to cause many scatter events and have a high albedo so that energy is not lost. Repeated scattering events then effectively reduce directional variance of radiance. Relying on this property, diffusion can be approximately solved via the dipole model, proposed for sub-surface scattering materials [Farrell et al., 1992] in medicine and later introduced to graphics [Jensen et al., 2001]. The dipole model led to a number of rendering techniques [Jensen and Buhler, 2002; d'Eon et al., 2007; Muñoz et al., 2011; d'Eon, 2012; Habel et al., 2013; Frisvad et al., 2014] with the main goal of realistically rendering human skin [Donner and Jensen, 2005, 2006; Donner et al., 2008; Jimenez et al.,

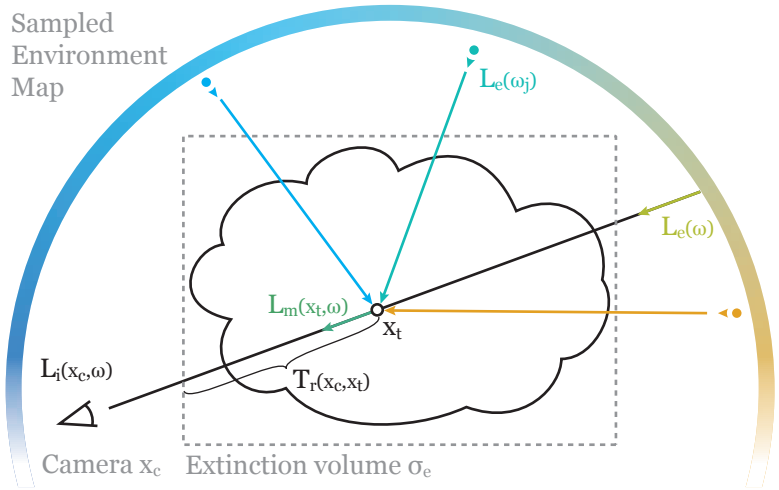


Figure 2.8: Single-scattering model and environmental illumination. Radiance is accumulated at sample points $\mathbf{x}_t$ that are distributed along the view ray of the camera. A sampling of the environment map creates directional light sources that cause attenuated incident radiance at the scatter points. The sampling can be global (i.e., each scatter point uses the same light sources), which allows for caching of visibility and transmittance with the help of shadow-mapping techniques.

2009], but also the model itself was extended and improved to yield better accuracy [Donner and Jensen, 2005; d'Eon and Irving, 2011]. Recent work shows that parameters of dense media can be changed such that rendering performance is increased while the appearance of the medium is maintained [Zhao et al., 2014]. To allow for variation in the media properties, diffusion-based methods often combine the diffusion (or other low-order) models with finite-elements models and hierarchical structures. The main disadvantage of these methods as well as its real-time derivatives [Zhou et al., 2008; Billeter et al., 2012] is that they support only coarse or no object visibility, as this cannot be accounted for in the model and can only be added using the discrete elements for which interchanged energy needs to be computed explicitly.

Environmental Illumination

For completeness, we mention the special case of environmental illumination (Figure 2.8), as we will assume environmental illumination for our medium stylization method in Chapter 7. The motivation for environmental illumination is the same as for surface rendering. It avoids modeling far-away scene content and can be considered a relatively cheap global-illumination / multiple-scattering approximation. The resulting equation is very similar to its equivalent for surface rendering with the illumination being distant by definition:

$$L_\mathrm{m}(\mathbf{x}_t, \vec{\omega}) = \rho(\mathbf{x}_t) \int_{\Omega^{4\pi}} p(\mathbf{x}_t, \vec{\omega}_\mathrm{i}, \vec{\omega})\, L_\mathrm{env}(\vec{\omega}_\mathrm{i}) V(\mathbf{x}_t, \vec{\omega}_\mathrm{i}) T_\mathrm{r}(\mathbf{x}_t, \vec{\omega}_\mathrm{i})\, \mathrm{d}\,\vec{\omega}_\mathrm{i}. \tag{2.17}$$

Besides visibility V, also the transmittance term T_r needs to be accounted for. As the light source is assumed to be at infinity, the participating medium must have a finite extent, as otherwise $T_\mathrm{r}(\vec{\omega}_\mathrm{i})$ is always zero. Please note that we ignore multiple scattering in this definition (similar to our definition of environmental illumination for surface rendering). Please also remember that this equation only describes the radiance due to light interaction (in-scattering) at $\mathbf{x}_t$ into direction $\vec{\omega}$. Volume rendering needs to solve Equation 2.14 with Equation 2.17 as L_m.

As such, unlike environmental illumination for surface rendering, the problem in estimating the radiance is not just L_env and V, but also the continuous in- and out-scattering along rays, which requires ray marching

along camera rays to estimate Equation 2.14 and another ray-marching procedure to estimate transmittance for Equation 2.17. In practice, it is efficient to globally sample the environment map (i.e., independent of any screen pixel, see Figure 2.8) to create a set of directional light sources $\mathcal{D}$ (similar to VPLs, see Section 2.3.2), simplifying Equation 2.17 to:

$$L_{\mathrm{m}}(\mathbf{x}_t, \vec{\omega}) = \rho(\mathbf{x}_t) \sum_{j \in \mathcal{D}} p(\mathbf{x}_t, \vec{\omega}_j, \vec{\omega})\, L_j(\vec{\omega}_j) V(\mathbf{x}_t, \vec{\omega}_j) T_{\mathrm{r}}(\vec{\omega}_j). \tag{2.18}$$

Efficiency is gained as rays are coherent for directional light sources and surface as well as deep/opacity shadow maps can be used to quickly evaluate visibility and transmittance. Still, many lights are required as the global sampling turns the high-frequency Monte-Carlo noise into structured noise (visible as banding artifacts) that is caused by the correlation of the light paths between pixels. Finally, rendering is performed by marching from the camera along view rays (to estimate Equation 2.14) and at each point summing the contributions due to the set of directional light sources (Equation 2.18). We will also represent environmental illumination by a discrete set of directional light sources for the stylization of heterogeneous media as presented in Chapter 7.

To achieve interactive performance, Elek et al. [2014] use a diffusion-based approach to propagate radiance from a sampled set of directional lights in the heterogeneous media (considering multiple scattering). However, like previous work that relies on diffusion, the propagation neglects solid objects in the scene and is therefore only useful for spatially-limited media that do not interact with the remainder of the scene, such as clouds.

2.4.4 Single Scattering for Homogeneous Media

Real-time solutions usually assume single scattering [Zhou et al., 2007; Ren et al., 2008; Nalbach et al., 2014b], similar to direct illumination for surface rendering. A further simplification is the assumption of homogeneous media, where interesting phenomena still occur, such as crepuscular rays (also called god rays), which were first simulated by Max [1986]. Some work focuses on the computation of single scattering in homogeneous media after light might have been refracted by scene objects, potentially causing caustics. Simple visibility tests of the light source are not sufficient; instead accounting for the geometric properties of the refracting object [Walter et al., 2009] or searching in higher dimensional line space [Sun et al., 2010] both give speed-ups that can reach interactivity for simple scene configurations. However, such scene setups with refracting objects are usually too difficult to handle in general real-time applications, which also display many other effects. Real-time applications therefore often ignore any surface-media interaction and single scattering is only caused by light that originates directly from the light source.

Under the assumption of homogeneous media and single scattering caused directly by a single light source with position $\mathbf{x}_l$ and direction $\vec{\omega}_l$, Equation 2.14 simplifies to:

$$L_{\mathrm{scat}}(\mathbf{x}, \vec{\omega}_{\mathrm{i}}) = \sigma_{\mathrm{t}}\, \rho \int_0^s e^{-t\sigma_{\mathrm{t}}}\, p(\vec{\omega}_{\mathrm{i}}, \vec{\omega}_l)\, L_{\mathrm{e}}(\mathbf{x}_t, \vec{\omega}_l)\, V(\mathbf{x}_t, \vec{\omega}_l)\, T_{\mathrm{r}}(\mathbf{x}_t, \mathbf{x}_l)\, \mathrm{d}t. \tag{2.19}$$

Here, we also added the very common assumptions that the phase function p only depends on the directions, and that the scattering albedo ρ is spatially constant and therefore lost its dependency on $\mathbf{x}$. As before, L_{e} denotes the unoccluded, incoming light due to the light source and V the corresponding visibility. Please also note that the focus on a single light source is without loss of generality, as light transport is linear and the contribution of multiple sources can be summed (see Section 2.1).

Just as was proposed for ambient occlusion, an additional common approximation is to factor out visibil-

ity [Baran et al., 2010; Chen et al., 2011; Wyman, 2011]. The equation then becomes:

$$L_{\text{scat}}(\mathbf{x}, \vec{\omega}_\text{i}) = \sigma_\text{t}\, \rho\, p(\vec{\omega}_\text{i}, \vec{\omega}_l)\, \bar{V}(\mathbf{x}, \vec{\omega}_\text{i}) \int_0^s e^{-t\sigma_\text{t}} L_\text{e}(\mathbf{x}_t, \vec{\omega}_l)\, T_\text{r}(\mathbf{x}_t, \mathbf{x}_l)\, \text{d}t$$
$$\bar{V}(\mathbf{x}, \vec{\omega}_\text{i}) := s^{-1} \int_0^s V(\mathbf{x}_t, \vec{\omega}_l)\, \text{d}t \tag{2.20}$$

The first integral can be computed analytically [Sun et al., 2005; Pegoraro and Parker, 2009] for directional, but also other types of light sources. The second integral represents average visibility of points on a view ray and cannot be solved analytically. The depth map of a G-buffer can be used to resolve the distance s that defines the segment in space for which visibility needs to be integrated. A naive solution directly uses a shadow map of the light source to resolve $V(\mathbf{x}_t, \vec{\omega}_l)$. Using a ray-marching process, the visibility $V(\mathbf{x}_t, \vec{\omega}_l)$ is tested for each step by a simple shadow-map lookup.

Interactive Approaches Early work simply blends parallel planes as a replacement of the ray-marching [Dobashi et al., 2002]. Toth and Umenhofer [2009] combine the naive ray-marching approach with an interleaved sampling to avoid the marching for all pixels. An efficient re-use of visibility data between pixels was proposed via a rectification of the shadow map that leads to a regular structure of the visibility queries of the view rays [Baran et al., 2010]. Intuitively, they build upon the idea that once an occluder was found along a light ray, the light ray does not contribute to any view ray past the occluder. Therefore, Baran et al. evaluate view rays (i.e., screen pixels) in the order of their distance to the epipole (the projection of the light source onto the image plane). Effectively, this corresponds to a sweep along the z-axis of the rectified shadow map as view rays and light rays are orthogonal to each other in the rectified space. This top-to-bottom evaluation can be asymptotically accelerated using a partial sum tree. While the method achieves good algorithmic complexity, the incremental evaluation order maps badly to modern GPUs. Alternatively, the rectification, coupled with a min-max acceleration structure enables the reduction of the number of marching steps that need to be employed [Chen et al., 2011]. As no ordering between pixels is required, this approach is also suitable for the execution on the GPU. Our approach in Chapter 4 will also rely on a rectified shadow map. A detailed explanation of the rectification and comparison to the work of Chen et al. [2011] is given there.

It is possible to reduce the number of steps further by deriving a marching interval via shadow volumes [Wyman and Ramsey, 2008], but such an approach is less useful for complex scenes. A more successful shadow volume application is their construction from a shadow map instead of the geometry. When rendering the shadow-volume faces with a special GPU shader, an accumulation in the framebuffer results in a convincing single-scattering approximation [Billeter et al., 2010]. Nonetheless, the performance highly depends on the shadow-map resolution and the method requires a high fill rate.

Performance-wise, voxelized shadow volumes are interesting [Wyman, 2011; Wyman and Dai, 2013], which also build upon an epipolar rectification. However, quality and speed of the methods depend on the chosen voxelization method, which is likely to be error-prone for high performance [Eisemann and Décoret, 2006] or costly [Eisemann and Décoret, 2008; Schwarz and Seidel, 2010] considering real-time applications. Storing the required high-resolution grid can also lead to extensive memory consumption as voxelized shadow volumes are not sparse. Additionally, the extension to area light sources [Wyman and Dai, 2013] uses imperfect shadow maps [Ritschel et al., 2008], which require a point-based representation of the scene. Therefore, both voxelized shadow-volume methods do not scale well to larger scenes. Further, the algorithm factors out visibility, which can lead to coarse results and textured light sources can only be handled via full ray marching.

Screen-space approaches also have a potential for high performance, but so far, they have been very approximate [Mitchell, 2007] and their performance strongly depends on the viewpoint and scene [Engelhardt

and Dachsbacher, 2010]. The latter approach performs an edge-aware filtering along epipolar lines based on the assumption that significant changes only occur at discontinuities. The performance of almost all solutions could potentially be increased (Wyman [2011] makes use of this algorithm). However, the assumption can fail and especially for detailed scenes, the edge detection can make many samples necessary, which might not pay off.

3

Bent Normals and Cones in Screen Space

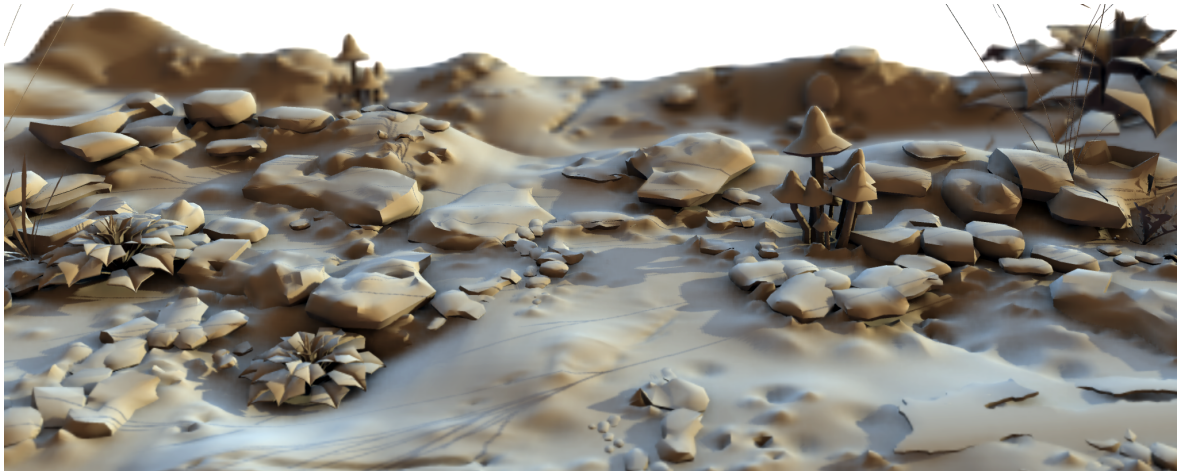


Figure 3.1: Possible game scenario with lighting using bent cones: 2048×1024 pixels, 60.0 fps, including direct illumination from a distant light source and a simple depth-of-field effect on a Nvidia GF 560Ti. Environment mapping produces natural illumination, while bent cones produce colored shadows.

In this chapter, we propose a novel method for rendering environmental illumination in real time based on a screen-space solution of ambient occlusion. Ambient occlusion (AO) is a non-physically-based approximation of environmental lighting and gives an impression of global illumination (Figure 3.1). It also acts as a visual cue for object-object relationships. AO achieves high performance by computing an average occlusion that is used to modulate surface shading without respecting the directionality of light. Consequently, the approximation is not always acceptable, for example when used in conjunction with high-frequency environment illumination.

Landis [2002] addressed the directionality issue by introducing so-called bent normals. While AO stores an average occlusion, bent normals are modified normals bent according to an estimate of the direction that is most unobstructed, i.e., the average unblocked direction (see Section 2.3.4). Bent normals can then be used to compute an illumination response that is closer to an actual sampling of the environment map. A useful property of bent normals is that they can usually be easily integrated into rendering engines. Basically, they can directly be used for shading just like standard normals, but result in an improved lighting.

3.1 Contributions

One popular incarnation of AO is its implementation in screen space (screen-space ambient occlusion, SSAO) [Mittring, 2007; Shanmugam and Arikan, 2007; Bavoil et al., 2008; Ritschel et al., 2009b; Loos and Sloan, 2010; Huang et al., 2011; Herholz et al., 2012; Timonen, 2013]. Our idea is to keep the simplicity of SSAO by relying on a screen-space solution, but to add the computation of bent normals. Furthermore, we describe bent cones that capture the distribution of unoccluded directions by storing their average and variance. We show that bent normals and cones are easy to compute and enable higher quality lighting results. As our technique extends existing solutions of screen-space ambient occlusion, our solution is easy to integrate into existing rendering pipelines.

In short, our contributions are:

- Extending existing screen-space ambient occlusion techniques to compute bent normals;
- Combining bent normals with the directional variation of unoccluded directions to bent cones.

3.2 Screen-space Ambient Occlusion

As introduced in Section 2.3.4, ambient occlusion is defined as the fraction of unoccluded directions in the upper hemisphere of directions and can be estimated by a Monte-Carlo approach:

$$AO_{\mathrm{MC}}(\mathbf{x}) := \frac{1}{|R|} \sum_{\vec{\omega} \in R} V(\mathbf{x}, \vec{\omega}) \approx AO(\mathbf{x}). \tag{3.1}$$

Screen-space ambient occlusion uses the depth buffer of the camera as input to cheaply approximate visibility tests. In its most basic form [Mittring, 2007], screen-space ambient occlusion is computed for a pixel i by comparing its camera space location $\mathbf{x}_i$ with other pixel's camera space position $\mathbf{x}_j$ in a pixel neighborhood P_i (see Figure 3.2):

$$AO_{\mathrm{SS}}(i) := \frac{1}{|P_i|} \sum_{j \in P_i} d(\Delta_{ij}) \approx AO_{\mathrm{MC}}(\mathbf{x}_i), \tag{3.2}$$

where $\Delta_{ij} := \mathbf{x}_j - \mathbf{x}_i$ denotes the vector from the surface point $\mathbf{x}_i$ to another sample $\mathbf{x}_j$. One possible implementation of the occlusion function $d(\Delta)$ is $d_{\mathrm{z}}(\Delta)$, which only tests the depth value (z-coordinate) of Δ:

$$d_{\mathrm{z}}(\Delta) := \begin{cases} 1 & \text{if } \Delta.\mathrm{z} > 0 \\ 0 & \text{else} \end{cases}. \tag{3.3}$$

$d_{\mathrm{z}}(\Delta)$ has two disadvantages: first, it ignores the local orientation of the surface, $\vec{\mathbf{n}}$, which causes self-occlusion; second, it does not account for outliers that should not cast shadows, i.e., geometry that is far in world space, but close in screen space. Both effects can be easily addressed by including the normal at the i-th pixel as well as using a distance-based falloff function f [Shanmugam and Arikan, 2007; Ritschel et al., 2009b; Loos and Sloan, 2010; McGuire et al., 2011]:

$$d_{\vec{\mathbf{n}}}(\Delta, \vec{\mathbf{n}}) := f(||\Delta||) \cdot \begin{cases} 1 & \text{if } \langle \Delta \cdot \vec{\mathbf{n}} \rangle > 0 \\ 0 & \text{else} \end{cases}. \tag{3.4}$$

It needs to be defined which pixels P_i may cause occlusion and therefore need to be compared to the current pixel i. They are usually chosen to be neighboring pixels in screen space [Mittring, 2007], or the projection of a set of sample points near to $\mathbf{x}_i$ in world space [Shanmugam and Arikan, 2007]. Instead of gathering occlusion from a

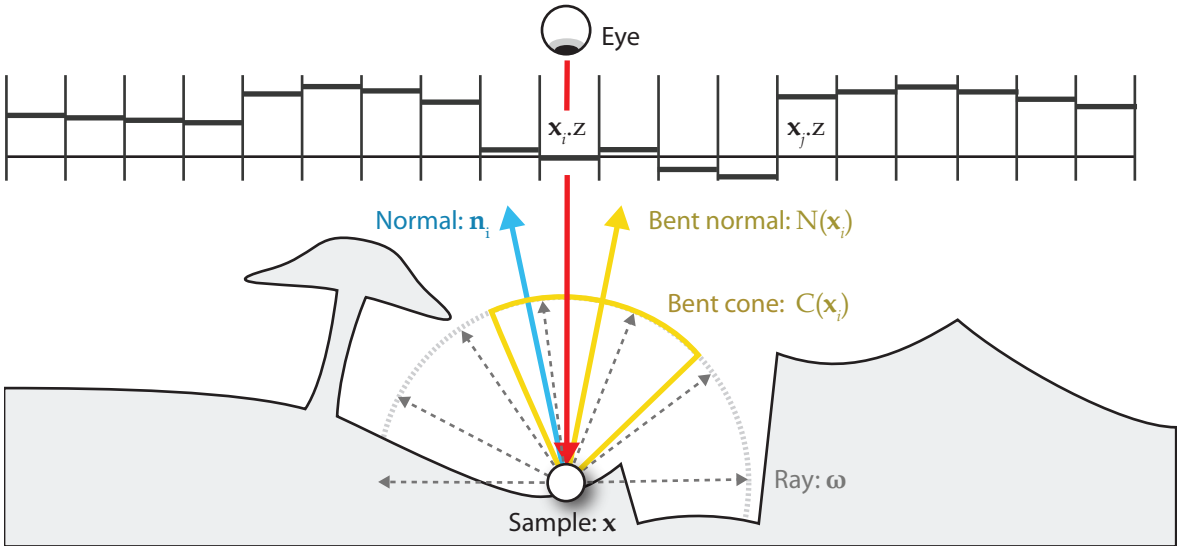


Figure 3.2: Illustration of our main variables. AO, bent normals and bent cones in flatland.

neighborhood P, Shanmugam and Arikan [Shanmugam and Arikan, 2007], as well as McGuire [McGuire, 2010] use a splatting approach to distribute AO from surfaces to pixels.

The underlying assumption of SSAO is, that summing occlusion of "nearby" occluders approximates true visibility. However, visibility is a nonlinear effect: two occluders behind each other in one direction do not cast a shadow twice. Hence, it is better to find the correct occlusion for a set of directions via ray marching in the depth buffer [Oat and Sander, 2007]. Alternatively, one can also compute a horizon angle [Max, 1988; Bavoil et al., 2008]. As an in-between solution, others [Szirmay-Kalos et al., 2010; Loos and Sloan, 2010] considered the free volume over a height field of depth values inside a sphere around a pixel as a better approximation.

3.3 Our Technique

In this section, we will describe our approach for interactive ambient occlusion and improved environmental illumination based on screen-space bent normals. We will first introduce their computation (Section 3.3.1), and then generalize them to bent cones (Section 3.3.2).

3.3.1 Bent Normals

We describe the basic implementation along the lines of the original Crytek SSAO [Mittring, 2007]. There, the original continuous AO defined in world space (Equation 2.6), was solved in a discrete way in screen space (Equation 3.2). We will apply this idea to the continuous world-space bent normals (Equation 2.3.4), and compute them in discrete screen space. To this end, we aggregate unoccluded directions Δ_{ij}:

$$\vec{N}_{\mathrm{SS}}(i) := \Big(\sum_{j \in P_i} d(\Delta_{ij}) \Big)^{-1} \sum_{j \in P_i} \frac{\Delta_{ij}}{||\Delta_{ij}||} d(\Delta_{ij}) \approx \vec{N}(\mathbf{x}_i). \tag{3.5}$$

The basic principle behind our approach is that, when computing AO in screen space, the direction Δ_{ij} and its visibility d are known and can be used to accumulate a mean direction that defines the bent normal. Our technique is mostly orthogonal to the type of SSAO used, allowing for different implementations of d. The bent normal can then be used for lighting (Section 3.3.3) in place of the original normal.

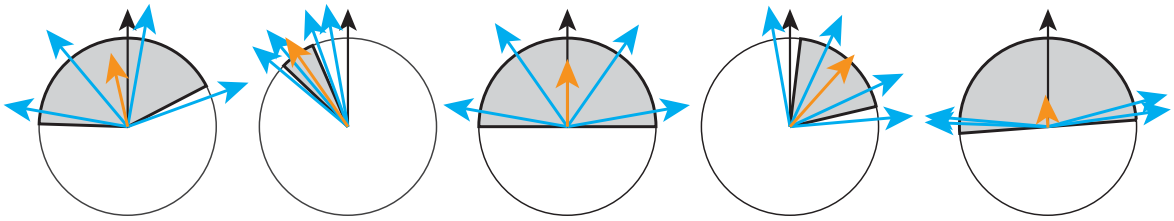


Figure 3.3: Resulting bent normal and cone of different visibility configurations. Blue arrows show unblocked directions within the hemisphere centered around the normal (black). Bent normals form the average direction (orange). The length of the bent normal describes the variance of the unblocked directions and is used to calculate the angle of the bent cone with the bent normal as center (darkened sector of circle).

3.3.2 Bent Cones

Bent cones are bent normals augmented by an *angle* (Figure 3.2). As the mean of the unoccluded directions gives the bent normal, the variance defines the angle. However, directions require the use of directional statistics instead of linear statistics in Euclidean spaces. To this end, we use an approach similar to the computation of variance in von Mises-Fisher (vMF) distributions [Mardia and Jupp, 2000]. There, variance is approximated from the length of the non-normalized bent normal as computed in Equation 2.3.4. While vMF distributions are defined for spheres, we estimate a distribution on a hemisphere. This leads to a simple estimation of angle, corresponding to the variance of the unoccluded directions as:

$$C(i) := (1 - \max(0, 2\,||\vec{N}_{SS}(i)|| - 1))\,\frac{\pi}{2}. \tag{3.6}$$

Hereby, the *variance* of the unoccluded direction is directly reflected by the length of the bent normal. The max-function ensures that the bent cones of unoccluded points exactly cover the hemisphere.

Bent normal and bent cone define a spherical cap of visibility. Figure 3.3 shows possible unblocked directions and the resulting bent normal and cone. However, instead of using the cone as a visibility approximation, we only restrict the directions from which we gather light. Hence, we use this cap in combination with shading methods such as preconvolved environment maps [Heidrich and Seidel, 1999] or irradiance volumes [Greger et al., 1998] that compute the incoming light inside a spherical cap. Note, that the clamping (max function) ensures that the length of the bent normal of a completely unoccluded surface point is mapped to a cone that covers the entire hemisphere.

We only use the bent cone to limit the directions from which we gather light. The cone does not describe a concrete visibility approximation. We still use AO to estimate the overall visibility. One can think of the cone as describing the illumination color whereas the AO controls brightness. This also allows for inaccuracies in the estimation of the actual visibility configuration. Using a cone with an angle of 90 degrees means a fall-back to illumination with bent normals only, hence, quality is not decreased compared to lighting with AO only. Next, we describe how to use the cones to evaluate the incoming illumination.

3.3.3 Preconvolved lighting

For shading using bent cones, we use preconvolved environment maps [Heidrich and Seidel, 1999] which are indexed by the direction and angle of the cone. The same concept applies to irradiance volumes [Greger et al., 1998]. Such precomputations are done at application startup or once per frame, and avoid computations at shading time. The convolutions involved are known to give very smooth results without noticeable noise, which

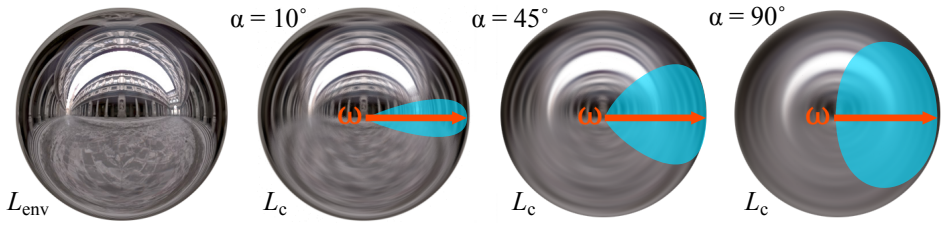


Figure 3.4: Preconvolution of distant lighting (left) into a series of products of light, and visibility of cones with varying angle α (left to right: 10, 45, and 90 degrees).

fit well to the perceptual requirements of games.

Preconvolution of distant illumination computes a directional function $L_p(\vec{\omega}_o)$. For every outgoing direction $\vec{\omega}_o$, it convolves the distant illumination $L_{env}(\vec{\omega}_i)$ and the geometric term (cosine of incoming light and normal):

$$L_p(\vec{\omega}_o) := \frac{1}{\pi} \int_{\Omega^+} L_{env}(\vec{\omega}_i) \, \langle \vec{\omega}_o \cdot \vec{\omega}_i \rangle \, \mathrm{d}\vec{\omega}_i. \tag{3.7}$$

To query this convolved environment map, the bent normal is used in place of $\mathbf{n}$: the computation reduces to a lookup in the preconvolved environment map and a multiplication with SSAO, which accounts for visibility:

$$L_o(\mathbf{x}) = AO(\mathbf{x}) \, L_p(\vec{N}(\mathbf{x})). \tag{3.8}$$

However, to be correct, preconvolution has to assume no shadows or at least, that visibility can approximately be moved outside the integral. In practice, this means the lighting result is just multiplied with a shadow term (AO). We propose to include visibility inside the preconvolution leading to a tri-variate function

$$\begin{aligned} L_c(\vec{\omega}_o, \alpha) &:= t \frac{1}{\pi} \int_{\Omega^+} L_{env}(\vec{\omega}_i) \bar{V}(\vec{\omega}_i, \vec{\omega}_o, \alpha) \, \langle \vec{\omega}_o \cdot \vec{\omega}_i \rangle \, \mathrm{d}\vec{\omega}_i, \\ t &:= (1 - \cos(\alpha))^{-1} \end{aligned} \tag{3.9}$$

that stores the outgoing radiance for a bent cone in direction $\vec{\omega}_o$ with angle α (Figure 3.4). The function $\bar{V}$ returns one if $\vec{\omega}_i$ and $\vec{\omega}_o$ form an angle smaller than α and zero otherwise. Note, that by doing so, with increasing α the preconvolved values get larger. As we do not use the bent cone as visibility approximation (cf. Section 3.3.2), we introduce the *normalization* term t. Thus, we get equal results L_c independent of α if L_{env} is constant. The mean direction $\vec{N}_{SS}(i)$ and angle $C(i)$ of the bent cone replace $\vec{\omega}_o$ and α to look-up a convolved environment map.

3.3.4 Geometric Term

We use a heuristic to apply the geometric term in our bent cones. It needs to be a part of the preconvolution because the incoming radiance for a specific direction is only known at this time. Correctly integrating the geometric term would be five-dimensional: 2D for the mean direction (bent normal), 1D for the angle of the cone, and 2D for the surface normal. We can approximate it by using only the bent normal and cone angle as:

$$\langle \vec{\omega} \cdot \vec{\mathbf{n}} \rangle \approx \left\langle \vec{\omega} \cdot \vec{N}(\mathbf{x}) \right\rangle \left\langle \vec{N}(\mathbf{x}) \cdot \vec{\mathbf{n}} \right\rangle. \tag{3.10}$$

If $\left\langle \vec{N}(\mathbf{x}) \cdot \vec{\mathbf{n}} \right\rangle \approx 1$, $\left\langle \vec{\omega} \cdot \vec{N}(\mathbf{x}) \right\rangle$ approximates the correct geometric term very well, while $\left\langle \vec{N}(\mathbf{x}) \cdot \vec{\mathbf{n}} \right\rangle$ vanishes by definition. If $\vec{N}(\mathbf{x})$ and $\vec{\mathbf{n}}$ diverge, the heuristic returns near correct results for light directions close to the bent normal ($\left\langle \vec{\omega} \cdot \vec{N}(\mathbf{x}) \right\rangle \approx 1$), as $\langle \vec{\omega} \cdot \vec{\mathbf{n}} \rangle \approx \left\langle \vec{N}(\mathbf{x}) \cdot \vec{\mathbf{n}} \right\rangle$. Additionally, the angle of the bent cone becomes smaller, such that the error for light directions within the cone remains small for all possible cones (cf. Figure 3.5). The

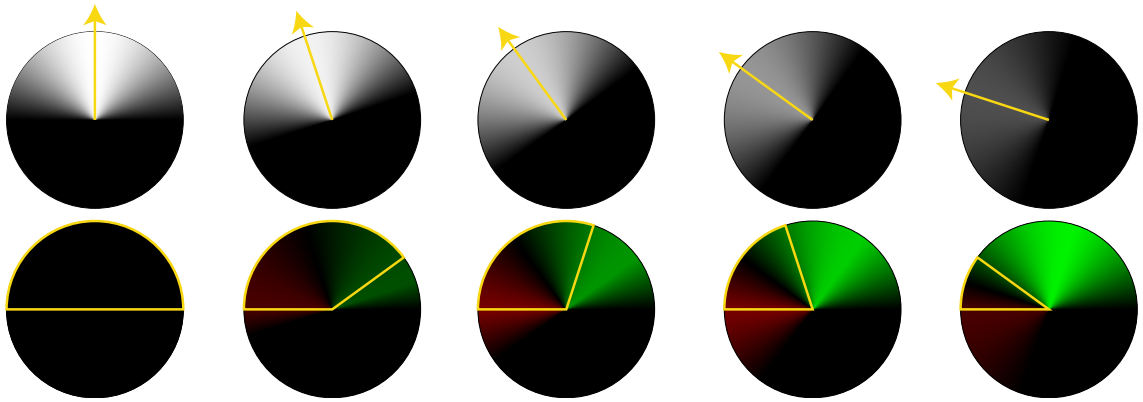


Figure 3.5: 2D examples of our approximation of the geometric term. The top rows shows the evaluated term, the bottom shows the error (compared to a normal with direction upwards; left). Directions with overestimation are red, while underestimation is green compared to the correct geometric term. The bent normal is increasingly rotated in the images to the right. The heuristic ensures correct weighting for unoccluded surfaces (bent normal and normal are equal; left). The more normal and bent normal diverge (images to the right), the greater the error becomes. However, at the same time, the angle of the bent cone becomes smaller and the error within the cone remains low. Note that the bent cone shown here is only exemplary as the angle depends on the distribution of the unoccluded directions.

heuristic regards the original normal and should be preferred to only using $\left\langle \vec{\omega} \cdot \vec{N}(\mathbf{x}) \right\rangle$. Further, it avoids a 5D preconvolution and gives correct results for unoccluded points (where $\vec{\mathbf{n}} = \vec{N}(\mathbf{x})$).

3.4 Implementation

Variants For our experiments, we combined the approach with several methods: `Crytek2D` [Mittring, 2007] uses a 2D sampling pattern to distribute samples in screen space. For higher quality, we perform jittering and reject samples outside the unit disc. `Crytek3D` [Mittring, 2009] uses a 3D sampling pattern to distribute samples in space that are projected to screen space. We generate samples on the unit hemisphere, optionally add ray-marching steps to increase the visibility-test quality, and apply randomized offsets in each direction to turn aliasing into more eye-pleasing noise. HBAO [Bavoil et al., 2008] uses random 3D directions orthogonal to $\vec{\mathbf{n}}$ that are marched to find the highest horizon angle. The samples are distributed on the unit disc and transformed according to $\vec{\mathbf{n}}$. Again, randomized offsets are used. We rely on OpenGL 3 and store all sample patterns in uniform variables. If a large pattern or many samples are required, we store the data in a texture. Listing 3.1 shows pseudo-code for how AO and bent normals are computed in a pixel / fragment shader.

Interleaved Sampling & Filtering Sub-sampling with interleaved patterns [Keller and Heidrich, 2001] allows for using only a low number of samples per pixel, which is important for performance. Here, using a randomization pattern of size $m \times m$ applies different sample sets for neighboring pixels, which prevents banding artifacts and turns under-sampling into noise. To blur the noisy AO and bent normals, a geometry-aware blur [Laine et al., 2007] regarding position and normal discontinuities is used. When applying an 16×16 interleaved-sampling pattern and using 2×2 super sampling for anti aliasing, a single sample is enough for AO. Bent cones require at least four samples.

```
void main() {
   // get point properties at pixel i
   vec3 positionX = backproject(depthTexture(pixelCoordinateI),
       inverseViewProjectionMatrix);
   vec3 normalX = normalTexture(pixelCoordinateI);
   // get ONB to transform samples
   mat3 orthoNormalBasis = computeONB(normalX);
   // select samples from interleaved-sampling pattern
   int patternOffset = getPatternOffset(pixelCoordinateI);

   float ao = 0.0;
   int validAODirectionCount = 0;
   vec3 bentNormal = vec3(0.0);
   float unoccludedDirections = 0.0;

   for(int index=0; index<sampleCount; ++index) {
      vec3 sampleDirection = orthoNormalBasis * getDirection(index, patternOffset);
      bool isOutlier = false;
      // use float instead of bool and
      // apply a fall-off function to get smooth AO
      float visibility = 1.0;
      // this function tests for occlusion in SS.
      // depending on the actual technique and sample distribution,
      // the implementation of this function varies
      checkSSVisibilityWithRayMarchingSmooth(sampleDirection, maxOccluderDistance,
          depthTexture, inverseViewProjectionMatrix, positionX, normalX,
          rayMarchingSteps, rayMarchingStartOffset, visibility, isOutlier);

      // we have insufficient information in SS
      // here, we simply ignore samples, which cannot be handled properly
      if(!isOutlier) {
         validAODirectionCount++;
         ao += visibility;
      }

      // for bent normals, we assume,
      // that outlier occluders are NOT real occluders!
      // sum up unoccluded directions
      // directions may be partially visible
      // => weight accordingly
      bentNormal += normalize(sampleDirection) * visibility;
      unoccludedDirections += visibility;
   }

   ao /= float(validAODirectionCount);
   bentNormal /= unoccludedDirections;
}
```

Listing 3.1: Fragment shader pseudo-code for screen-space bent cones.

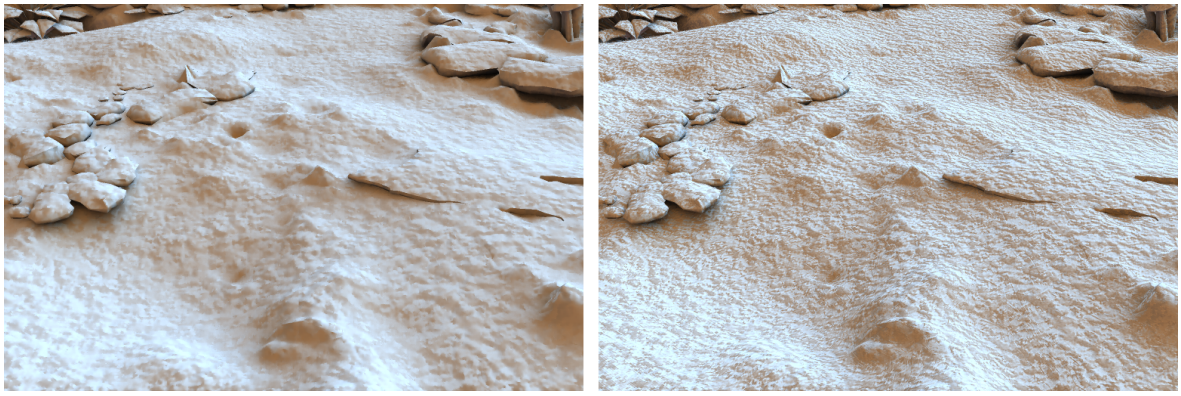


Figure 3.6: Filtering of change of normals. Fine geometric details, e.g., due to bump maps are lost when directly filtering bent normals (left). When filtering the change of normals instead, details are preserved (right).

Instead of interpolating the bent normal directly, we compute the difference between normal and bent normal and blur this difference. The result is added to the original high-frequency normals. Hereby, the details in the normal field itself are preserved as shown in Figure 3.6 and the bent information is propagated.

Preconvolved Lighting For preconvolved environment maps, we store L_p as a floating-point cube texture. Preconvolution including cone angle variation is tri-variate and can be stored most efficiently using the cube-map-array extension of OpenGL. This extensions stores an array of cube maps that can be accessed with a direction and an index. We discretize α to 8 levels and apply linear filtering between the levels, which has shown to be sufficient. Due to the convolution, low resolution cube maps provide sufficient quality, resulting in $\sim$7 MB of storage for a typical 64×64 cube map-array with 8 layers. More efficiently, in the context of glossy reflections Kautz and McCool [2000] propose to store the third dimension in cube map MIP levels. This could even allow for a fully dynamic convolution of the environment map at reduced quality, but also reduced storage costs.

Practical Considerations Bent normal and cones integrate very well in an existing deferred shading rendering pipeline. The bent normal simply replaces the normal without requiring special handling. Further, the bent normal can be scaled with AO. Using the scaled bent normal for shading, AO is included automatically, also avoiding further storage requirements. For bent cones, no extra storage is required besides the bent normal and AO. However, a special preconvolution step is required for a number of possible cone angles.

3.5 Results

In this section, we present our results that increase accuracy while adding only a small performance penalty compared to SSAO. For the following images, we used the 3D sampling pattern with three ray-marching steps, which best approximates the integrals of Equations 2.6 and 2.3.4.

A performance breakdown of Figure 3.1 at resolution 2048×1024 on an Nvidia Geforce 560Ti is as follows: 4.2 ms for AO and bent normals, 4.7 ms for geometry-aware blurring, 1.5 ms for the deferred shading buffer, and 1.6 ms for direct light with PCF shadows. The overhead for bent normals compared to AO only using the same number of samples is 7 % for the computation and 25 % for the blur, in total less than 11 %. We still see some space for engineering improvements, depending on the actual quality and performance constraints in real applications. For example, using data packing and 8 bit for bent normals may especially improve the

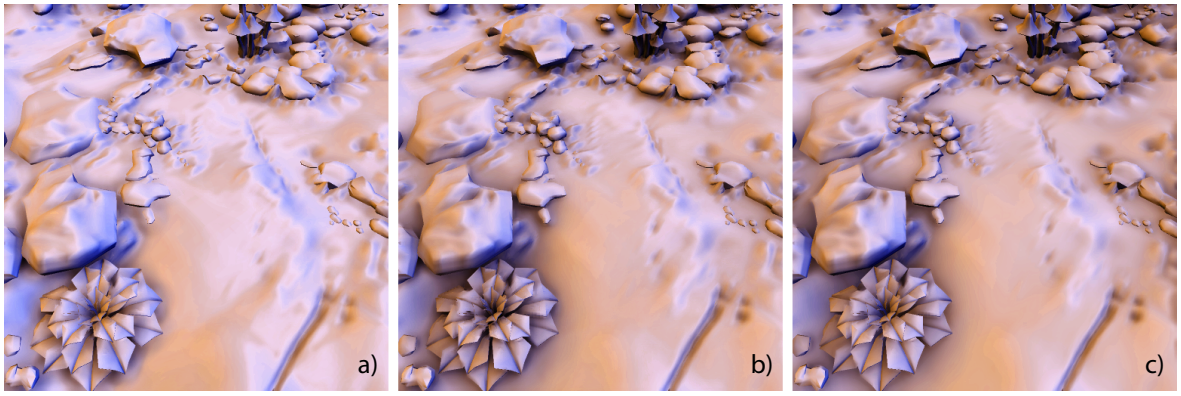


Figure 3.7: Our approach is largely orthogonal to the particular SSAO implementation used: two-dimensional sampling (`Crytek2D`, *(a)*), three-dimensional sampling (`Crytek3D`, *(b)*), horizon-based ray-marching (HBAO, *(c)*).

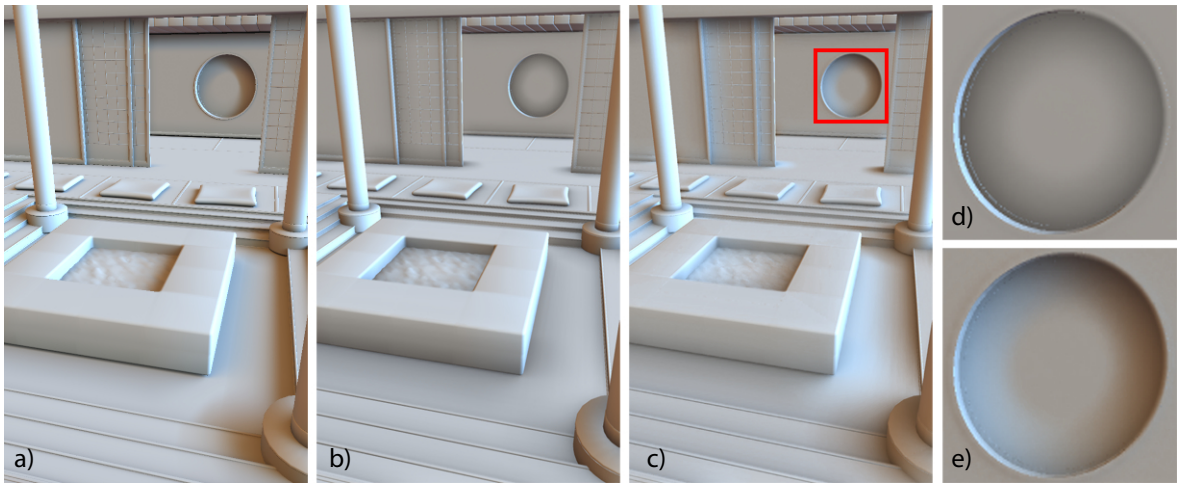


Figure 3.8: Lighting using ray-tracing *(a)*, SSAO *(b)*, and our screen-space bent normals *(c)*. Details have more articulated lighting when AO *(d)* is enhanced by our bent normals *(e)*.

performance of the filtering.

Lighting using our bent normals is closer to a reference solution than AO alone (Figure 3.8). It can be seen how AO decouples lighting and visibility, which leads to grey shadows that are perceived as a change of reflectance, rather than an effect of lighting. Our screen-space bent normals are similar to real bent normals (Figure 3.9) which leads to only a small difference between lighting using accurate bent normals and our screen-space bent normals. Additionally, it shows that testing visibility in screen-space is sufficient in most cases. The reference solutions were created using path-tracing using several hundred samples.

Bent cones improve on bent normals as the cones do not necessarily gather light from all directions in the upper hemisphere of a point (see Figure 3.10). Ritschel et al. [2009b] (screen-space directional occlusion, SSDO) regard the directionality of light by applying a brute-force sampling for unblocked directions. This sampling implies an additional texture lookup, which results in an overhead of about 10 % in execution time. Even more importantly, noise appears if the sample count is not increased. In our tests, we required 16 to 24 samples per-pixel in combination with interleaved sampling to avoid visible artifacts. Our bent cones produce smooth results with 8 samples with similar quality. This low sample count results in speedups by a factor of two.

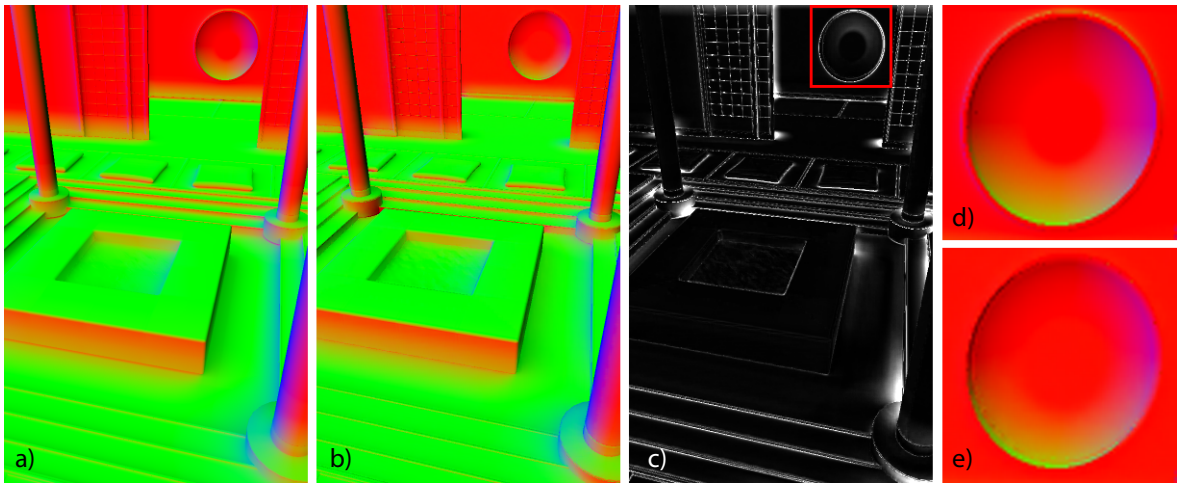


Figure 3.9: Ray-traced bent-normals *(a)*, our screen-space bent normals *(b)*, and the 8× angle difference *(c)*. Ray-traced bent normal details *(d)* are similar to our approximation *(e)*.

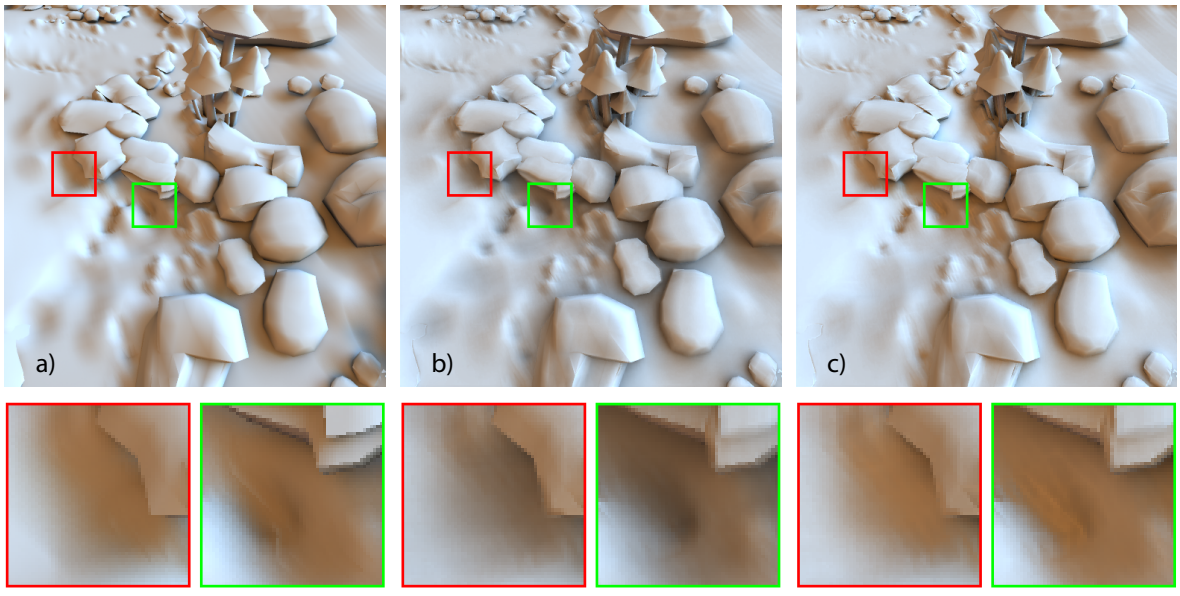


Figure 3.10: Left to right: lighting using ray-tracing *(a)*, SS bent normals *(b)*, and SS bent cones *(c)*. Applying cones, shadows are more colored and appear similar to the Monte Carlo reference, because only visible light is gathered (cf. insets).

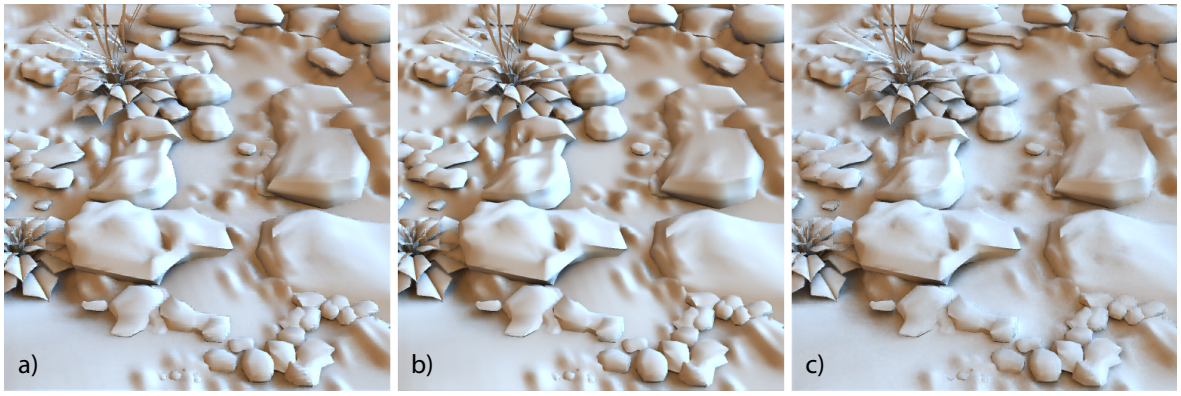


Figure 3.11: Environment map importance sampling (IS) *(a)*, SSDO [Ritschel et al., 2009b] *(b)*, and SS bent cones *(c)*. IS requires 32 samples per pixel (18.8 ms), SSDO 16 samples (12.5 ms), and bent cones only 8 samples (6.0 ms) to achieve the results (1024×1024 resolution; 2×2 super-sampling). All techniques use three ray-marching steps per sample for the SS visibility tests, an 8×8 interleaved-sampling pattern, and an according geometry-aware blur.

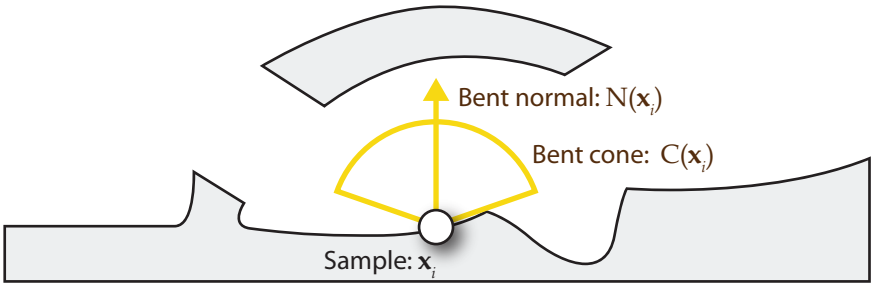


Figure 3.12: Bent normals and cones assume blockers to form a simple horizon. In case of more complex occlusion, such as shown here, the direction can diverge.

SSAO returns good results with such a low number of samples and the cone does not need to be very accurate (Figure 3.11). Additionally, we compared our technique to importance sampling of the environment map (IS), which requires at least 32 samples to achieve equal quality. Note, in contrast to sampling-based techniques, bent cones are not able to handle high-frequency illumination. Thus, we used low-frequency illumination changes only to allow for a fair comparison. Otherwise, bent cones would simply blur the high-frequencies, SSDO requires many more samples (up to hundreds), and IS at least 64 samples.

3.6 Discussion and Conclusion

Screen-space bent normals improve accuracy of shading without imposing much additional computation costs. The bent cones further improve on the directionality of lighting. Bent cones can limit the spherical cap depending on the variance of the unoccluded directions. The success clearly depends on whether the actual visibility configuration fits a cone (Figure 3.12). The cone is chosen such that it mostly includes unblocked directions, possibly overestimating the overall visibility. The overestimation is not a problem as we still rely on AO to account for the actual overall visibility. As a worst case, bent cones fall back to lighting with bent normals. While screen-space directional occlusion [Ritschel et al., 2009b] achieves similar results, it requires the evaluation of shading for every sample. This is closer to screen-space Monte-Carlo rendering involving many samples, but

does not agree with the original goal of AO to decouple shading and visibility as much as possible. In contrast, our approach keeps shading and visibility separated, leading to a significant speed-up.

Bent normals and cones are most suitable for lighting techniques that compute lighting within a spherical cap, such as environmental lighting with light coming from many directions. They are less useful for direct illumination of very directed light such as from the common directional, spot, or point light sources. Consequently, bent normals and cones are not able to handle high-frequency illumination changes.

We showed that our extension is mostly orthogonal to the particular SSAO implementation used [Mittring, 2007; Bavoil et al., 2008], and thus can be integrated into existing implementations easily. It is intended to be used as part of a deferred shading pipeline, most interestingly for real-time applications. By performing the visibility test in screen-space, only an incomplete scene representation is available, sharing the screen-space limitations with previous SSAO techniques. However, the screen-space visibility test could be replaced with a more accurate test, such as against a voxel representation of the scene.

3.6.1 Future Work

We used statistics to represent directional-dependent effects of the local visibility of a surface point. Gaussians may offer an interesting alternative to approximate visibility [Green et al., 2006, 2007]. Later work [Herholz et al., 2012] showed that spherical harmonics are useful as they allow for the combination with irradiance volumes [Greger et al., 1998] for precomputed global illumination. Also, our technique relies on precomputation to convolve environmental illumination, but we believe that an approximate real-time convolution is possible as long as scene materials are kept smooth.

Our bent normals and cones build upon screen-space ambient occlusion. While many advances have been proposed, we want to explicitly mention the recent technique of Hendrickx et al. [2015] that employs visibility filtering similar to our filter-based scattering that will be discussed in the next chapter. While their statistical model allows for a high sample count at very high performance, it prevents the computation of directional effects. We believe that there is interesting future work that could combine such an efficient occlusion estimation with directionally-dependent lighting, as provided by our technique.

Finally, many screen-space techniques rely on some sort of sub-sampling, in our case interleaved sampling, which requires a blurring to avoid high-frequency noise. We currently rely on simple geometric properties such as position and normal to perform a cross-bilateral filtering. We believe that recent high-quality denoising techniques [Zwicker et al., 2015] could be simplified and adapted to high-performance rendering.

4

Real-time Single-Scattering Rendering of Homogeneous Media

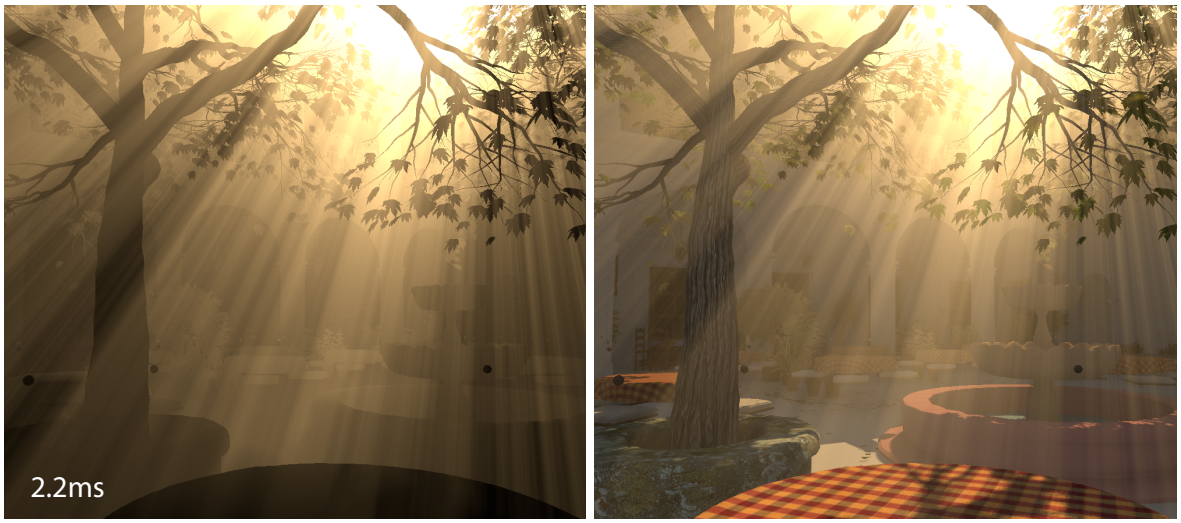


Figure 4.1: Our method needs only 2.2 ms for single scattering using a 1024^2 shadow map and a $1920\times$ 1080 full-HD screen resolution (a cropped version is shown). It has close to constant cost per screen pixel, achieving speedups of order of a magnitude for similar quality compared to state-of-the-art methods whose performance is scene-dependent [Chen et al., 2011].

In this chapter, we focus on the computation of real-time single scattering in homogeneous participating media due to direct illumination. Participating media influence the appearance of a scene drastically. A careful simulation can not only add realism, but also spatial cues regarding scene layout, similar to ambient occlusion (cf. Figure 3.1). When light interacts with thin participating media, crepuscular rays (also called god rays or light shafts) appear, which are a strongly visible phenomenon and often serve even artistic purposes (that we will investigate in Chapter 6). In contrast to light-surface interaction, which is evaluated at surface points, participating media require solving for the in-scattering of light along entire view rays as introduced in Section 2.4. Commonly, real-time methods limit the participating media to be homogeneous and approximate the light interaction with it to a single-scattering model to be able to handle fully dynamic scenes. This single light-medium interaction event enforces that the light is redirected towards the camera along the view ray. This methodology is similar to direct illumination for surfaces. Although this simplifying assumption led to several methods that are faster than an accurate multiple-scattering model, their performance is usually scene-dependent

and they struggle when employed in real-time applications. The naive solution is to launch a ray marching per screen pixel along the view ray. At each step along the ray, a lookup in a shadow map is performed to determine if light can be in-scattered at this location. To accelerate the algorithm, a rectified shadow map can be used, in which view rays always stay within a single row of the shadow map. Using a tree structure [Baran et al., 2010] or a GPU-friendly min-max hierarchy [Chen et al., 2011], longer marching distances can be used employing only a single shadow test. Nonetheless, for complex scenes, these structures can become relatively inefficient, making many steps along the view ray necessary.

4.1 Contributions

We show how to compute single scattering with lower complexity than previous methods for fully dynamic scenes, practically causing a speed-up of up to an order of magnitude. The key insight of our solution is to replace the ray marching with a filtering process, in the spirit of Annen et al. [2007]. This *prefiltering*, independent of actual view rays of the camera, is applied to a rectified shadow map and results in a simple computation to evaluate the scattering contribution for each screen pixel. The benefit is that we compute visibility at once for screen pixels that lie on a so-called epipolar line. Thus, the algorithm is mostly scene independent, as the heavy work is performed in image space, independent of the scene or the content of the shadow map. The rendering of the shadow map is the only geometry processing, but this is required for surface shadows as well and does not cause a computational overhead. The actual scattering evaluation per pixel is a single dot product, which is also independent of the scene complexity and cheap. Therefore, our method is mostly independent of scene complexity, resolution, and viewpoint, using a constant amount of operations per pixel. In consequence, even high-resolution renderings can be achieved in milliseconds. Additionally, we present a novel rectification method for the shadow map, which results in a linear matrix transform, hereby, avoiding pixel-shader scattering operations as used in previous reprojection schemes [Chen et al., 2011].

Summarized, our contributions are as follows:

- A single-scattering method using prefiltering, leading to reduced computational complexity;
- A single-step shadow-map rectification using a matrix multiplication;
- Efficient GPU-oriented filter strategies.

4.2 Background

In this section, we explain the scattering model and the rectified shadow map. This background knowledge makes it easier to follow the core of our algorithm. In particular, a good understanding of the properties of rectified shadow maps is helpful. We will introduce our own rectification method later in this chapter.

4.2.1 Scattering Model

We are concerned with volumetric single scattering, for which all contributing scattering events for a given pixel happen along its corresponding view ray from the camera. We further assume a homogeneous medium, which allows us to compute medium transmittance and scattering probability analytically. In Section 2.4.4, we derived the resulting equation that needs to be solved. In this work, we make the additional simplifying assumption of a directional light source, i.e., $L_e(\mathbf{x}, \vec{\omega}) := L_e(\vec{\omega})$, and so we drop the spatial dependency. We will discuss other light sources in Section 4.3.2. Given a single directional light, the emission term of the light source and the

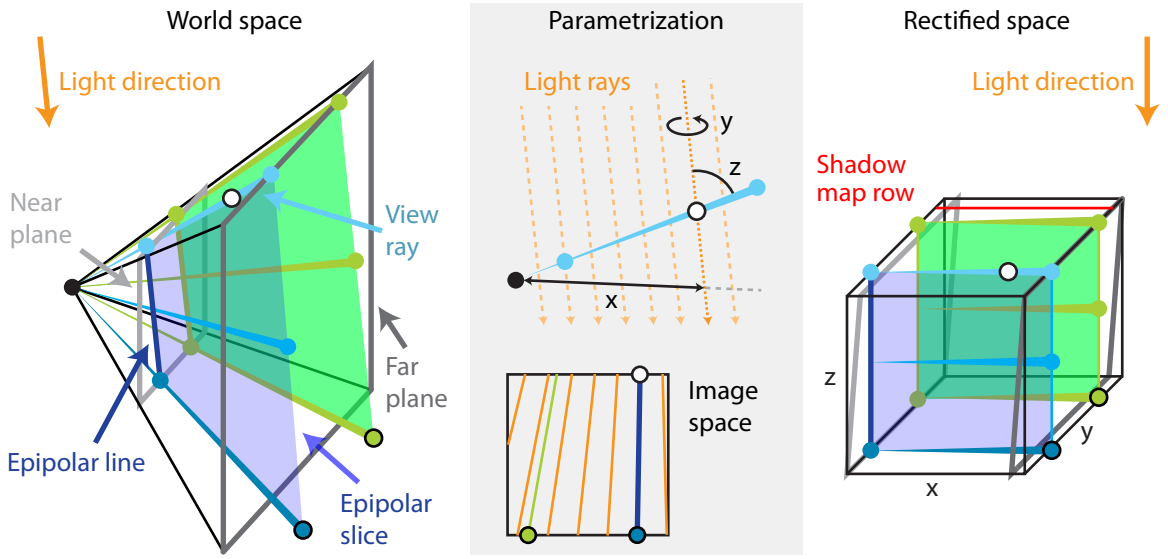


Figure 4.2: A rectification transforms camera view rays in such a way that they become parallel to each other; View rays in the same epipolar slice share the same y coordinate and the light direction is aligned with the z axis.

phase function can be removed from the integral of Equation 2.20, leading to:

$$L_{\text{scat}}(\mathbf{x}, \vec{\omega}_{\text{i}}) = \rho\ \sigma_{\text{t}}\ p(\vec{\omega}_{\text{i}}, \vec{\omega}_l)\ L_{\text{e}}(\vec{\omega}_l) \int_0^s T_{\text{r}}(\mathbf{x}, \mathbf{x}_t)\ V(\mathbf{x}_t, \vec{\omega}_l)\ \mathrm{d}t. \tag{4.1}$$

Please note that we also omitted the transmittance term from the scatter point to the light source. As the medium is assumed to be homogeneous, i.e., spatially unbounded, transmittance towards the light source at infinity would be zero. Throughout this chapter we will use the shorthand of $C := \rho\ \sigma_{\text{t}}\ p(\vec{\omega}_{\text{i}}, \vec{\omega}_l)\ L_{\text{e}}(\vec{\omega}_l)$ to denote all ray-constant (i.e., independent of $\mathbf{x}_t$) factors. As mentioned before, a common approximation to Equation 4.1 is to factor out visibility, leading to:

$$\begin{aligned} L_{\text{scat}}(\mathbf{x}, \vec{\omega}_{\text{i}}) &\approx C\ \overline{V}(\mathbf{x}, \vec{\omega}_{\text{i}}) \int_0^s T_{\text{r}}(\mathbf{x}, \mathbf{x}_t)\ \mathrm{d}t, \\ \overline{V}(\mathbf{x}, \vec{\omega}_{\text{i}}) &= s^{-1} \int_0^s V(\mathbf{x}_t, \vec{\omega}_l)\ \mathrm{d}t. \end{aligned} \tag{4.2}$$

The first integral can be computed analytically as the medium is assumed to be homogeneous: $\int_0^s T_{\text{r}}(\mathbf{x}, \mathbf{x}_t)\ \mathrm{d}t = \int_0^s e^{-t\sigma_{\text{t}}}\ \mathrm{d}t = \frac{1}{\sigma_{\text{t}}}(1 - e^{-s\sigma_{\text{t}}})$. Thus, the challenge lies in the computation of the average visibility $\overline{V}$ for a given view ray. In Section 4.3.2, we will show how to lift this simplification of out-factored visibility and compute a better approximation of Equation 4.1. In the following explanations, we will drop the direction parameter $\vec{\omega}_l$ for V for clarity reasons as we only consider a single directional light source.

4.2.2 Rectified Shadow Map and Epipolar Geometry

In a rectified shadow map, view rays share the same y,z-coordinates and are in turn parallel to the x-axis, as well as to each other (Figure 4.2). Consequently, the view rays can be parameterized by two parameters: y to indicate the shadow map row, and z, its depth. View rays go from left to right, hence, it is possible to map all view rays formed by the camera model to a range of $[0;1]$ for y and z. In contrast, light rays are indexed by (x,y) similar to usual shadow mapping. With epipolar coordinates, z denotes the angle spanned by the view-ray direction and the light direction. All view rays with the same y-coordinate lie in a so-called epipolar slice. In turn, an epipolar slice projects on an epipolar line in the camera view. The advantage of such a rectification is that

one can accelerate the ray marching along view rays, e.g., by a 1D min-max mipmap [Chen et al., 2011]. Our approach will also make use of the constant z of each view ray. Details on the rectified shadow map construction via reprojection and our novel projective transformation will be presented in Section 4.4.

4.3 Single-Scattering Method

Our main observation is that solving average visibility in Equation 4.2 is similar to filtering hard shadows for surfaces. In fact, percentage-closer filtering (PCF) performs visibility tests for a single surface point against multiple nearby shadow-map texels in a 2D kernel (*one-to-many*). In our case, for single scattering with a standard shadow map, the filter kernel becomes a line, corresponding to the projection of the view ray into the shadow map. However, one major difference is that the depth along the view ray changes. Hence, different sample points are tested against their corresponding shadow-map texels (*many-to-many*). This issue can be resolved by relying on a rectified shadow map [Chen et al., 2011] (Section 4.4). The constant z-coordinate along a view ray ensures that all sample points will have equal depth. Hence, we are back to testing a single depth value (view ray) to multiple shadow map entries (*one-to-many*) – only within a 1D kernel, derived by the projected view ray.

This conceptional similarity to PCF does not yet result in any performance advantage, as we still need to evaluate each shadow-map texel, which is identical to common ray marching. Our goal is to enable a filter process on the shadow map, which allows us to query the response efficiently. We take inspiration from acceleration schemes for PCF. Annen et al. [2007] approximate the visibility function by a linear combination of basis functions, which enables *prefiltering*, i.e., the shadow map is filtered *before* visibility is evaluated for a specific point. The idea is to decompose the shadow-test functions (each corresponding to a shadow-map texel) into a linear combination of basis functions. This allows to filter the basis functions independent of the coefficients. In consequence, it is also possible to remove all ray-related terms from the single-scattering integral. The performance advantage stems from the fact that filtering a single row in the shadow map results in the filtered response for all rays in the corresponding epipolar slice. Because walking along a ray means traversing the shadow map from left to right, the filtering is a (weighted) prefix summation. The result of the scattering operation for a ray is then obtained by first deriving a few easy-to-compute ray coefficients depending on the ray's depth, and then performing a dot product with the prefiltered basis functions. This leads to a four-step algorithm: computing the rectified shadow map, deriving basis functions, performing prefiltering in form of a prefix sum, and evaluating per pixel the in-scattering along the ray by computing its coefficients.

In the following section, we detail the core of our method: the definition of the basis functions, the prefiltering, and the final ray evaluation. We first focus on the simpler case of computing average visibility of a view ray, before giving a more accurate approximation of Equation 4.1 in Section 4.3.2.

4.3.1 Average Visibility

To compute the average visibility for a view ray, we need to determine:

$$\overline{V}(r) = s^{-1} \int_0^s V(\mathbf{x}_t)\, dt, \tag{4.3}$$

which allows us to solve Equation 4.2. We assume a rectified shadow map as input, hence, the view ray $r := (\mathbf{x}, \omega_i)$, when transformed into the rectified shadow map, is parallel to the x-axis and starts at $x = 0$. Therefore, all points on the view ray have equal y (epipolar slice) and z (depth). In the following, we will denote z_r as the depth of the view ray in the rectified shadow map. Due to the discrete nature of a shadow map, the

binary function V is piecewise constant along the view ray with equally sized steps and its integral can therefore be represented as a sum of functions $\mathcal{V}_{z_i}(z_{\mathbf{x}})$, which describe the visibility function as for shadow mapping:

$$\overline{V}(r) \approx s^{-1} \sum_{i=1}^{s} \mathcal{V}_{z_i}(z_{\mathbf{x}_i}), \tag{4.4}$$

with $\mathbf{x}_i$ describing a sample point on the view ray and s denoting the texel index of the first visible surface point $\mathbf{x}_s$, hit by ray r. $\mathcal{V}$ is a step function with $\mathcal{V}_{z_i}(z_{\mathbf{x}}) = 1$, if $z_{\mathbf{x}} - z_i \leq 0$, else $\mathcal{V}_{z_i}(z_{\mathbf{x}}) = 0$. Note that Equation 4.4 still corresponds to the usual ray-marching procedure with a visibility tests for each sample.

Next, we apply the linearization method of Annen et al. [2007], which allows us to remove the ray-dependent $z_{\mathbf{x}_i}$ from the sum. This enables us to perform a 1D prefiltering along the x-axis on the rectified shadow map, independent of the view rays within an epipolar slice. In detail, we represent each visibility function $\mathcal{V}_{z_i}$ by a linear combination of M basis functions $B_k(z_i)$, such that $\mathcal{V}_{z_i}(z_{\mathbf{x}_i}) \approx \sum_k^M a_{k,i}(z_{\mathbf{x}_i}) B_k(z_i)$. $B_k(z_i)$ can be stored as *basis textures*; a 2D texture array indexed by k. The values are computed directly from the shadow map and each pixel's depth serves as an input to derive the values at the corresponding location in these basis textures.

Key of the previous step is that the coefficients $a_{k,i}$ depend only on the sample points $\mathbf{x}_i$ of r. Further, due to the rectification, the z-values are constant along the ray; $z_r = z_{\mathbf{x}_i} = z_{\mathbf{x}_j}\ \forall i, j$ and, consequently, the coefficients $a_{1,i} \dots a_{M,i}$ are independent of i, and we simply write $a_k(z_r)$:

$$\overline{V}(r) \approx s^{-1} \sum_{i=1}^{s} \left(\sum_{k}^{M} a_k(z_r) B_k(z_i) \right). \tag{4.5}$$

In consequence, we can swap the order of the sums and remove $a_k(z_r)$ from the sum over the texels:

$$\overline{V}(r) \approx s^{-1} \sum_{k}^{M} a_k(z_r) \left(\sum_{i=1}^{s} B_k(z_i) \right). \tag{4.6}$$

Now, the inner sum only depends on the depth values stored in the shadow map, which makes it possible to filter the basis functions $B_k(z_i)$ independent of the ray. However, while Annen et al. could directly apply a fixed filter kernel to the basis textures, our situation is different. The kernel size is not constant, but requires a ray-dependent limit s, which determines the limit for the summation over the shadow-map texels. To compute s, we need to project the surface point hit by the view ray into the shadow map, indicating where the ray stops.

Adapting the Filter Kernel The solution to the varying kernel size is to compute and store the 1D prefix sum for all possible view ray distances up to texel $s_{\max}$, which corresponds to the number of columns of the rectified shadow map. Consequently, the resulting *filtered-basis textures* share the resolution of the rectified shadow map and each row still corresponds to an epipolar slice. A texel (x, y) gives us access to the sum over the first i basis functions of row y (for each basis function k). We store all k basis functions in the channels and layers of the filtered-basis textures, resulting in a *filtered-basis vector* accessible at each texel. To obtain the sum of the functions between two locations u, v on the ray, it is sufficient to retrieve the *filtered-basis vector* for these two locations and to compute a difference between the two.

Evaluating a Ray Given the filtered-basis textures, we can evaluate the scattering along a ray r via a dot product in a constant-time evaluation. The first vector corresponds to the coefficient vector $\vec{a}(z_r) := a(a_1(z_r), \dots, a_M(z_r))^\top$ that we compute from r. For the second, we transform the hit point $\mathbf{x}_s$ into the rectified shadow map, and retrieve the filtered-basis vector from the corresponding texel in the filtered-basis textures.

Optionally, we can also retrieve the filtered-basis vector for the ray's origin on the near plane projected into the rectified shadow map and subtract this contribution to get in-scattering only within the near and far plane.

Nonetheless, in practice, as this correction only affects the part of the ray from the camera to the near plane, it can usually be safely neglected.

Fourier Series as Basis Functions One important choice regarding the algorithm is the specific basis functions to use, which can have a strong impact on quality and performance. In practice, we found that a Fourier series as used by Annen et al. [2007] is also the best choice for our purposes. Choosing the Fourier series implies that the function to be transformed needs to be periodic. To satisfy this requirement, we observe that $\mathcal{V}_{z_i}(z_{\mathbf{x}_i}) = H(z_{\mathbf{x}_i} - z_i)$, where H is the Heavyside step function ($H(x) = 1$ for $x < 0$, else $H(x) = 0$). As z_i and z_r only take values in the range of $[0;1]$, H will only be used with values in $[-1;1]$. As we are only interested in the domain $[-1;1]$, we can conceptually assume H to be periodic. Annen et al. suggest to use $H(\pi x) - 0.5$, which is a stretched and shifted version, leading to the domain $[-\pi, \pi]$ as well as point symmetry, both important aspects for the Fourier series transformation. Applying the Fourier series and some trigonometric identity operations (see Annen et al. [2007] for details), results in relatively simple coefficients:

$$\begin{aligned} a_{(2k-1)}(z_r) &= 2c_k^{-1}\cos(c_k z_r), & a_{(2k)}(z_r) &= -2c_k^{-1}\sin(c_k z_r) \\ B_{(2k-1)}(z_i) &= \sin(c_k z_i), & B_{(2k)}(z_i) &= \cos(c_k z_i) \end{aligned} \tag{4.7}$$

with $c_k = \pi(2k-1)$. Thus, we obtain:

$$\begin{aligned} \mathcal{V}_{z_i}(z_r) &= (H(\pi(z_r - z_i)) - 0.5) + 0.5 \\ &\approx \sum_{k=0}^{M} a_k(z_r) B_k(z_k) + 0.5 \\ &= (a_1(z_r), \ldots, a_M(z_r))\ (B_1(z_i), \ldots, B_M(z_i))^\top + 0.5, \\ &= \vec{a}(z_r)\ (B_1(z_i), \ldots, B_M(z_i))^\top + 0.5, \end{aligned}$$

which implies

$$\sum_{i=1}^{s} \mathcal{V}_{z_i}(z_r) = \vec{a}(z_r)\ (\sum_{i=1}^{s} B_1(z_i), \ldots, \sum_{i=0}^{s} B_M(z_i))^\top + 0.5.$$

The truncation to M basis functions enables a practical use of the Fourier series, but only approximates the visibility function. Implications are discussed in Section 4.5.

We also tried alternative basis functions, such as exponential functions [Annen et al., 2008; Salvi, 2008]. Unfortunately, they fail because the resulting approximation of V leads to values that exceed one by far when evaluating them off the surface. While this effect is less problematic for surface shadows, where the function is actually evaluated on the surfaces that define V, it is a real problem for scattering because the visibility needs to be checked anywhere in space. Other alternatives include statistical methods, such as variance shadow maps [Donnelly and Lauritzen, 2006] and exponential variance shadow maps [Lauritzen and McCool, 2008] that give a too coarse approximation (Figure 4.3).

4.3.2 Extensions

The method described in Section 4.3.1 allows us to compute an average visibility, but it omits weighting by transmittance (cf. Equation 4.2). Correctly applied, the weights have a exponential distribution just as transmittance along a ray. Hence, average visibility mostly works well for optically-thin media, where transmittance has low impact with a flat exponential distribution. For thicker media, we show how an approximate weighting can be employed to increase accuracy. Further, we explain the use of textured and spotlight sources.

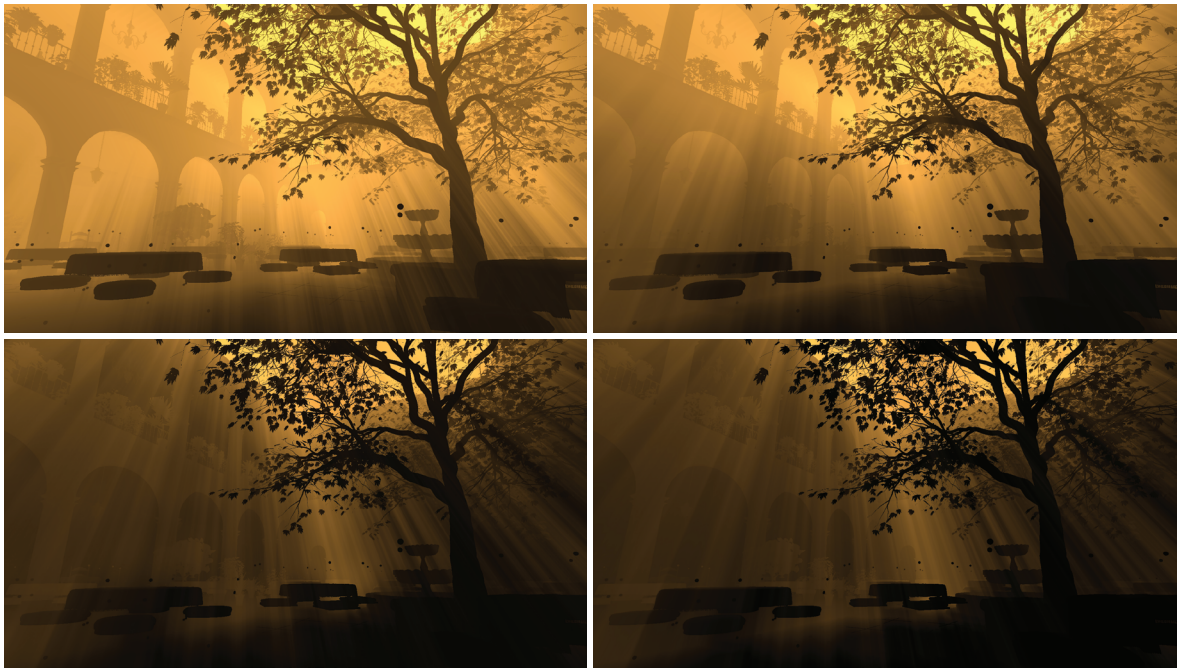


Figure 4.3: Alternatives to Fourier series. Top: variance shadow maps (left), exponential variance shadow maps (right). Bottom: our method (left), reference (right). Filterable, efficient statistical methods approximate V too coarsely. Note, we used factored visibility here (Equation 4.2), resulting in slightly different images than shown in Figure 4.6, which use the more accurate solution presented in Section 4.3.2.

Transmittance Weighting Along View Rays In Equation 4.1 visibility V is weighted by the transmittance term $T_{\mathrm{r}}(\mathbf{x}, \mathbf{x}_t) = e^{-t\sigma_{\mathrm{t}}}$, which can be evaluated analytically for a piecewise constant V based on a shadow map. Starting with the discrete space of the shadow map, we denote as w_i as weighting factor for the i-th segment along the view ray. Then, we obtain:

$$\begin{aligned} L_{\mathrm{scat}}(\mathbf{x}, \omega_{\mathrm{i}}) &\approx C \sum_{i=1}^{s} \left(\int_{(i-1)\Delta}^{i\Delta} e^{-t\sigma_{\mathrm{t}}}\, dt \right) \mathcal{V}_{z_i}(z_{\mathbf{x}_i}) \\ &= C \frac{1}{\sigma_{\mathrm{t}}} \sum_{i=1}^{s} w_i\, \mathcal{V}_{z_i}(z_{\mathbf{x}_i}), \\ w_i &:= \sigma_{\mathrm{t}} \int_{(i-1)\Delta}^{i\Delta} e^{-t\sigma_{\mathrm{t}}}\, dt = e^{-(i-1)\Delta\sigma_{\mathrm{t}}} - e^{-i\Delta\sigma_{\mathrm{t}}}, \end{aligned} \tag{4.8}$$

where Δ denotes the step size. The weighting by w_i is incorporated into the prefiltering (cf. Equation 4.6) as:

$$\begin{aligned} L_{\mathrm{scat}}(\mathbf{x}, \omega_{\mathrm{i}}) &\approx C \frac{1}{\sigma_{\mathrm{t}}} \sum_{i=1}^{s} \left(w_i \sum_{k}^{M} a_k(z_r) B_k(z_i) \right) \\ &= C \frac{1}{\sigma_{\mathrm{t}}} \sum_{k}^{M} a_k(z_r) \left(\sum_{i=1}^{s} w_i B_k(z_i) \right). \end{aligned} \tag{4.9}$$

To apply the prefix-sum-like prefiltering (i.e., weighted sum over the basis functions) the weights and basis functions need to be ray-independent, which is not the case anymore. The step size Δ is implied by the texels of the rectified shadow map and its magnitude is different for view rays in the same epipolar slice, as it is defined by the angle between the light and view-ray direction. Therefore, Equation 4.9 is not yet a ray-independent, weighted filtering.

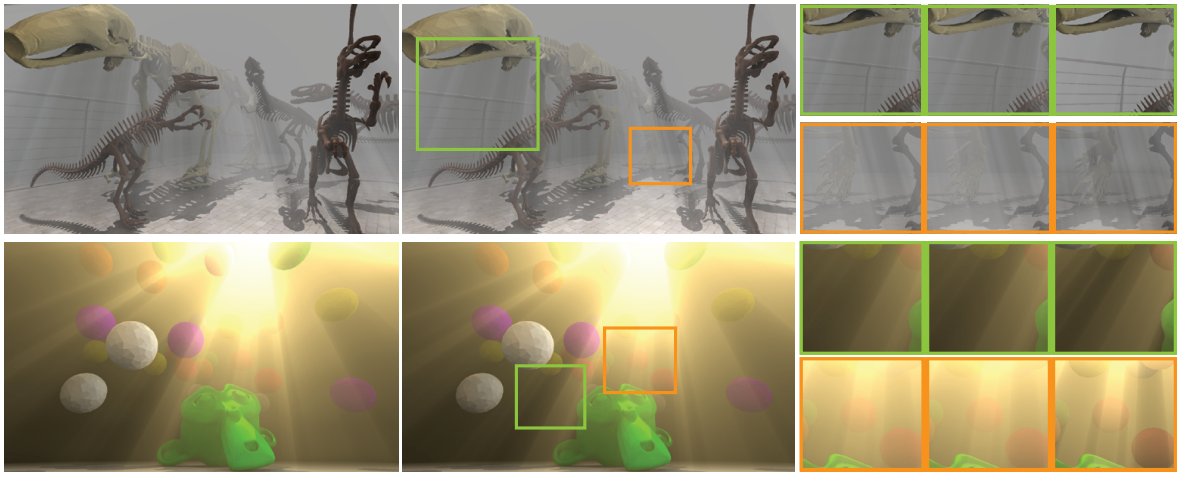
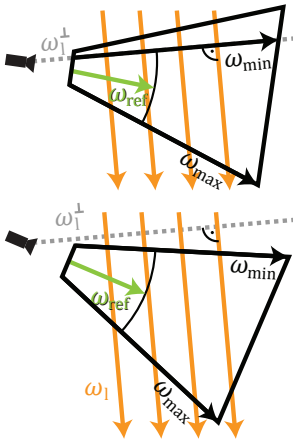


Figure 4.4: Importance of transmittance-weighted visibility. Left: our method; center: reference ray-marched in-scattering. The insets on the right show (left to right): our method with transmittance weights; ray-marching reference with correct attenuation; non-weighted average visibility. Average visibility gives plausible results, but many significant differences occur. Shadows are missing or are overly dark. The difference increases with the thickness of the participating media.

As a compromise, we opt for finding a good constant for all rays inside the same epipolar slice. To this extent, we pick a reference ray in the epipolar slice whose world step size Δ_{ref} corresponding to a shadow-map texel will be used as a representative for all rays when computing w_i.



There are several ways such a ray could be defined depending on the optimization strategy one chooses. We propose the following solution. We start by relating the different Δ's to each other. The minimal distance $\Delta^\perp$ to cross a shadow map texel is given by the ray that is orthogonal to the light direction $\omega_l^\perp$. The Δ of another ray in the same epipolar slice with direction ω_i can be expressed in dependence of its angle to $\omega_l^\perp$: $\Delta = \frac{1}{k(\omega_i)}\Delta^\perp$ with $k(\omega_i) := \omega_i \cdot \omega_l^\perp$ being the cosine of the angle between both. Thus, the relationship between any two rays with directions ω_i and ω_i' is given by $c = \frac{\omega_i \cdot \omega_l^\perp}{\omega_i' \cdot \omega_l^\perp} = \frac{k(\omega_i)}{k(\omega_i')}$. Therefore, we seek a reference ray, for which this ratio is close to one for all rays in the epipolar slice. The first task is to find the rays with directions $\omega_{\text{min}}, \omega_{\text{max}}$ that have $\Delta^{\text{min}}, \Delta^{\text{max}}$ (the minimum and maximum Δ values) within the epipolar slice inside the camera frustum. If $\omega_l^\perp$ is part of the frustum, it directly gives ω_i^{min} as it has the minimal distance $\Delta^\perp$ to cross a shadow map texel. If this is not the case one of the limiting rays must be ω_{min}, basically, the angularly closest direction to $\omega_l^\perp$ within the camera frustum. ω_{max} has a similar relationship; if any of the view rays is parallel to the light direction ω_l, we get $\omega_{\text{max}} = \omega_l$, because it forms the largest angle of 90 degrees with $\omega_l^\perp$; otherwise, ω_{max} is found as one of the limiting rays, angularly farthest away from $\omega_l^\perp$. Any reference view ray should now lie between ω_{min} and ω_{max}. A simple and robust approach is to use the bisectrix between $\omega_{\text{min}}, \omega_{\text{max}}$. This, however, is not necessarily the optimal choice. To get an optimal ω_{ref}, we seek to minimize the maximal errors that occur for ω_{min} and ω_{max}:

$$c_{\text{min}} = \frac{\omega_{\text{ref}} \cdot \omega_l^\perp}{\omega_{\text{min}} \cdot \omega_l^\perp} = \frac{k(\omega_{\text{ref}})}{k(\omega_{\text{min}})} \leq 1, \qquad c_{\text{max}} = \frac{\omega_{\text{ref}} \cdot \omega_l^\perp}{\omega_{\text{max}} \cdot \omega_l^\perp} = \frac{k(\omega_{\text{ref}})}{k(\omega_{\text{max}})} \geq 1. \tag{4.10}$$

We use the shorthand $k_{\text{ref}} := k(\omega_{\text{ref}})$ in the following. As we care about the relative scale of c_{min} and c_{max}, we

need to invert one to compare them. Now, $c_{\min}^{-1}$ and $c_{\max}$ have a range of $[1;\infty)$ and k_{ref} can be optimized to minimize both. Writing both as functions of k_{ref}: $c_{\min}(k_{\text{ref}}), c_{\max}(k_{\text{ref}})$, we can formalize the optimization as:

$$\min_{k_{\text{ref}}} E(k_{\text{ref}}); \;\; E(k_{\text{ref}}) := \max(c_{\min}^{-1}(k_{\text{ref}}), c_{\max}(k_{\text{ref}})). \tag{4.11}$$

Due to the inversion $c_{\min}^{-1}$, increasing k_{ref} monotonically decreases $c_{\min}^{-1}(k_{\text{ref}})$, but also monotonically increases $c_{\max}(k_{\text{ref}})$ and vice versa. Hence, the intersection of both functions $c_{\min}^{-1}(k_{\text{ref}}) = c_{\max}(k_{\text{ref}})$ yields the minimum of $E(k_{\text{ref}})$ at point:

$$k_{\text{ref}} = \sqrt{k(\omega_{\min})\, k(\omega_{\max})} = \sqrt{(\omega_{\min} \cdot \omega_{\text{l}}^{\perp})(\omega_{\max} \cdot \omega_{\text{l}}^{\perp})}. \tag{4.12}$$

We optimized for $k_{\text{ref}} = \omega_{\text{ref}} \cdot \omega_{\text{l}}^{\perp}$, which is the cosine of the angle of the optimal reference ray to the orthogonal light direction and is used to derive ω_{ref}. Note that $k(\omega_{\text{l}})$ approaches zero for ω_{l}. Hence, we cannot compute the optimum, iff $\omega_{\max} = \omega_{\text{l}}$ as in this case $c_{\max} = \infty$. Due to this practical limitation, we generally use the bisectrix as the reference ray, which already delivers a very good approximation (Figure 4.4).

It is noteworthy that other work completely factors out visibility [Wyman, 2011] or uses low-rank approximations [Baran et al., 2010; Chen et al., 2011]. Our w_i weights are a cheap, but good approximation with a positive impact on the visual quality (Figure 4.4). As our weighting has practically no impact on performance, we generally recommend using this weighted variant over average visibility.

Local Lights A challenging task is to integrate the distance falloff along light rays, which, in contrast to directional sources, is visually important for local light sources. Factoring visibility and accounting for the distance falloff in an analytical solution for unoccluded scattering [Pegoraro and Parker, 2009] as in Wyman [2011] is trivially supported by our approach. The difficulty lies in the combined evaluation and other approaches face this issue as well; we could follow the suggestion of Baran et al. [2010] who rely on additional basis functions for the falloff. However, such a solution is a bit cumbersome and combining all falloff and shadow basis functions would result in an increased memory consumption. A better alternative would be to directly choose basis functions which directly expand fractional visibility along light rays and support the falloff [Jansen and Bavoil, 2010], but such a choice would decrease the shadow accuracy.

For omnidirectional sources, a single shadow map is insufficient, as it cannot cover the entire sphere. This case can be treated by defining six independent spotlights for each directional sector, corresponding to the face of a cube map, similar to surface shadows. The different contributions of these sources are simply added.

Angular-dependency of Light Sources Integrating light sources with changing angular contribution is straightforward; during the prefiltering step, each visibility function is simply weighted by the light's contribution just as is done for the transmittance weighting. In particular, this weighting allows us to simulate an angular falloff. If the light has an associated color texture, we need to store one value for each color component. Unfortunately, this step triples storage amount and computation costs (for RGB).

4.3.3 Implementation

In this section, we take a careful look at the implementation of the algorithm given a rectified shadow map as input, which can be built with our technique presented in the next section, or one of the existing previous solutions that rely on a resampling of a regular shadow map [Chen et al., 2011].

Prefiltering The first task is to compute the basis textures from the rectified shadow map, which is a direct implementation of Equation 4.7. The results of the sine and cosine operations are additionally mapped to the

```
#define NUM_BASES 32
// ...

void main() {
   // ...
   int y = gl_GlobalInvocationID.y; // epipolarSlice
   // in practice: pack 4 bases in vec4
   float filtB[NUM_BASES];
   for(int i=0; i<textureSize(texInBasis,0).x; ++i) {
      for(int k=0; k<NUM_BASES; ++k) {
         float basis = texelFetch(texInBasis, ivec3(x,y,k), 0).r;
         // accumlate bases == increasing filter kernel
         // == Σ_{i=1}^{s} B_k(z_i) / s
         filtB[k] = mix(filtB[k], basis, 1.f / (i+1));
         imageStore(imgOutFilteredBasis, ivec3(i,y,k), filtB[k].rrrr);
      }
   }
}
```

Listing 4.1: Basis-texture filtering.

range of $[0;1]$, which allows us to store the basis textures as a RGBA8 Texture2DArray (usually 8 layers for $M = 32$ coefficients). Pseudo code is shown in Listing 4.1.

The next step is to filter the basis textures in a prefix-sum-like manner. Interestingly, the simplest implementation of the prefix sum also turned out to be the fastest and we present it in the following. We implemented a compute shader that launches one thread per row of the rectified shadow map. Due to the rectification, this scheme corresponds to a parallel execution over epipolar slices. As each thread starts on the left and proceeds to the right, the values are written into the texels of the filtered-basis texture. In consequence, only very few registers are needed and the execution is fast.

The above computation addresses average visibility, however, as suggested in Section 4.3.2 more accuracy can be achieved by performing a weighted filtering. To compute ray-independent weights, in the last section we suggested to choose a reference ray per epipolar slice first. Computing this ray can be done in a 1D compute shader implementing the previously sketched procedure. The ray's world-space distance Δ_{ref} for a shadow-map texel will determine the weighting factors. Δ_{ref} is simply the distance between two points on the ray, projecting to two neighboring texels of the rectified shadow map. Listing 4.2 shows the adaptations to the filtering to account for the non-uniform filter kernels.

Instead of storing $\sum_{i=1}^{s} w_i B_k(z_i)$ for each k, we normalize the sum by the filter weights: $\frac{\sum_{i=1}^{s} w_i B_k(z_i)}{\sum_{i=1}^{s} w_i}$ (compare the input weights to the mix in Listings 4.1, 4.2). The values of these normalized, filtered basis functions are therefore in the range of the actual basis functions, hence, $[-1;1]$ for sines and cosines and can be safely mapped to $[0;1]$. In consequence, low-precision filtered-basis textures can be used – no improvement was visible for textures with more than 8bit – leading to a significant reduction in memory and an increase in performance. It is easy to recover the correct value during the scattering evaluation for a specific view ray by multiplying the filtered basis-vector with $\sum_i^s w_i$ in the ray evaluation step.

A possibility to reduce bandwidth costs is to merge the shader to compute the basis transformation with the shader to perform the filtering. This effectively saves reading and writing of the entire basis texture once, which would otherwise happen in the filtering and transformation shader. The marginal downside of this is the added

```
uniform vec3 fPlaneItsWRefRay; // far plane intersection point
uniform float Δref;

void main() {
   // ...
   float stepSize = Δref;
   float opticalDepth = stepSize * sigma_t; // Δref σt
   // transmittance of a single step
   float stepTransmittance = exp(-opticalDepth);
   float totalTransmittance = 1;
   float sumWeights = 0;
   for(int i=0; i<textureSize(texInBasis,0).x; ++i) {
      // directly use transmittance as weight ui = e^{-iΔref σt} = ∏_{j=0}^{i} e^{-Δref σt}
      // it has the same distribution as wi, but is faster to compute
      // hence, ui = c wi, while c does not matter
      totalTransmittance *= stepTransmittance;
      float weight = totalTransmittance;
      sumWeights += weight;
      for(int k=0; k<NUM_BASES; ++k) {
         // ...
         // == (Σ_{i=1}^{s} ui Bk(zi)) / (Σ_{i=1}^{s} ui) = (c Σ_{i=1}^{s} wi Bk(zi)) / (c Σ_{i=1}^{s} wi) = (Σ_{i=1}^{s} wi Bk(zi)) / (Σ_{i=1}^{s} wi)
         filtB[k] = mix(filtB[k], basis, weight / sumWeights);
         // ...
      }
   }
}
```

Listing 4.2: Weighted basis-texture filtering.

computational workload to the filtering shader as it needs to read shadow map values and transform those to the basis-vector (see Listing 4.3).

The parallel execution over epipolar slices results in a process of low parallelization, potentially not fully utilizing the GPU during the prefiltering. We present two strategies to split the sequential filtering into smaller tasks that add a small amount of bandwidth costs but increase parallelism in the prefiltering stage. The first choice is to parallelize over the k bases, i.e., multiple threads compute the filtering for the same epipolar slice, but for different bases (as bases are packed in `vec4`, the maximum parallelization is achieved with 4 bases per thread). However, this causes each thread to access the shadow map and compute the filtering weights although these tasks are basis-independent. The second strategy shifts work from the prefiltering stage to the final scattering evaluation. Instead of performing the prefix sum for an entire row, we split the row uniformly, e.g., into three regions, and compute the filtering independently, hereby increasing parallelism. When evaluating scattering for a ray r, we consequently need several lookups; one from the region that contains the first visible surface of r, and one more for each filtered-basis texture region in front of it – with three regions, a maximum of three lookups. In combination, these two parallelization strategies increase the amount of parallel threads by an order of magnitude (e.g., $3\times$ by uniform splits and $4\times$ by each thread filtering only 8 of the 32 bases; our default). While performance can be improved in this way, we found it difficult to provide exact suggestions on how many splits to perform and bases to use per thread. The reason is that these numbers are highly hardware dependent. Furthermore, the speed-up is mostly influenced by how efficient the initial parallelization over epipolar slices already performs, which, in turn, also depends on the shadow map resolution. For our hardware

```
void main() {
   for(int i=0; i<textureSize(texInDepth,0).x; ++i) {
      float depth = texelFetch(texInDepth, ivec2(x, y), 0).r;
      // ...
      for(int k=0; k<NUM_BASES; ++k) {
         float basis = transformBasis(depth, k);
         // ...
      }
   }
}
```

Listing 4.3: Live transformation of the shadow map into basis functions.

Table 4.1: Computation time of our method with different parallelization parameters at the prefiltering stage for 32 basis functions. Note that we use the naive parallelization over epipolar slices and the 32 bases as the performance reference. As test scene we used *San Miguel* with a screen resolution of 1920×1080 and a shadow map resolution of 1024×1024, which is sufficiently high to produce artifact-free results in all the scenes shown in this chapter.

# Regions per Slice	# Bases per Thread	Performance in ms			
		Prefix Sum	Ray Evaluation	Total	+%
1	4	1.85	0.62	2.47	0%
1	8	2.49	0.62	3.11	-21%
1	16	4.05	0.62	4.67	-47%
2	4	1.19	0.96	2.15	+6%
2	8	1.37	0.96	2.33	-21%
2	16	2.03	0.96	2.99	-17%
3	4	1.50	1.04	2.54	-3%
3	8	0.93	1.04	1.97	+25%
3	16	1.32	1.04	2.36	+5%
4	4	1.28	1.33	2.61	-5%
4	8	1.16	1.33	2.49	-1%
4	16	1.10	1.33	2.43	+2%

(an Nvidia Titan), three regions and eight bases per thread led to a speed up of about 20% compared to a naive parallelization over all bases (in addition to the parallelization over epipolar slices); see Table 4.1.

We also tested a GPU-optimized prefix sum [Harris et al., 2007] designed for sums of complete arrays (while we only require a weighted prefix sum per row and per basis). It makes use of shared memory and avoids some synchronization by performing work in predefined warp sizes. We implemented the algorithm using OpenGL compute shaders (applying the same optimizations as the original work that uses CUDA) and adopted the version to perform filtering instead of computing sums. However, this variant is two times slower than the previously-described simpler approach. Probably, remaining synchronization steps and shared memory create a bottleneck, but this may again depend on the used hardware.

Evaluating a Ray The scattering evaluation for the view rays is performed in a simple fragment/compute shader, executed per screen pixel (see Listing 4.4). The view ray is easily computed in the shader, e.g., by linearly blending the frustum vertices at the near / far plane.

Assuming a deferred-shading pipeline, we retrieve the depth component from the G-buffer to compute the

```
void main() {
   Ray viewRay;
   vec3 sPosWS;
   float disToSurface;
   getViewRaySurfacePos(depthMap, pixelCoord, viewRay, sPosWS, disToSurface);
   vec3 sPosRS = toRectifiedSpace(sPosWS);
   vec4 coeff[NUM_BASES/4], basis[NUM_BASES/4];
   computeCoeff(sPosRS.z, coeff);
   fetchBases(basisFilteredTex, basis);
   float avgVis = 0.5;
   for(int k=0; k<NUM_BASES/4; ++k) {
      avgVis += dot(coeff[k], basis[k]);
   }

   vec3 C = getRayConstantScatteringParam(viewRay);
   vec3 inScat = C / σt * (1 - exp(disToSurface*σt)) * avgVis;
}
```

Listing 4.4: Scattering evaluation per view ray.

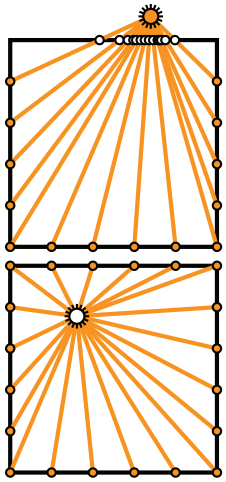
surface point up to which the single scattering integral has to be computed. We map it to the space of the rectified shadow map (Section 4.4) to query the filtered-basis vector – on a side note, in the case of our novel rectification the mapping is a simple projective transformation just as with usual shadow mapping. Next, we compute the coefficients of the view ray using Equation 4.7 and the z-component of the transformed surface point, which results in the z value of the view ray, as the latter is constant along the ray, hence, also at the hit point. We evaluate the dot product between coefficients and filtered-basis functions (mapped to range $[-1;1]$ after the texture access) in a loop and add a constant of 0.5 as required by the Fourier series approximation. Note that the pseudo-code is the same for average visibility as well as transmittance-weighted visibility as in the latter case the filtered-basis vectors are stored normalized (by $\sum_{i=1}^{s} w_i$) and are restored by the transmittance to the surface point as by definition $\sum_{i=1}^{s} w_i = \sigma_t \int_0^s T_r(\mathbf{x}, \mathbf{x}_t)\, dt = 1 - e^{-s\sigma_t}$.

4.4 Rectification

In this section, we discuss the creation of a rectified shadow map, in which all view rays have a constant z_r value along the rays as required by Equation 4.6, run from left to right in the shadow map and are all parallel to each other. We first recapitulate the standard resampling approach of Baran et al. [2010], which is based on epipolar coordinates. Then, we introduce a novel linear rectification scheme that uses a projective transformation and allows to directly rasterize a scene into a rectified shadow map.

4.4.1 Epipolar Geometry

The method of Baran et al. [2010] uses epipolar geometry to obtain a rectified shadow map. In principle, the mapping of a point from world to the rectified space is defined as follows: z is the angle spanned by the view ray and the light direction in a range of $[0;\pi]$, y is the angle, which in conjunction with the camera position and the light direction defines the epipolar slice, which has a range of $[0;2\pi]$, and x is the length of the vector from the camera to the projection of a point onto a reference ray in the epipolar slice, see Figure 4.2. Conveniently, we can use the ray that is optimal for transmittance weighting (defined in Section 4.3.2) as a reference ray.



However, x still needs to be bounded tightly and view-dependent bounds need to be found for y, z. These tight bounds are important to avoid undersampling and to increase precision. Given the bounds, we can then define the mapping from world to rectified space.

To find the bounds for y, indicating the epipolar slice, it is possible to sample the screen border pixels and derive their epipolar slices based on their view ray directions respectively. Hereby, we implicitly define the y resolution of the rectified shadow map (up to $2\times$ width and height of the screen) and ensure that any screen pixel is at most half a pixel away from the closest epipolar slice that is captured in the rectified shadow map. All other pixels on the screen can be associated to their corresponding slice by projecting its position along the epipolar line to the border of the image. Engelhardt and Dachsbacher [2010] as well as Baran et al. [2010] discuss optimal epipolar sampling strategies that avoid some of the unnecessary oversampling of the screen border approach.

Bounds for the angle z between view ray and light direction can be found by examining the limiting view rays of an epipolar slice. The first limiting ray is already given by the epipolar slice sampling approach (border pixel, yellow dots in figure), the second is either the epipole (light source), if it is within the screen area or the projection of the first border pixel into the direction of the epipole to the screen border (white dots).

Finding an upper bound for x, the length of the vector from the camera to the projection of a point within the epipolar slice onto a reference ray is also simple. For each epipolar slice, we take its reference ray and proceed as follows; we project the far plane points of the limiting rays of the epipolar slice onto the reference ray, compute the distance to the camera, and take the maximum.

The transformation of the scene into the rectified shadow map based on the bounds derived for each epipolar slice does not preserve straight lines, which is a challenge for the graphics card. Hence, it is more efficient to first create a standard shadow map, which is subsequently point-resampled into a rectified shadow map. The reprojection is performed in a compute shader over all texel in the rectified shadow map. Each light ray, defined by the texel position in the rectified shadow map, can be mapped to world space and then to the shadow map to recover the necessary input shadow map texel. Given its value, we can then compute the necessary depth value (the angle) for the rectified shadow map.

One catch are blockers between the light and the camera frustum. They would not necessarily fall within the ranges defined by the bounds of the rectified shadow map. This problem can be solved easily by clamping their original shadow-map depth to the camera frustum. Hereby, their values become compatible with the range and they still cast the same shadow into the visible part of the scene.

Although the performance of the resampling is high (1024×1024: 0.1ms, 4096×1536: 0.6ms, 4096×4096: 1.5ms), it is preferable to avoid resampling in general because it can lead to under- or over-sampling. Further, this mapping does not allow us to easily perform efficient lookups, e.g., via a simple matrix multiplication, as for shadow mapping. Instead a projection of the pixel onto the screen border is needed and more involved. In the following, we will present a new technique to address these issues.

4.4.2 Our Projective Rectification

Here, we introduce our linear rectification, which avoids reprojection completely by constructing a special projection matrix to replace the standard light view-projection matrix during the rendering of the shadow map. In consequence, points can be mapped easily into this space by applying a homogeneous transformation. Hence, not just the creation of the rectified shadow map is simplified, but also lookups into it. To find the corresponding

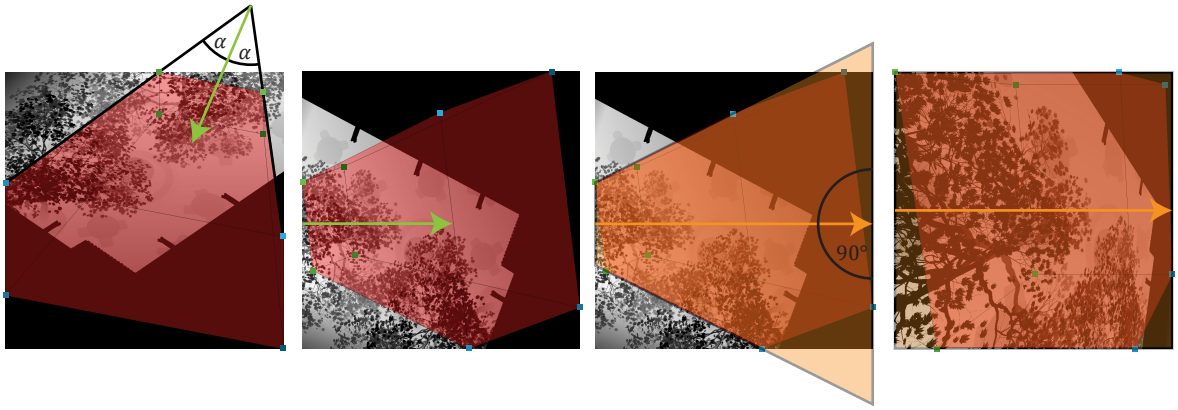


Figure 4.5: Camera frustum overlayed on a shadow map. Rectification (left to right): 2D bisectrix is found by applying the standard light view transformation; a new light view transformation is computed such that bisectrix goes from left to right; light frustum is constructed with near/far planes parallel to the light direction; rectification is achieved via a perspective projection.

transformation, we construct a frustum with an associated matrix that maps the scene in such a way that it fulfills the constraints of our scattering approach, i.e., all view rays in an epipolar slice map to a row of the shadow map, the depth along a view ray is constant, rays in this light space go from left to right, and the shadow map tightly encompasses the camera frustum.

Constructing the Linear Rectification Matrix Figure 4.5 gives an overview of the construction steps for a directional light source. While they could be grouped, this step-by-step construction is easier to follow. We first determine the 3D bisectrix of the camera frustum projected along the light direction (Figure 4.5, top-left), which is achieved in multiple small steps as follows. We apply a standard light space *view* transform from the camera, in direction of the light using any possible up vector. Next, we project the camera frustum corners into this space and find the 2D vectors from the projected camera position to projected corners that span the biggest angle. From these two vectors, we compute the 2D bisectrix, which is turned into a 3D vector by setting $z = 0$ in the light space. In consequence, this 3D bisectrix is orthogonal to the light direction. The bisectrix is used to find a new up vector for the light space *view* transform, which is a rotated version around the light direction, such that the bisectrix now goes from left to right (Figure 4.5, second from left).

Next we seek a transformation with the camera as the center of projection to get view rays that are parallel to each other. We need to find a light frustum that is wrapped around the camera's frustum. Choosing the y and z axis to define the parallel planes of the frustum allows the use of the common projection matrix form with the parameters near/far, left/right, and top/bottom offsets from the center of projection. However, different from the common use of perspective projection along z, we perform a projective transformation along x, hence, near and far correspond to the minimal and maximal x value, respectively, of the camera frustum corners (Figure 4.5, second to right). The frustum's left/right (side planes, y axis) and top/bottom (top planes, z axis) offsets are also found by the minimal and maximal y,z coordinates of the corner points (marked dots in Figure 4.5). Filling the projection matrix with the computed parameters yields the final transformation. Note that the bisectrix causes that parameters left/right to have the same absolute value, which not necessarily holds for top/bottom. Hence, the light frustum can be skewed, but this has no negative effect on the projection. The final matrix can be used to generate a rectified shadow map with all needed properties (Figure 4.5, right). Additionally, using depth clamping (NV_DEPTH_CLAMP) when rendering the shadow map ensures that objects shadowing the camera

frustum, but lying outside of it, are rendered in the shadow map as well.

It is important to note that the construction is not possible if any of the view rays is parallel to the light direction. This is the case when the epipole is within the screen area, i.e., when looking directly into the light source or into the same direction as the light. In this situation, we need to switch back to a texel-wise reprojection.

Non-linearity of the Rectification Our frustum-based rectification results in a non-linearity along the view rays (x-axis in the shadow map), similar to the non-linearity of the depth buffer. The effect increases the more view and light rays are parallel to each other. Thus, we also return to texel-wise reprojection for close-to-degenerated cases (easily detectable as distance of the epipole to screen border). However, the increased shadow map resolution near the observer can be an advantage, as, due to transmittance, the in-scattered light near the observer receives more precision.

We reduce the non-linearity by using a method similar to split shadow maps [Zhang et al., 2006]. Here, we split the light view into different regions (from left to right), which increase in size. If each region is then rendered into a shadow map of the same resolution, the higher resolution near the observer is counteracted. Similar to epipolar geometry, straight lines are not preserved anymore over all splits, hence, we need to rasterize the scene multiple times, once for each split. The splitting here is conceptually equal to the splitting for increased parallelization at the prefiltering stage, described in Section 4.3.3 and can be combined with this step.

Using the rectification has an impact on the weights w_i that modulate the prefix sum in the scattering approach (cf. Equation 4.8). In our rectification, the travelled distance along a view ray increases by a constant factor d from one shadow-map texel to the next. This result stems from the fact that a perspective transformation, such as our rectification, preserves the cross ratio of points [Heckbert, 1989]. The weight becomes $w_i' := e^{-\Delta\sigma_t \sum_{j=1}^{i} d^{j-1}}(1 - e^{-\Delta\sigma_t d^i})$. To compute d, we transform the first three texel centers of a shadow-map row into world space (the depth does not matter, but need to be the same for all three) and compute the cross ratio of the distances. For a given shadow-map row, the factor d only needs to be determined once before launching its prefiltering. Please note that this ratio relation along the ray is not the same as the previously established ratio relation between different rays of an epipolar slice.

Discussion Our novel rectification method has several advantages. First, no resampling is needed. Second, mappings from world to rectified space boil down to a matrix-vector multiplication similar to common shadow mapping. This operation is faster to execute and easier to implement than the projection of a pixel to the screen border to compute the epipolar slice. Third, the shadow map resolution can be chosen independently of the transformation; complex epipolar slice sampling is avoided. Due to the regular sampling of the epipolar slices, the transformation provides near optimal sample distribution under any resolution. While we select the rectified shadow map resolution manually, it remains future work to find the minimum y resolution under the constraint that each pixel should be at most half a pixel away from the closest epipolar slice as has been done for epipolar geometry [Baran et al., 2010]. Nonetheless, given that our approach also supports texture filtering, this half-pixel distance does not have to be as accurate as in previous solutions.

The projection warps a light frustum placed around the camera frustum by using the camera as center of projection. Consequently, the camera itself must not lie in the light frustum and the resulting rectified shadow map misses visibility information close to the camera. However, by construction, it contains the entire camera frustum, thus, at most the region from the camera to the camera's near plane is not accessible in the rectified shadow map.

Furthermore, as indicated in Section 4.3.1, the camera's near plane is usually not parallel to the light direction, therefore, the rectified shadow map covers some space between the camera and the near plane (see Figure 4.5, second to right, near plane corners in green). This potentially unwanted segment up to the near plane can be easily subtracted at the cost of an additional lookup as described earlier. However, we want to underline again that, practically, this didn't cause any artifacts in our test cases because the near plane is usually close to the camera and the differences of covered space between camera and near plane area for different pixels is thus very small.

As the camera itself must not lie within the light frustum, our approach cannot replace the epipolar resampling scheme in all cases. Although, while highly impractical, one could avoid view rays being parallel to the light direction by splitting the camera frustum in multiple smaller frusta (such that the corners of the sub-frustum span an angle less than 180 degrees, cf. the first step of the matrix construction) and omit pixels on the epipole to fix this situation.

4.5 Results

We tested our prototype on an Intel Core i7 system with an Nvidia Titan card. The method was implemented in OpenGL using compute shaders. In this section, we present our results and compare to previous work. We also discuss benefits and limitations of our solution.

All results in this section were computed at full-HD screen resolution (1920×1080). Per default, we use $M = 32$ coefficients (sine and cosine alternating) per texel, which proves sufficient in practice. Figure 4.6 shows a comparison between different methods. We evaluated quality and performance on scenes with differing complexity. Our method performed in all cases below 3 ms, while still resulting in a high-quality rendering. Note that our timings exclude rendering time for the creation of the shadow map, and we only state the overhead due to the additional effect of single scattering.

The upper row in each scene set shows the result with optimal quality settings of each method, i.e., the parameters were chosen as suggested by the authors of the previously-published techniques, although, when not leading to a lower quality, we reduced the shadow-map resolution to enable a fairer comparison. Our approach reached the highest performance in all cases. For the San Miguel model (top rows of Figure 4.6 and also shown in Figure 4.7), our solution was over $20\times$ faster than the min-max-mipmap approach [Chen et al., 2011]. The low performance is a direct consequence of the complex depth map; the many leaves of the tree lead to an overhead on the min-max hierarchy (as also observed in Baboud et al. [2011]). For simple scenes, such as the terrain, the min-max structure reaches optimal performance, but our method is still 1.8 times faster (due to lower resolution shadow maps with hardware filtering). These comparisons show the main benefit of our technique: it is independent of the scene complexity and configuration, while the cost of other methods is not easy to predict. In this regard, our solution is very different from other approaches whose run time is strongly scene dependent [Billeter et al., 2010; Engelhardt and Dachsbacher, 2010; Baran et al., 2010]. Our cost is $\mathcal{O}(wh + ad)$ for a screen resolution of $w \times h$, a maximum of d view ray integration steps, and the number of epipolar slices a. Brute-force ray marching leads to d integration steps per pixel, yielding $\mathcal{O}(whd)$. Acceleration structures [Baran et al., 2010; Chen et al., 2011] can lead to an improvement of, potentially, $\mathcal{O}(wh \log d + ad)$. However, while $\log d$ is the optimal case, it can become d in the worst case. Our method shows constant performance.

The lower row in each scene set shows an equal time comparison, where we adapted the shadow map resolution of the competitors until our performance was roughly matched. The resulting images clearly exhibit artifacts stemming from discretization problems. Further, our method makes it possible to enable hardware

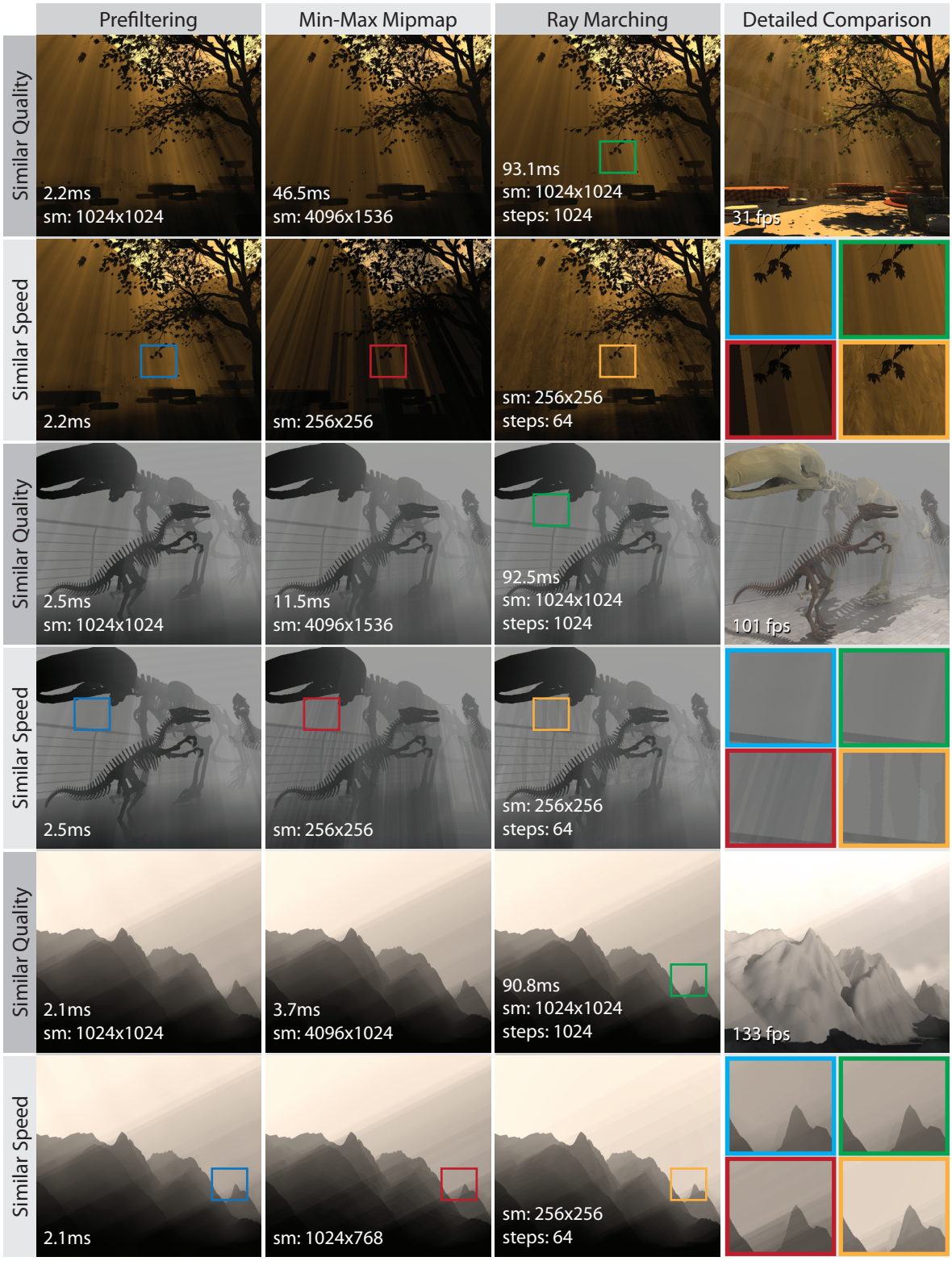


Figure 4.6: Quality/Performance comparison at 1920×1080. To compare quality, we chose the suggested settings of all methods. For same time, we reduced the shadow-map resolution. Our result compares favorably to reference quality, even for low-resolution shadow maps.



Figure 4.7: Single scattering contributes significantly to the appearance of the scene and, using our method, its computation time is low and predictable. Making it possible to meet a fixed target framerate, which makes it a good candidate to be used together with other lighting effects that share this property (here ambient occlusion, surface shadows, alpha matting).

Table 4.2: Computation time and memory consumption of our method with different parameters. As test scene we used *San Miguel* for a screen resolution of 1920×1080, where the method achieves near ground truth results with the green highlighted parameters.

Shadow-Map Resolution	Number of Basis Functions	Performance in ms			Memory in MB
		Prefix Sum	Ray Evaluation	Total	
512×1024	32	0.54	1.47	2.01	16
1024×1024	32	0.94	1.30	2.24	32
2048×1024	32	2.17	1.31	3.48	64
4096×1024	32	4.32	1.35	5.66	128
1024×512	32	0.97	1.47	2.44	16
1024×768	32	1.03	1.50	2.53	24
1024×1536	32	1.78	1.52	3.31	48
1024×1024	16	0.96	0.92	1.88	16
1024×1024	64	3.07	2.63	5.71	64
1024×1024	128	4.61	5.00	9.61	128

filtering, while other approaches do not have this option. In consequence, our results remain visually pleasing, even with lower resolution shadow maps.

A performance breakdown of our solution for San Miguel is shown in Table 4.2. The scattering cost is constant, while the prefix sum scales linearly with resolution, as expected. Changing the amount of coefficients behaves sub-linearly. This result is probably linked to the cache and texture mechanisms; as 32bit floating point computations are standard and on newer cards even 64, it is likely that the bandwidth has been increased correspondingly. Hence, our solution also seems well-suited for future hardware developments.

In theory, compared to the min-max tree of [Chen et al., 2011], we require 2.6× (32bit floats for depth, min-max) resp. 5.3× (16bit floats) more memory. Nevertheless, we can rely on hardware filtering when accessing the prefiltered basis functions, to counter potential under-sampling artifacts. This possibility even allows us to reduce the resolution significantly (usually by a factor of six; see Figure 4.6). Using this bilinear filtering is beneficial along the x-axis, which averages discrete integration steps, as well as along the y-axis, which blends neighboring epipolar slices. The latter is often desired as it avoids unnaturally sharp edges without requiring an extensive oversampling. Due to hardware filtering our method constantly achieves high performance and quality, while previous work relies on higher resolution textures [Chen et al., 2011] or customized upsampling

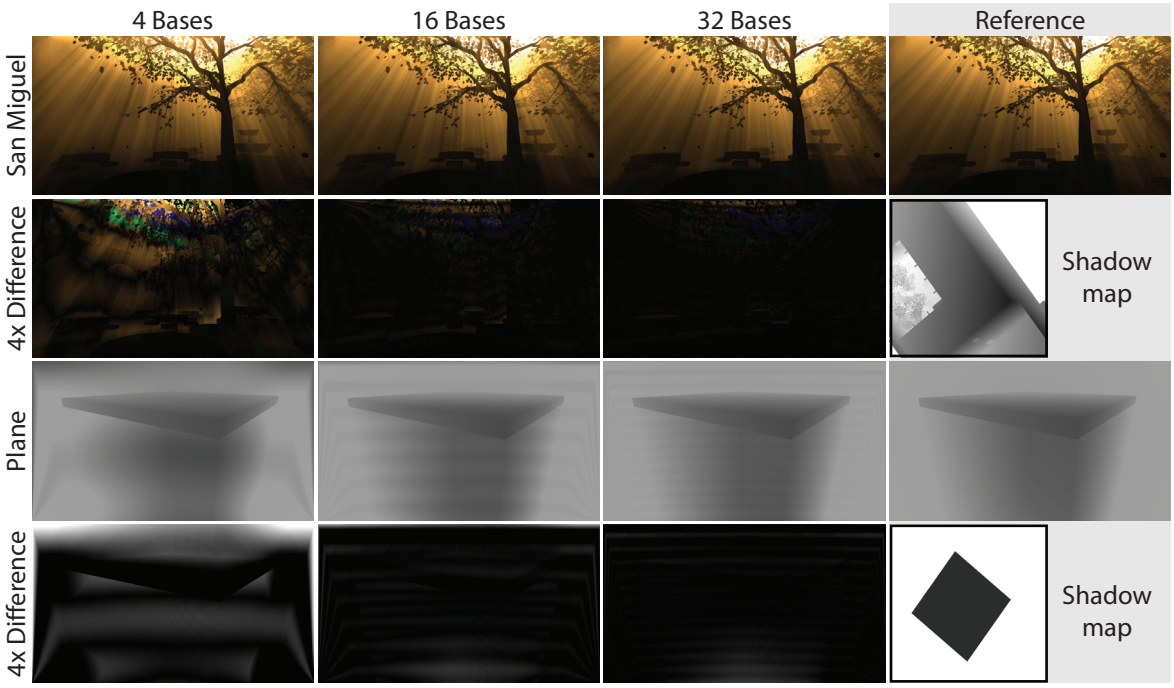


Figure 4.8: The effect of ringing dependent on the number of basis functions (scattering component of Figure 4.7). Right column: reference (ray marching) and the shadow map for the scene. Ringing is less prominent for shadow maps with increased depth variation.

strategies [Baran et al., 2010; Wyman, 2011].

Ringing is a potential issue, but is mostly hidden with 32 bases (Figure 4.8 third column). For more complex scenes that create depth variation in the shadow map the ringing artifact is even completely masked when integrating along the view ray. This can be observed for the San Miguel scene (Figure 4.8 top row), where our method achieves good quality even with a very low number of 4 basis functions. Hence, our method works well with complex geometry as can be found in today's computer games. In contrast, the masking does not occur for planar regions in the shadow map, hence, the shown 'Plane' scene with a directional light from the top is a particularly difficult case. Note that also scene details (textures, materials, surface lighting, etc.) tend to hide the artifact even further (usually completely). Our choice of 32 basis functions is conservative to also avoid any temporal artifacts.

Similar to Baran et al. [2010], we need to perform a brute-force ray marching in a small block of pixels around the light source, if it is directly visible. Epipolar geometry has low precision in these areas as view rays and light rays are nearly parallel. As result, the light source *bleeds* through the occluders or their occlusion is overestimated. This effect can be reduced if a depth warp is applied to the rectified shadow map, which basically stretches the rays. However, the effect cannot be avoided completely as shown in Figure 4.9.

4.6 Conclusion

We present an efficient single-scattering approach, with a linear, but decoupled complexity $O(wh+ad)$ of screen pixels wh and shadow map resolution ad. The prefiltering process is fast and our method is compatible to hardware interpolation. The latter allows us to use smaller resolutions for the shadow maps than in previous approaches, which results in lower memory requirements. While our method can exhibit ringing artifacts, those

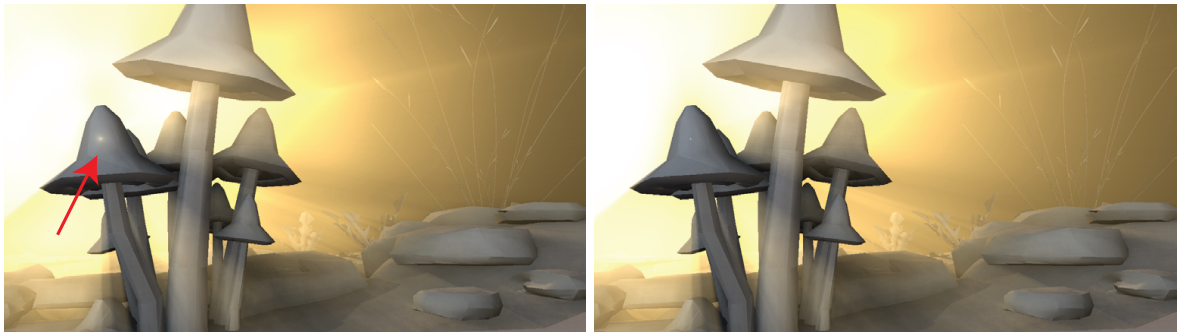


Figure 4.9: Left: Our method causes light bleeding around the epipole as the precision in the epipolar geomtry is low. Right: Reference using ray marching. This can be practically fixed by ray-marching a small block of pixels around the epipole.

partially cancel out in complex scenes, due to the increased depth variation in the shadow map. Textures and other scene details mask the effect further, making 32 bases a safe choice. In contrast to other solutions, our method delivers predictable and generally high performance combined with convincing image quality, and is independent of the scene size, complexity, or detail level. Hence, our approach is a good candidate for time-critical applications.

We also propose a new rectification method, which makes use of view and projection matrices similar to shadow-map approaches. Applied to the scene, the matrices lead to a rectified shadow map where view rays are orthogonal to the light rays and which can be used for our filtering to compute single scattering. This rectification avoids resampling of a standard shadow map, however, the procedure is only possible if the view and light rays are not parallel to each other in world space, i.e., the light source is not directly visible. In this case, we need to resort to resampling approach as previous methods.

4.6.1 Future Work

An extension of our scattering technique to multiple scattering would be highly interesting. The method is currently bound to shadow maps, hence, a generalization to multiple scattering may be possible based on virtual point lights [Keller, 1997]. Alternatively, we think that the scattering technique could even be extended to a general Monte-Carlo setting. The technique achieves high performance due to the concurrent evaluation of visibility for entire rays instead of single points. As long as this coherence in the evaluation is maintained, a generalization seems possible.

Similarly to multiple scattering, an extension to heterogeneous media is of great interest. Currently, one ingredient to high performance is the analytical evaluation of transmittance. Heterogeneous media strictly require a transmittance estimation (e.g., using ray marching), a relatively costly step. A potential avenue could be the reuse of cached transmittance that needs to be computed for other techniques (e.g., the attenuation of the directly visible surfaces), similar to the current reuse of the shadow map.

We demonstrated our the method using directional light sources; the handling of local light sources is limited to outfactored visibility, similar to previous work. However, how light-falloff functions could be accurately integrated (similar to our weighted filtering for transmittance along view rays) remains an open question.

Finally, we want to investigate whether our rectification can be coupled more closely to the actual piecewise integration to increase performance and quality. Additional control over the perspective distortion may be provided by adjusting the rectification frustum or applying an additional warping in a post-process.

Part II

Stylization and Appearance Manipulation of Volumetric Scattering

5

Related Work on Appearance Manipulation and Reconstruction

In this chapter, we will review related work on the manipulation and stylization of virtual scenes. There are many approaches to stylize natural phenomena and give control over the appearance of a scene, a few of which we will mention in the following. We give an overview, but not exhaustively explore the field. Instead, we refer the interested reader to the very recent state-of-the-art report of Schmidt et al. [2014].

5.1 Light Transport

Even for a skilled artist, it is often very difficult to tweak the appearance of a scene without destroying its plausible appearance. For this reason, many artistic modifications start from a physical simulation that is modified until the desired appearance is achieved. Unfortunately, these modifications are often complicated because physically-based parameters can be non-intuitive and the physical models complex.

Cinematic Relighting Previsualization systems [Pellacini et al., 2005; Hašan et al., 2006; Ragan-Kelley et al., 2007] generally give artists the possibility to predict the final rendering in order to support them in tasks such as lighting design and material definitions. They often assume a fixed view and static geometry, while they allow for efficient re-renderings if the lighting or materials are changed. However, the underlying calculations in these systems are physically-based, and the possibilities for abstraction and stylization are mostly restricted to the scene setup. In recent years, solutions to influence physics for the purpose of expressiveness have received increasing attention, offering artists more possibilities.

Lighting Design Early work proposed the manipulation of shadows as a more intuitive approach to control lighting [Poulin and Fournier, 1992; Pellacini et al., 2002]. Shortly later, the concept of inverse rendering was proposed to provide a goal-based design approach for light-source editing [Kawai et al., 1993; Schoeneman et al., 1993]. The latter work optimizes light parameters (intensity and color) in a least-squares sense to satisfy target images provided by the user, similar to our work that we present in Chapter 7.

A recent example for inverse rendering derives all light-source parameters based on user input for a fixed number of light sources [Pellacini et al., 2007]. Here, the optimization is non-linear and treated as a black box that uses rendering as an evaluation process to uncover the optimal parameter set. In general, it is important to realize that non-linear functions are difficult to optimize. They often suffer from local minima and, due to numerical differentiation, certain limits exist in the number of parameters that can be optimized for. Usually,

only a slow convergence is achieved. In our solution (Chapter 7), we opt for a linear optimization to ensure a fast execution of our algorithm.

We target the modification of the appearance of scattering. To reach this goal, we will also consider the modification of environmental illumination to optimize a volume's appearance instead of the volume itself. Therefore, part of our work (Chapter 7) is similar to Schoeneman et al. [1993] as we also optimize for light intensities and colors by solving a linear system. More recent previous work showed inverse rendering for environmental illumination for surface-based scenes [Okabe et al., 2007; Bousseau et al., 2011], while we propose an efficient solution for volume data. Other work focuses on the modification of spot lights [Kerr et al., 2010] and indirect illumination effects [Obert et al., 2008].

Object & Shadow Manipulation Where light is, there must be shadow. And so, there is a very thin line between algorithms that target the intuitive manipulation of light sources and those addressing shadows. Several techniques focus specifically on providing control over shadows as they play an important role in the appearance of objects in a scene.

DeCoro et al. [2007] describe an image-based system to stylize shadows of an object. They first compute a visibility map using a conventional hard-shadow algorithm. The main input to stylization is a distance field, computed on the visibility map. The following image operations then allow for the blurring or thresholding of visibility to abstract the shape of the shadow caster. Later work provides a more fine-grained control over the shape of the shadows [Mattausch et al., 2013] and supports environmental illumination [Obert et al., 2010].

Object manipulations are an alternative to provide control over shadows. Schmidt et al. [2013] propose the use of path-proxy linking, where they define invisible copies of scene objects. These can be modified using affine transformations and only affect a certain individual component of the light transport, such as shadows. The work of Mitra and Pauly [2009] addresses artistic shadows; a single occluder object is optimized to cause multiple predefined shadows.

The work presented in Chapter 6 will also make use of occluder manipulation as a tool to influence the resulting volumetric shadows that cause light shafts. Different to the mentioned previous work, we propose specialized and parameterizable manipulation methods that are interesting for stylizing scattering. To this end, we manipulate occluders using morphological operators, as recently applied by Calderon and Boubekeur [2014] to 3D point clouds. However, we work in the space of a 2D shadow map as this gives direct control over creating, removing, and enhancing light shafts.

Material Manipulation The appearance of an object is heavily influenced by its material and several techniques target the intuitive manipulation of materials [Poulin and Fournier, 1995; Ben-Artzi et al., 2006; Nguyen et al., 2012, 2013]. Although we share general concepts such as a goal-based editing using inverse rendering [Colbert et al., 2006], these works mostly focus on solid objects. As we target volumetric scattering, we will not go into further detail of this area and refer to Schmidt et al. [2014] once more.

Rendering Effects Similar to material manipulation, we only list some exemplary work that targets the manipulation of specific light-transport related effects. Systems that allow for the editing of reflections [Ritschel et al., 2009c], general on-surface signals [Ritschel et al., 2010], caustics [Gutierrez et al., 2008], and the full light transport [Schmidt et al., 2013] have been proposed. Stylization was also investigated for camera-related aspects, including focus control [Cole et al., 2006], depth of field [Lee et al., 2009], and motion blur [Schmid et al., 2010]. The general goal of these techniques is the guidance of the observer to specific regions of interest in the scene or image.

Visualization Light transport has been visualized [Reiner et al., 2012; Spencer et al., 2015] to provide insights on how light distributes in the virtual scene, eventually answering the question why a rendered image looks as it looks. Consequently, it is easier to modify the lighting, scene layout, or material properties.

In the field of scientific and medical visualization, transfer functions have been studied to provide a better understanding of volumetric data [Preim and Botha, 2013]. Hereby, these methods share our goal of appearance change [Ropinski et al., 2008; Guo et al., 2011], but to analyze the data and reveal hidden information, not to enable stylization. Barla et al. [2006] made use of transfer functions as a tool for non-photorealistic surface shading. The method shows the strong expressiveness of transfer functions, which motivated us to use them for the coloring of scattering effects (Chapter 6). Hereby, our mapping of parameters to colors also differs from scientific visualization: unlike in volume visualization [Guo et al., 2011], we do not input medium properties at a point in space, but evaluate scattering-related values along the view ray as parameters for our transfer functions.

5.2 Artistic Manipulation of Volumetric Scattering

Participating media have received little attention for appearance manipulations due to the very complex relationship between the volume's properties and their final rendering. Often, volumes are populated by means of simulation [Stam, 1999] or procedural models [Perlin and Hoffert, 1989], but such solutions remain non-intuitive and do not allow for fine-grained appearance control. Specialized approaches have been introduced for shape control [Treuille et al., 2003; McNamara et al., 2004], but these techniques do not provide an intuitive control of the volume's parameters to ensure a certain appearance under complex illumination conditions. An approach for sub-surface scattering editing exists [Song et al., 2009] and the results can look convincing, yet the scattering needs to be relatively strong and objects need to be rather opaque.

More closely related to our work is the problem of style transfer from a photograph to a 3D scene. The work of Dobashi et al. [2012] reconstructs the parameters of a cloud from a single photograph. In contrast to our solution (Chapter 7), the authors aim at estimating a few global volume parameters in a non-linear optimization scheme instead of estimating per-voxel properties. As the problem is heavily under-constraint, additional priors are required and the results are of relatively low quality, yet convincing.

Artistic stylization of volumetric scattering is addressed by Nowrouzezahrai et al. [2011]. Their system lets the user modify individual light beams by influencing shape, fall-off, and color. The color definition is optimized to find a plausible mapping to properties of the participating medium. To do so, the volume-rendering process is split into multiple functions that are changed individually instead of a global optimization procedure. Treating beams individually can be an advantage, but global control becomes more time-consuming and difficult. In contrast, our methods provide control over the color that is observed in the image, i.e., we consider all beams at once, effectively providing manipulations at global instead of local scope. Further, all changes of their method are allowed to be non-physical, even curved rays are possible. To display the artistically-designed beams, the method of Nowrouzezahrai et al. relies on a special rendering method. Similarly, We will also propose non-physical modifications to scattering that require special rendering techniques (Chapter 6). Alternatively, we enable the reconstructing volumetric properties (Chapter 7) that obey physical rules and can be rendered with any physically-based technique.

Recently, Hašan and Ramamoorthi [2013] presented an approximation to quickly re-render an image after a change of the single-scattering albedo of a volume. Different to our work, they support full light transport, including multiple scattering. While they also target the manipulation of volumes, they focus on speeding up re-rendering with novel parameters. Instead, our work investigates intuitive design approaches to manipulate

scattering and volume properties. Further, their approximation is limited to dense volumes and requires a costly and memory-intensive preprocess.

5.3 Reconstruction of Volumetric Effects

Producing realistic volumetric data sets is known to be difficult, which is one of the reasons for the existence of various methods that capture properties of natural phenomena; flames [Ihrke and Magnor, 2004], smoke [Hawkins et al., 2005; Ihrke and Magnor, 2006], or refractive elements [Ihrke et al., 2005; Atcheson et al., 2008]. Often, the process involves tomographic reconstruction [Ihrke and Magnor, 2004, 2006; Atcheson et al., 2008]. It is surprising, that even a small number of views (8 to 16) often leads to a convincing volumetric description. This is one of our motivations to define volume properties in an image-based fashion (Chapter 7). Nonetheless, previous work employed simple image formation models and restricted the capturing process to a single phenomenon such as emission [Ihrke and Magnor, 2004, 2006], or refraction [Atcheson et al., 2008]. In contrast, our method provides control over the shape (extinction coefficient), emission, and single-scattering albedo of a volume. One exception is the work by Linţu et al. [2007], who attempt to recover emission and absorption simultaneously to reconstruct a nebula from astronomical observations. In contrast, we target to artistically control and modify volumes to achieve a certain appearance under complex illumination.

5.4 Fabrication

Our work also shares some characteristics with recent developments in fabrication. Previous work showed solutions to built physical shapes that have a predefined way of interacting with light in the real world. These methods make it possible to produce objects that cast certain shadows [Mitra and Pauly, 2009; Baran et al., 2012], caustics [Papas et al., 2011, 2012], or exhibit a certain surface reflectance [Weyrich et al., 2009]. Such work also relates to light-field display technology [Lanman et al., 2011; Wetzstein et al., 2012], 6D displays [Fuchs et al., 2008], or other special displays [Holroyd et al., 2011; Wetzstein et al., 2011]. The displays make it possible to achieve a specific appearance when lit uniformly from the back and observed under particular viewing angles. In contrast to our work (Chapter 7), these methods consider only a small number of three to five layered light modulating planes, which are viewed from a well-defined viewing zone. Further, the approaches involve only one type of effect, such as absorption [Holroyd et al., 2011; Wetzstein et al., 2011] or change of polarization [Lanman et al., 2011]. Our goal is to give control over the appearance of volumes having a full resolution along all three spatial dimensions. We consider complex and full surround views and complex environmental illumination. Further, we consider many properties of the volume, which can be easily defined for our rendering context, but that would lead to problems for physical object manufacturing. One particular example are refracted rays that we can trace through a known volume.

Finally, recent work enables the scanning [Gkioulekas et al., 2013] and fabrication [Papas et al., 2013] of homogeneous volumetric materials such as milk, blue curacao, soap, or chocolate. Similar to our work, an optimization procedure is employed to reconstruct all physically-based volume parameters. However, they have to consider multiple scattering and cannot apply additional constraints, leading to a highly complex and non-linear reconstruction. Instead, we only consider single scattering, but support heterogeneous media (Chapter 7).

6

Stylization of Volumetric Scattering

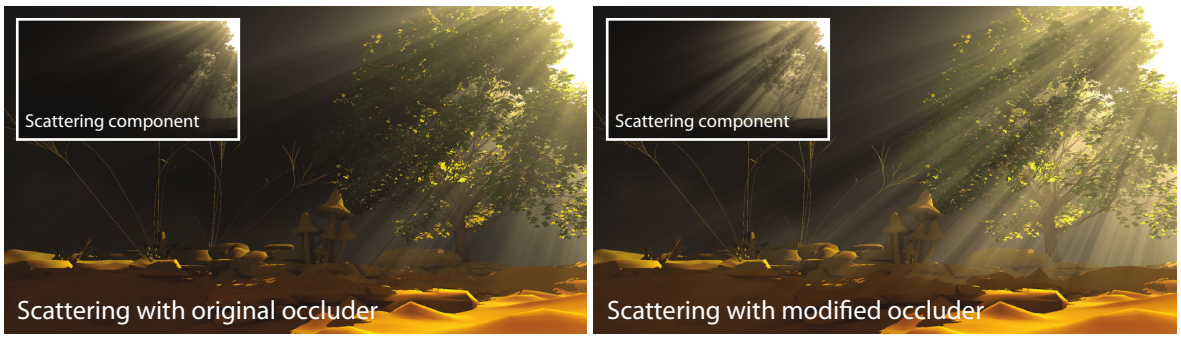


Figure 6.1: Example of our stylized scattering. Left: single scattering using the original geometry of the scene. The leaves of the tree block most of the light, causing only a subtle scattering effect. Right: scattering created by occluder manipulation. Using our system, an artist can easily add holes into the shadow map of the tree, causing an increased amount of and more interesting scattering effects. While physically incorrect, it is not visible to the viewer that the right image uses fake occlusion information. Surface shadows are created from the unmodified shadow map.

Participating media can drastically influence the appearance of a scene. The effects resulting from light scattering in these media often add realism and spatial cues for a better scene understanding, or can serve artistic purposes. Chapter 4 focused on the efficient rendering of a dominant effect that results from scattering: crepuscular rays, which we will call light shafts in the following. We have seen that shafts of light are caused by light rays that illuminate optically thin participating media, while neighboring rays are blocked by an occluder. Occlusion is the key element that causes those shafts to form and become visible. In nature, this can be observed if there is an opening in the clouds. While visually pleasing, the underlying physical processes (see Section 2.4.4) makes the resulting appearance hard to predict.

There are several examples that inspired our work. Two of them are shown in Figure 6.2, which contain physically implausible light shafts. Animation movies, comics, and concept art often tend to add exaggerated light shafts to emphasize characters. The harshness of the lighting can be particularly striking, as most real-world light shaft boundaries appear smooth.

Another example consists in abstractions of the shape. Consider a focus object that is illuminated by a directional light such as the sun. The light shafts are often kept very simple, e.g., even if foliage of surrounding trees would result in a complex pattern, in many cases artists use a simpler representation. Inversely, as light shafts can be visually pleasing, they are sometimes added as well.



Figure 6.2: Non-photorealistic scattering used in movies and concept art. Light shafts are an important scene element for the image on the left (Frozen by Disney). The shafts have unnaturally strong edges, which makes them more visible and increases the tension of the scene. Concept art and comics usually use a very simplified scattering, often in the form of stripe-like elements such as shown in the image on the right (Big Hero 6 by Disney). Image credits and copyrights Disney.

6.1 Contributions

Our work offers new ways of influencing the appearance of light scattering to achieve similar effects by employing two manipulation concepts. First, by changing the occluders in the scene, our algorithm is able to add, remove, and sharpen light shafts. Figure 6.1 illustrates an addition of light shafts to brighten the scene and add more detail, leading to a significant change in appearance. Second, our algorithm provides simple controls to achieve various styles and expressive changes through the use of transfer functions, which are easily modifiable to influence the mood of a scene.

As with most artistic tools, it is crucial that users can change the parameters interactively to explore possible choices. For this reason, we focus on stylization and scattering methods that can be computed in real time, and enable immediate feedback when parameters are changed. Our approach is able to consistently stylize the scattering effects in an entire 3D scene, making it suitable for animations. Furthermore, its high-level definitions help in generalizing the settings and the transfer of a general style to various scenes of different geometry, camera, and light settings. Also, we enable artists to create very specific styles for controlled animated scenes, such as cut-scenes in games. Results for these methods are showcased in the supplemental video of the publication [Klehm et al., 2015].

Specifically, our work makes the following contributions:

- light-shaft addition, removal, and enhancement using image-based occluder manipulation;
- light-shaft color modification via user-editable transfer functions based on view ray properties;
- light-shaft animation by dynamic occluder manipulation, and key-framed transfer functions for animated scenes.

6.2 Stylized Single Scattering

Our method consists of two major directions to perform scattering stylization, which can also be combined. The first strategy modifies occluders in order to influence their scattering behavior, i.e., light shafts are enhanced, added or removed. The second approach consists in the definition of a transfer function that can be used to

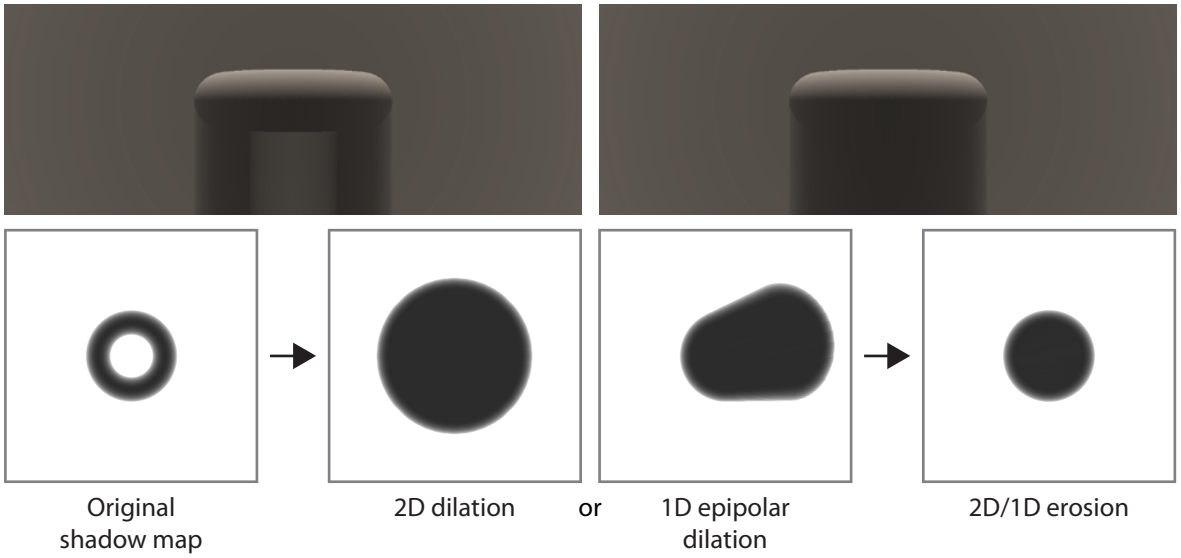


Figure 6.3: Morphological filtering. Top row: rendering with unmodified and hole-filled shadow map for a simple torus scene with directional light coming straight from the top. Bottom row: shadow maps, left to right: unmodified input; 2D dilation of input; 1D epipolar dilation of input; correspoding 2D or 1D erosion, applied on the respective dilation result. Both, 2D and 1D, erosions give equal results for this particular scene.

drastically influence color, brightness, and contrast. The idea is to rely on values derived from the view ray (e.g., scene depth) to define the final output color.

For both of these techniques, we assume the same scattering model as used in Chapter 4: we only consider homogeneous media and ignore multiple scattering. We will also make use of average visibility of a view ray as defined in Section 4.2.1. Please note that average visibility can be computed efficiently by a number of recent techniques [Baran et al., 2010; Chen et al., 2011; Wyman, 2011], including, but not limited to our method that we presented in Chapter 4.

6.2.1 Occluder Modification

The main observation is that the appearance of single scattering in a scene largely depends on the number, size and contrast of light shafts. They become visible due to differences in the average visibility between neighboring view rays (i.e., screen pixels). These differences are often caused by openings in the occluder; that is, holes through which the light can shine, such as the opening in a cloud. Our system enables artists to modify the scattering by enhancing, adding or removing light shafts. The main idea is to edit the shadow map that is used to capture the occluders in the scene. In the following, we will present the various modification options.

Hole Filling Physically-based scattering can sometimes produce unwanted effects. For instance, while the top image in Figure 6.11 is visually pleasing, there are little light shafts between the palms that one may consider distracting. To appreciate the image more, an artist may want to remove distracting details. Remembering that they are caused by small openings in the occluders, we want to close these holes, effectively simplifying the overall appearance of the light shafts. This reduces the emphasis of the scattering for an object that may otherwise exhibit a complex light-shaft appearance, such as the palm tree.

To fill holes in the occluders, we make use of an image-based approach in which we directly modify the shadow map used for the scattering evaluation. A simple solution for hole filling is the use of 2D morphological

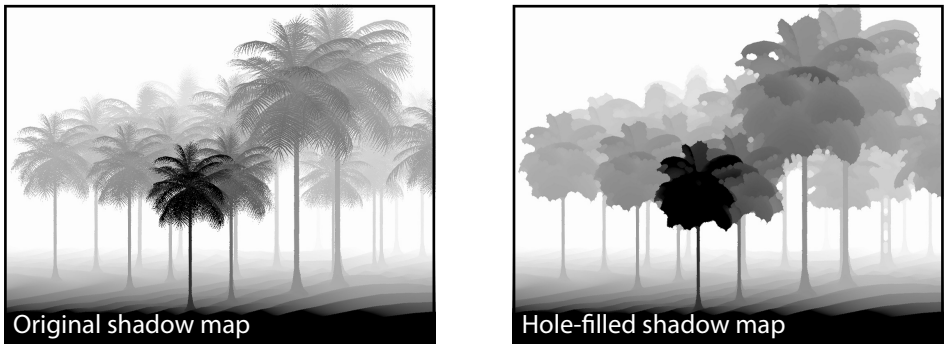


Figure 6.4: Applying hole filling on the shadow map of the palm tree scene of Figure 6.11. Left: original shadow map. Right: resulting shadow map using a kernel size of 10×10.

filters. More precisely, a closure operation can be applied, which consists of a dilation followed by an erosion, both elementary operators. A dilation in the shadow map replaces a value by its minimum in a certain neighborhood, whereas an erosion replaces the value by the maximum. All holes that are removed in this way, also lead to the elimination of the corresponding light shafts. For illustration, an exaggerated example is shown in Figure 6.3, where we remove the entire hole in the torus, leading to a simpler light-shaft appearance.

One important factor is the neighborhood to be considered around each pixel, often referred to as the *structural element* or *filter kernel*. Traditionally, a box or circle is used, which we apply as well, additionally giving the user control over the size of the element, which defines the strength of the hole-filling process. The result of applying this technique to the shadow map of the palm tree scene is shown in Figure 6.4.

Silhouette Enhancement We have just shown how to remove attention from the scattering, but in some cases the opposite is desired. At times, the scattering can be very subtle, while an artist may want more prominent light shafts, e.g. as in the middle image of Figure 6.11. To remedy this, an approach very similar to hole filling can be taken to *enhance* light shafts caused by the silhouette of an object. The idea is to extrude objects along the view rays, increasing their "thickness". We achieve this by a 1D dilation of the shadow map away from the epipole, i.e., the camera position projected into the shadow map (see Figure 6.3 right). This way, the newly created blockers are always hidden by the object itself, as the camera only sees the first surface. However, the object's volumetric shadow is increased.

The 1D epipolar dilation works as follows. For each texel t we construct a 1D kernel given by the 2D line from t towards the epipole. A standard dilation on the shadow map simply computes the minimum of all samples within the kernel. However, this does not satisfy the required extrusion from the camera position, as the min filter effectively extrudes points in the plane orthogonal to the light direction. It needs to consider the changing z-coordinate along the view ray, which forms a sloped filter kernel. For this, we modify the depth of t given an input sample s as follows:

$$z_t' = \min(z_t, \frac{\mathrm{dis}_{2D}(t,l)}{\mathrm{dis}_{2D}(s,l)}(z_s - z_l) + z_l), \tag{6.1}$$

with $\mathrm{dis}_{2D}(t,l)$ denoting the distance between texel t and epipole l as projected in 2D on the shadow map, and with z_t and z_s the values of t and s in the shadow map, respectively, and z_l the z-coordinate of the epipole l. This boils down to a mix of z_s and z_l modulated by the distance ratio of t to l compared to s to l, which, as per the definition of a dilation, is used only if it is lower than the current lowest depth value z_t. The process is illustrated in Figure 6.5.

When applied, the actual enhancement of the light shafts caused by the silhouette of the object is two-fold.

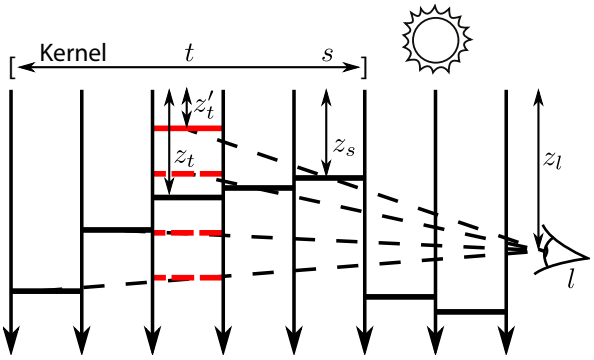


Figure 6.5: 1D epipolar dilation. Areas between the parallel light rays represent texels and black bars denote their depth in the shadow map. A 1D kernel (of size 5 in this example) is constructed for texel t with depth z_t. Given the epipole l with depth z_l, the whole kernel is sampled and Equation 6.1 is applied. The minimum value z'_t is in this case found for the sample s with depth z_s.

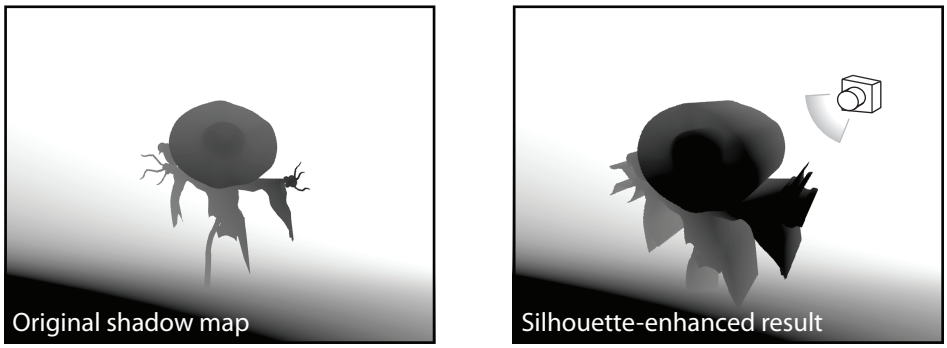


Figure 6.6: Applying silhouette enhancement on the shadow map of the scarecrow scene of Figure 6.11. Left: original shadow map. Right: resulting shadow map using a kernel size of 100; the scarecrow is visibly extruded in the view direction.

First, the 1D epipolar dilation fills holes, removing small light shafts. Second, as the extrusion process is aligned with the view rays, it increases contrast between neighboring pixels due to sharper boundaries. Occluders become more drastic and will block more light as they are effectively larger. Figure 6.6 shows the resulting shadow map for the scarecrow scene.

Hole creation Silhouette enhancement makes the light shafts more prominent, but for that to have an effect, sufficient light shafts need to be present. Situations can occur where this is not the case, like for the bottom image of Figure 6.11. Here, only a few light shafts fall through the flowerbed, creating subtle scattering no matter how much silhouette enhancement is applied. An artist however may want more light to burst through the flowers. For this reason, besides removal, we also offer a solution to *add* additional light shafts. In contrast to the previous hole-filling operation, we instead create random holes, the result of which can be seen in Figure 6.1. As before, we work directly in image space by manipulating the shadow map. Each object for which this operation is applied is rendered in a separate shadow map that then undergoes the hole creation process. Basically, a hole is defined by pushing the depth values at its location to one, which corresponds to the far plane. The modified depth maps are then composited with the rest of the scene to yield the depth map for the scattering computation. Alternatively, one could discard fragments of these objects during rasterization, which makes multiple shadow

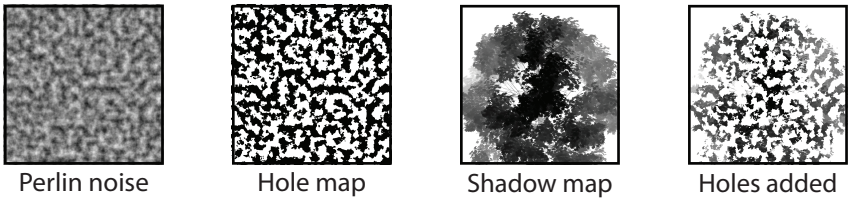


Figure 6.7: Random hole creation, results of which are shown in Figure 6.1. Left to right: Perlin noise with user-defined frequency; thresholding with user-defined hole probability; original shadow map of tree; shadow map with added holes from hole map.

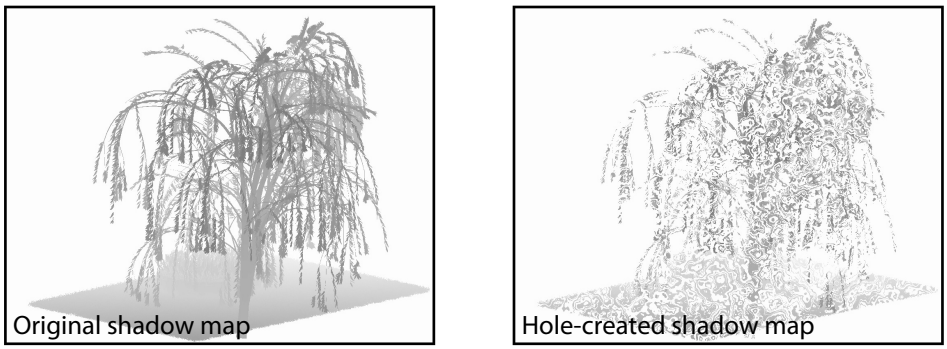


Figure 6.8: Applying animated hole creation on the shadow map of the flowerbed scene of Figure 6.11. Left: original shadow map. Right: resulting shadow map. Note that in this case the hole creation was also applied on the ground, which may give strange results depending on the viewpoint. As mentioned before, we simply solve this by using two shadow maps and compositing them later on when necessary.

maps unnecessary.

To give control over the hole-creation process, we provide the user with a set of parameters consisting of the average size and the density of holes. In order to avoid perfectly uniform holes, for which the resulting light shafts can look too regular, we make use of Perlin noise [Perlin, 1985]. By using a thresholding operation, this scalar noise can be transformed into a binary mask that will be used as the hole map, exhibiting randomization in shape. As Perlin noise is easily parameterizable, users can intuitively steer the hole creation and obtain the desired properties; the number and size of the holes can be influenced via the number of octaves and threshold parameter. The binary mask that is used as the hole map has the value $H(t)$ at the texel with 2D texture coordinates t, and is given by:

$$H(t) = \begin{cases} 0 & \text{if } N(tgf) \geq h \\ 1 & \text{otherwise} \end{cases} \tag{6.2}$$

with h the user-defined hole probability or threshold value, f the user-defined frequency, g the Perlin-noise gradient-grid size. $N(q)$ with $q = tgf$, is given by:

$$N(q) = \sum_{i=0}^{o-1} p^i P(q) \frac{1}{\max(1, o-i-1)} \tag{6.3}$$

with o the number of user-defined octaves, p the user-defined persistence and $P(q)$ classical 2D Perlin noise.

The process is illustrated in Figure 6.7, where a frequency of $f = 0.3$ and a persistence of $p = 0.5$ are used to create 5-octave Perlin noise, which is then thresholded using a hole probability of $h = 0.5$, resulting in the hole map shown in the figure.

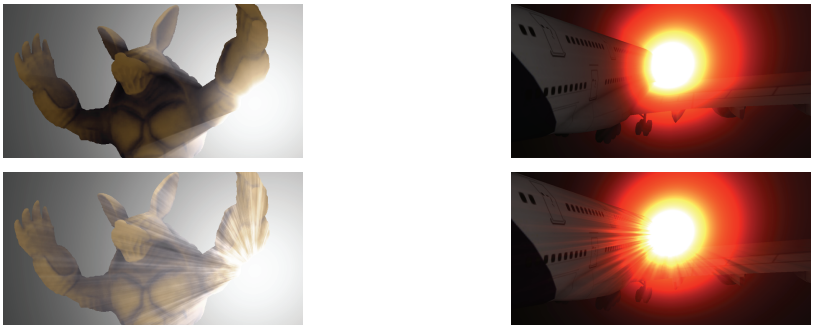


Figure 6.9: Hole creation applied to solid objects. Top: the original image. Bottom: image obtained by creating holes in the shadow map. Left: the creation of a halo around a character. Right: the illusion of motion.

Using Perlin noise enables easy animation of the hole creation process. To do so, we use a simple two-step lookup. First, a time-based offset is added when sampling the noise. Instead of using the result directly, we use it to re-sample the noise again with an added rotation, to prevent the animation from being a simple translation, and effectively randomize the movement. This can simulate the effect of leaves moving in the wind or fake the notion of movement through participating media. When applied to the shadow map of the flowerbed scene, it produces the result shown in Figure 6.8.

The hole-creation process is not suitable for all objects. Solid objects for instance can appear unrealistic, and the technique should generally be applied to less recognizable shapes, such as foliage, flowers or other detailed geometry. Still, it can be applied to solid objects in certain cases to create a sort of halo around a character or simulate the idea of motion, as illustrated in Figure 6.9.

6.2.2 Transfer Functions

An effective way of stylizing scattering is to influence the overall appearance by applying a transfer function that describes the mapping of properties of a view ray (effectively a pixel), to the scattering component's output color. In order to keep the definition of this transfer function simple, we decided to focus on mapping two parameters to a color. Consequently, the transfer function is defined by a 2D texture, similar to the X-Toon approach [Barla et al., 2006].

Remembering a view ray is uniquely defined by its underlying pixel, an artist can easily influence the entire scene appearance in a consistent and effective way by modifying the transfer function. As an example, imagine an artist wants a certain set of pixels with similar properties that are currently white, changed to an orange color for stylization purposes. With a simple modification of the transfer function, this becomes easy. Furthermore, using a transfer function, the general atmosphere due to scattering can be easily transferred from one scene to another.

In practice, we use the average visibility along the view ray and its linearized depth as parameters for influencing the scattering component's color. With those parameters, the result looks physically plausible due to the direct link of the parameters with the actual visibility in the scene. We also experimented with other parameters, such as the average position of visible samples along the view ray, the angle between the view ray and light direction, and the angle between the image x-axis and pixel-to-projected-light direction, but these did not produce meaningful results for the cases we considered, and are therefore not included in our results. Note that we use values along the view ray, i.e., we rely on 3D information, as otherwise, the stylization would appear to be a 2D overlay (shower-door effect), which would become directly apparent for animated cameras.

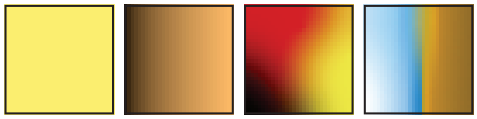


Figure 6.10: Different layer types supported by our transfer-function editor: solid, gradient, diffusion, and image layers.

To create a transfer function, we enable artists to interactively design the texture in our framework to explore the various possibilities. This is advantageous over a general image editor as it immediately shows the resulting image, which can be difficult to envision when creating the transfer function beforehand. The on-the-fly editing of the transfer function texture uses a painting utility based on layers. In our prototype, solid colors, gradients, diffusion layers and images are supported (see Figure 6.10), which are optionally blended with each other to produce the final transfer function texture, using multiplicative, additive or alpha blending.

Diffusion layers contain constraints, consisting of a position in the transfer-function texture, a color, and an alpha value. The user can place constraints at any position on the texture, after which they are diffused throughout the layer. One constraint would diffuse to a uniform color for the whole layer, two produce a gradient, while more can create more complex color combinations, like the third image in Figure 6.10. For stacked layers, the alpha blending is guided by the constraints' alpha values.

However, specifying constraints only in the 2D parameter space of a transfer function might not always be intuitive. In order to directly define a desired scattering result at a point in the scene, an artists can simply select a location by clicking in the scene to create a 3D constraint. The parameter space position of that constraint is then computed by projecting its world space position to screen space, querying the pixel's underlying view ray parameters and using these to obtain the new position in the parameter space of the transfer function.

The selected colors of the constraints are diffused throughout the layer to create a smooth transition in the scene. This results in all pixels in the scene with similar parameters to change accordingly, making it an effective tool for a more global control over the stylization of scattering for similar pixels. 3D constraints are also beneficial for dynamic viewpoints or geometry changes, as we optionally allow the constraint to follow the position of a selected object rather than staying fixed in world space while the camera moves.

The stylization so far is not expressive enough for modifying the scattering appearance of predefined scene animations. Therefore, we propose key-framed transfer functions to produce stylized animations. This can be beneficial, for instance, for an in-game cut-scene with a known camera path. Distinct transfer function's can be defined for positions along this path, with linear interpolation between them leading to smooth transitions.

Finally, the stylized scattering can be combined with physically-based scattering behavior. For instance, the transfer function can be monochrome and the stylized scattering modulated by the light source color, which ensures that surface lighting and scattering remain consistent, as in Figure 6.13. This enables artists to perform more subtle stylization, like making the scattering more or less dominant. Note that we use the Henyey-Greenstein phase function for the scattering as it makes the result more physically correct and believable. Nonetheless, if desired, the phase function can also be set to a constant.

6.3 Results and Discussion

Our method works in image space and directly modifies the shadow map that is used for the scattering computation. This has several benefits over working in object space, as the performance does not depend on

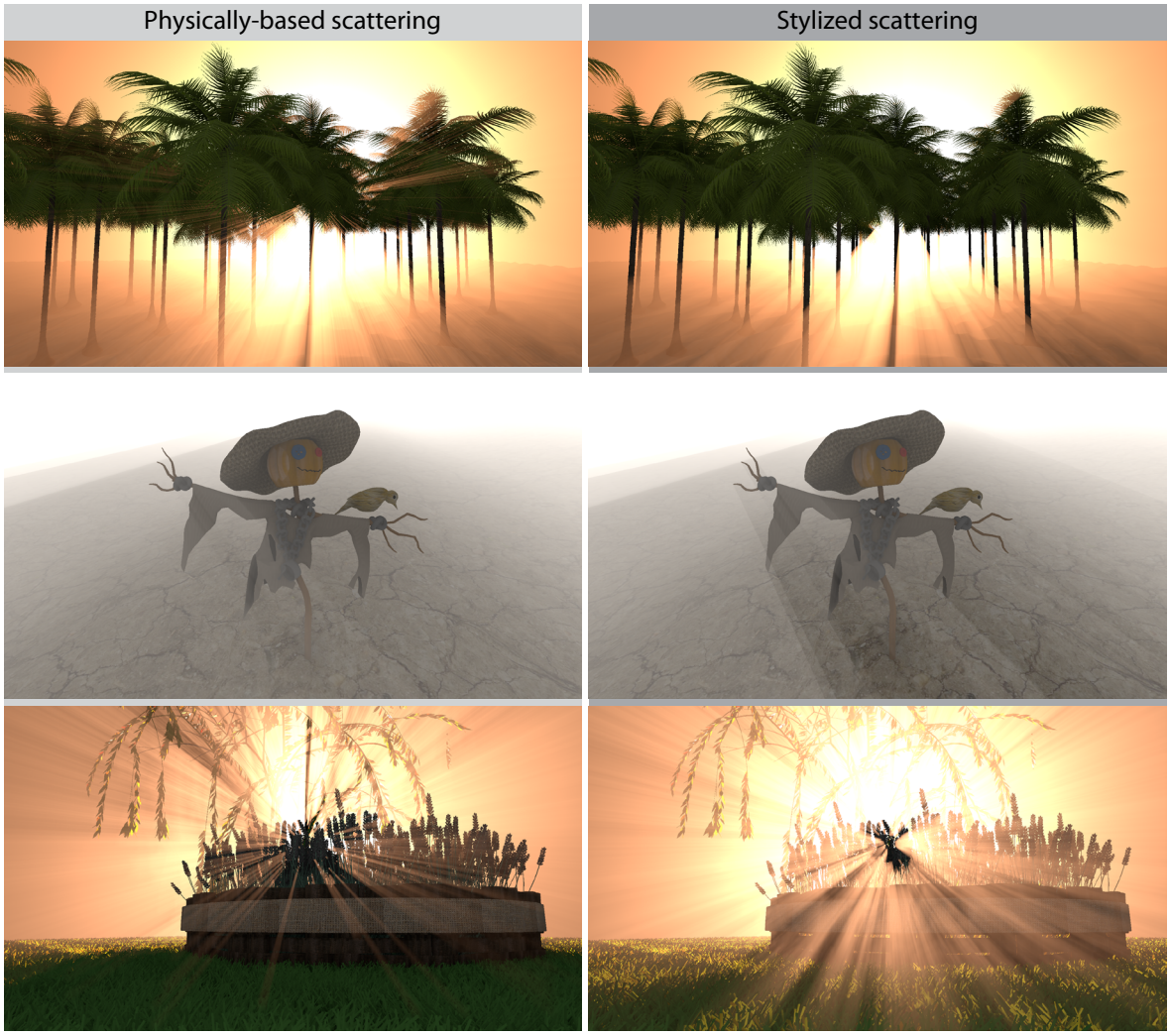


Figure 6.11: Stylized scattering applied on a palm tree, scarecrow and flowerbed scene using our occluder manipulation tools. Left column: original images with physically-based scattering. Right column: results using our occluder manipulation stylization tools. From top to bottom: hole filling, silhouette enhancement and hole creation.

Table 6.1: Performance (in ms) of our prototype for the scene shown in Figure 6.1, rendered in full HD for different shadow map (SM) sizes. Measurements for hole creation include Perlin noise creation as well as the SM modification. A transfer function (TF) for color stylization is indexed using the depth map of a G-buffer as well as the view ray average visibility. Using an acceleration method like the one proposed in Chapter 4 is required as naive ray marching is an order of magnitude slower (67.8ms for full HD and 1024 marching steps).

SM Size	SM Creation	Holes	G-buffer	Scattering Vis.	TF
512^2	1.3	0.1	4.0	1.6	0.2
1024^2	1.6	0.3	3.9	2.5	0.2
2048^2	4.0	1.4	4.1	4.6	0.2

Table 6.2: Performance (in ms) for hole filling for the tree used in the scene shown in Figure 6.1, rendered in full HD for different SM and kernel sizes. As we work in image space, the kernel size needs to be adapted to the SM size as well as the content. The right column shows how many holes are filled for the given SM and kernel size.

SM Size	2D Closure		1D Epipolar Closure		Filled
	Kernel	Filtering	Kernel	Filtering	
512^2	11	1.1	15	0.6	All
512^2	21	3.7	30	1.1	All
512^2	41	13.3	60	1.9	All
1024^2	11	4.5	15	1.7	Most
1024^2	21	15.4	30	2.9	All
1024^2	41	56.7	60	5.2	All
2048^2	11	18.0	15	5.4	Many
2048^2	21	60.9	30	9.3	Most
2048^2	41	222	60	16.8	All

geometric detail or scene complexity. Furthermore, image-space techniques are efficient on today's hardware and are well-suited for parallel execution. Also, operations such as hole filling are easy to perform because the neighborhood of objects is automatically resolved. Tables 6.1 and 6.2 show performance measurements on an Nvidia Titan in full HD resolution for occluder manipulations and pipeline steps for computing and coloring light shafts.

Figure 6.11 demonstrates the effect of occluder manipulation. Performing a closure operation to fill holes in the shadow map simplifies the scattering appearance, effectively removing the unwanted little light shafts in the palm tree scene. Note that the 2D filter can cause occlusions to happen in mid-air due to newly created, floating occluders. However, there is little visual impact due to the fact that we integrate along the view ray. In theory, this can produce slightly darker areas that are just distinguishable. A solution is to extend a 1D epipolar dilation by a subsequent erosion (see Figure 6.3, right), rather than working in the 2D space. However, this is not essential for any of our scenes. For our palm trees, we do not observe any artifacts, and the scattering appearance is correctly simplified.

In contrast, an artificial extrusion of the scarecrow away from the observer (1D epipolar dilation) enhances edges in the scattering (Figure 6.11). It causes almost a feeling of fright and emphasizes the object strongly. The resulting effect is similar to the harshness of such effects, often used in comics, to illustrate activeness. As a side effect, the extrusion also darkens the image slightly, due to the larger occluder; a transfer function could compensate for this (a combination of occluder manipulation and use of of transfer function is shown in Figure 6.13).

Applying the morphological operations to animated scenes requires special care, as popping artifacts may be introduced if shadow-map holes change in size due to the animation. This can occur when the closure kernel

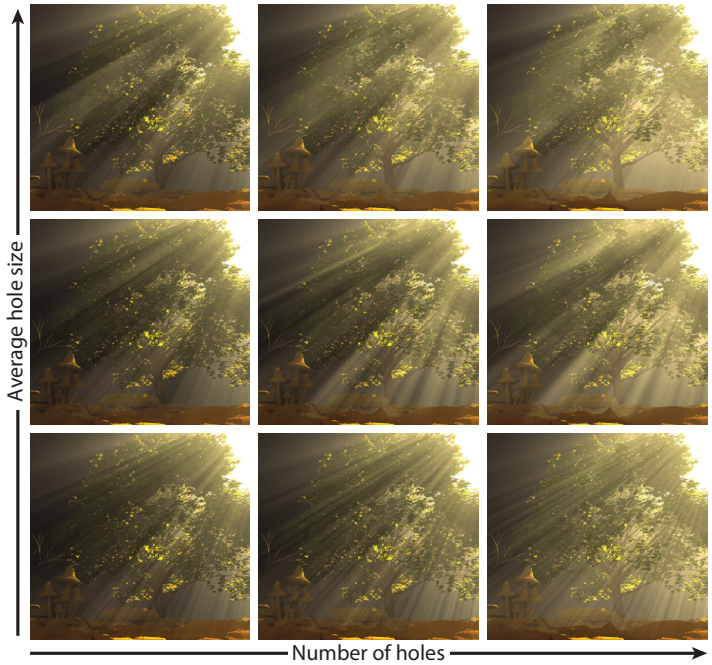


Figure 6.12: Effect of noise parameters on the resulting scattering.

size is not chosen appropriately. If wanted, the kernel size can be key-framed according to the animation.

While silhouette enhancement enhances light shafts, hole creation enables artists to easily make the appearance of the scattering more complex. The flowerbed scene in Figure 6.11 shows that much more light falls through the flowers after our hole-creation process has been applied. As discussed in Section 6.2.1, the Perlin noise allows for various parameters when setting up the hole creation. To demonstrate this, Figure 6.12 shows the effect of different parameters on the example scene of Figure 6.1. Very small holes (bottom row) can be reduced in size until they average out due to the integration along the view ray, while very large holes (top row) can drastically simplify the scattering behavior. The control supplied by these parameters make it easy to transition between more physically-based scattering with the original occluders, and the more exaggerated alternatives. Note that the axes are not absolute; the number of large holes is less than for smaller holes in the same column. The scene can also be animated, e.g., imagine a sunrise for Figure 6.1. The inherent complexity of the tree masks any potential artifacts that one may expect from the 2D hole map, which is fixed to the light space. In general, we expect the hole creation to be applied to objects of similar complexity, for which the shadows are not easy to predict.

Finally, the turtle scene (Figure 6.13) shows an example for a combination of occluder manipulation and stylization via a transfer function. The initial shot exhibits an unlucky overlap of occluders (head, body, fins). An artist may want to simplify the scattering to put more emphasis on the actual object in the scene and remove attention from the light shafts. The 2D morphological filtering closes the hole between the fins of the turtle and smoothes the occluder's silhouette, which gives the light shafts a more simplified appearance. Then, the applied transfer function (Figure 6.13, right) makes it possible to reduce the shadows, while keeping a plausible scattering appearance. To take it further, a transfer function can even be used to remove the potentially distracting dark stripe on the turtle's body, caused by the shadow of the head, because these pixels have similar average visibility and scene depth.

Transfer functions make it possible to quickly achieve strong changes in the overall appearance of the

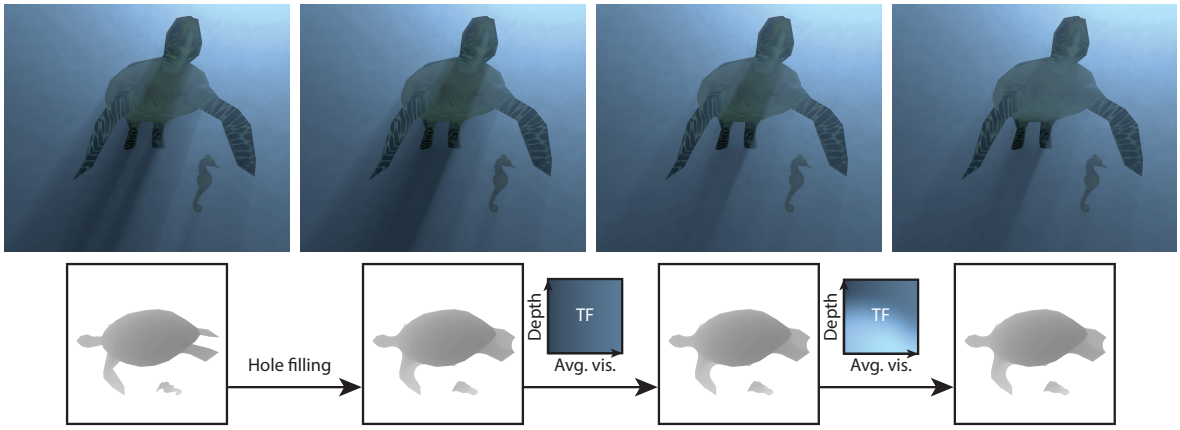


Figure 6.13: Combined use of occluder manipulation and a transfer function. From left to right: the original image; hole filling unifies the fins of the turtle; a simple transfer function is applied to reduce the darkness of the shadow; another transfer function is used to remove the dark patch on the turtle's body.

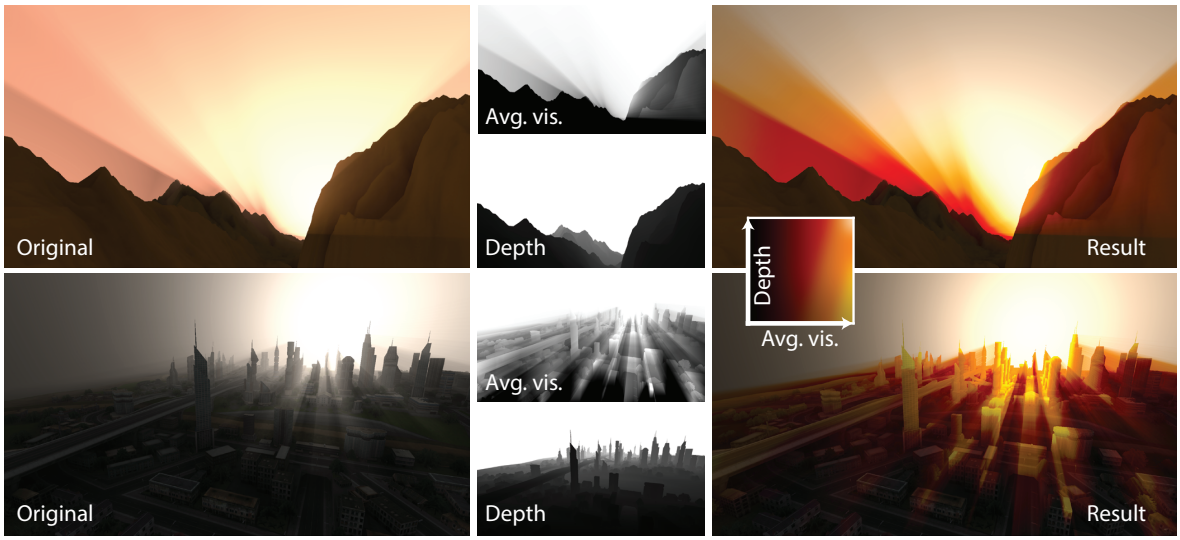


Figure 6.14: Expressiveness of transfer functions. Physically-based scattering (left) is stylized (right) using a transfer function (inset), which is parametrized by the average visibility and linearized depth of the view rays (center).

scattering process, independent of surface, light, or medium parameters. These modifications enable artists to easily redefine the mood of a scene, as illustrated in Figure 6.14. Here, the transfer function creates an alarming atmosphere in the scenes by adding multiple colors with strong edges between them, which causes quantization effects in the resulting scattering. Additionally, while physically-based scattering has an exponential fall-off with respect to the length of the view ray (see Equation 2.20), the transfer function used here abandons this fall-off, and more scattering is produced close to the camera. Finally, the figure also illustrates the possibility to pass over a given style from one scene to the next.

6.4 Conclusion

We have presented strategies that let artists stylize volumetric single scattering in various expressive ways. The stylization can be easily transferred to different scenes and also works for animations.

Our image-based occluder manipulations modify the complexity of the scattering appearance. Light shafts are added using hole creation, which is randomized but controlled with a few parameters. Simplification of light shafts and silhouette enhancement are achieved by morphological filter operations.

We also propose the usage of transfer functions to control the final appearance and to transport the general mood of scattering effects. By parameterizing the transfer function with the average visibility and linear depth, results look physically plausible, yet expressiveness remains. Transfer functions are modified by artists interactively, where we enable them to stylize the scattering for similar view rays and to animate the atmosphere of a scene over time.

Our solution comes at a low computational cost, which makes it ready to use for real-time applications, enabling a quick exploration of the various settings. Our work makes a first step in the direction of a general scattering stylization, and its ability to reproduce various scenarios that would otherwise require laborious manual tweaking, makes it an interesting addition to the artistic toolbox.

6.4.1 Future Work

Our current occluder-manipulation techniques modify the shadow map, hereby being largely independent of the actual scene objects. Future work could investigate a tighter coupling of scene objects to light-shaft manipulations, i.e., we imagine that specific objects may manipulate the scattering in an artistically controlled manner.

We are also planning to integrate heterogeneous media. An approach that would be compatible with the current technique would be *soft* heterogeneity, i.e., the extinction coefficient is constant, but the volume has non-constant *intensities*. Such manipulations would be highly interesting as this kind of heterogeneity is already used in current computer games. Artistic control over the scattering intensities, the amount of heterogeneity, and changes in time could be provided.

Finally, we believe that the proposed transfer functions are effective for modifying scattering colors, so we want to investigate other source dimensions that map to scattering colors. Furthermore, we think that the definition and editing of the functions can be extended. We think that the functions could be inferred from user strokes, simplifying the interaction with the system.

7

Property and Lighting Manipulations for Volume Stylization

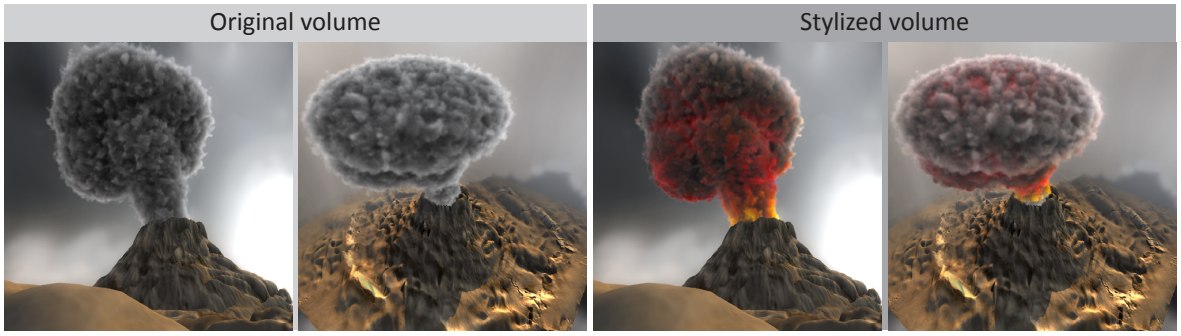


Figure 7.1: Volume stylization of an environmentally lit, static smoke cloud of a volcano. On the left, we show the original volume model, whereas on the right, the volume was stylized to increase the atmospheric tension of the scene. The modifications include a yellow to red to gray color gradient, and a red color cast, that mimics an active glow from the inside. We further added irregular red/yellow stripes that are commonly used in comics to give the impression of ongoing dynamics. Our technique optimizes for the static volume properties emission and albedo from a handful of user-defined images. Rendering the new volume reveals details following the user input such as the glow from the center of the volume.

In this chapter, we target the appearance manipulation of heterogeneous participating media. The presented work has a number of differences to the technique discussed in Chapter 6, beginning with the heterogeneity in the scattering properties. Further, we will not rely on specialized rendering techniques, but optimize for scattering parameters of a volumetric element. To do so, we propose the use of an inverse rendering approach that enables a goal-based editing instead of a direct manipulation of parameters.

Heterogeneous volumetric phenomena [Kajiya and Von Herzen, 1984; Max, 1995; Jarosz et al., 2011b; Novák et al., 2012a] are an important element to make synthetic scenes appear richer and less sterile. As introduced in Section 2.4, participating media are often represented by voxels, which store physically-based statistical properties such as extinction coefficient and single-scattering albedo. The difficulty lies in the selection of the right parameters for each volume element. As mentioned in Chapter 5, volumes are often generated by means of simulation [Stam, 1999] or procedural models [Perlin and Hoffert, 1989] – techniques that provide little control over the appearance of volumes. The goal of our work is to provide an intuitive interaction with participating media to manipulate their parameters.

In the previous chapter, we proposed a set of tools that are not bound to physical rules. It is true that physical accuracy is not necessarily mandatory for a convincing image, but it is usually difficult to achieve plausible results without a proper physical basis. For this reason, artists often start with a realistic simulation before modifying the appearance, e.g., Obert et al. [2008]; Kerr et al. [2010]; Nowrouzezahrai et al. [2011]. Unfortunately, changing the appearance of participating media is difficult. Due to transparency, changes applied in one view usually affect the entire data set in a non-intuitive manner. Hence, matching a certain appearance while maintaining a realistic overall look or style is difficult.

7.1 Contributions

We investigate appearance control and let the user modify a static volume directly to match its appearance for certain viewpoints using familiar image-editing operations. The user can be completely oblivious of physical models and does not need to estimate the influence of environmental illumination. All that needs to be provided is a drawing of the desired scene appearance for some viewpoints and our algorithm optimizes the physically-based properties (albedo, emission, or – under special conditions – extinction) in order to match the appearance as closely as possible. Hence, we provide a goal-based design approach that uses inverse rendering to infer scene parameters. For the adjustment of the volume's properties, our solution is limited to single scattering. Nevertheless, even with this restriction many important cues are captured and the resulting images remain convincing. This choice also ensures that interactive editing sessions become possible with update times in the order of seconds.

We show that our solution can also be used to optimize the environmental lighting while keeping the volume properties constant (Section 7.3.4). Previously, such optimizations have only been applied in the context of surface-based representations [Bousseau et al., 2011]. Here, we show how to extend these optimizations to the context of volume data. We also analyze the applicability of our stylization method to animated scenes.

The contributions of this work are as follows:

- Based on the radiative transport equation (Equation 2.12), we derive conditions under which static volume-appearance stylization via a fast linear optimization is possible (Section 7.3.1).
- We present an environmental lighting optimization that can be coupled with the volume stylization (Section 7.3.4).
- The implementation of our approach is efficient in terms of execution time and memory cost (Section 7.4).

7.2 Scattering Model

As introduced in Section 2.4, participating media are sufficiently accurately described by the properties of local emission $L_{\text{emit}}(\mathbf{x})$, single-scattering albedo $\rho(\mathbf{x})$, and extinction coefficient $\sigma_{\text{t}}(\mathbf{x})$. We assume throughout this chapter that the parameters are aligned in a voxel grid, and, hence, we will derive them per voxel.

We also already introduced volume rendering with the according equations that are based on the radiative transfer equation. Considering multiple scattering in participating media is not only extremely challenging for rendering, but also for the inverse approach that we will use. In line with the previously described methods, we simplify the problem to single scattering.

In addition, we limit ourselves to environment illumination, $L_{\text{i}}(\mathbf{x},\vec{\omega}_{\text{i}}) = L_{\text{env}}(\vec{\omega}_{\text{i}})\, V(\mathbf{x},\vec{\omega}_{\text{i}})\, T_{\text{r}}(\mathbf{x},\vec{\omega}_{\text{i}})$, and isotropic scattering, $p(\mathbf{x},\vec{\omega},\vec{\omega}') = \frac{1}{4\pi}$ as introduced in Section 2.4. These additional constraints will simplify the

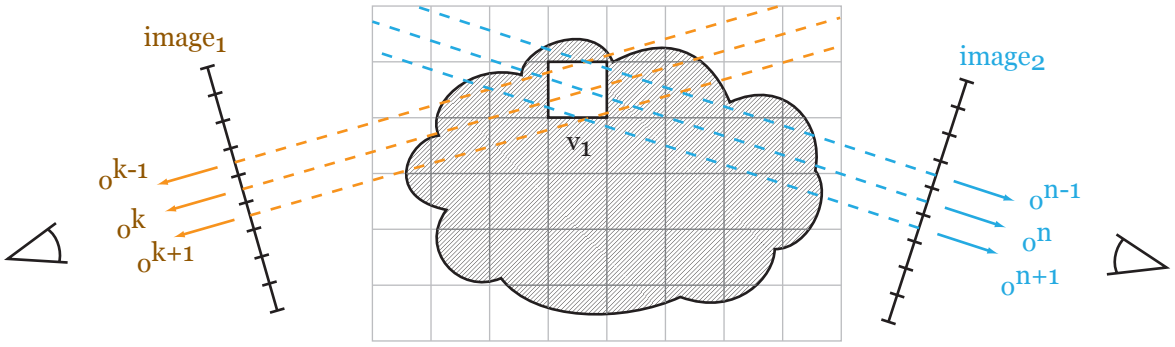


Figure 7.2: Volume from two viewpoints. Problem: changing voxel v_1 influences several pixels in image$_1$ (o^{k-1}, o^k, o^{k+1}) and in image$_2$ (o^{n-1}, o^n, o^{n+1})

derivations, but are not strict and the technique should be easily extendable to handle anisotropic scattering as well as other types of light sources.

7.3 Volume & Light Reconstruction

To control the appearance of heterogeneous participating media, we let the user define a single or multiple images that encode the desired appearance of the medium. These images contain the expected radiance values that a camera should observe. The goal is to invert the rendering process to optimize for the volume properties such that user constraints are matched best. We treat images as a collection of N *constraint pixels*. Given the corresponding camera view, a pixel with index $k \in [1;N]$ corresponds to a ray (origin $\mathbf{x}^k$ and direction $\vec{\omega}^k$) and we denote its radiance value $L_i^k(\mathbf{x}^k, \vec{\omega}^k)$ as specified by the user. The ray may hit a surface at $\mathbf{x}_s^k$, which otherwise is assumed to be the background at ∞. We want to modify properties of a volume V, such that rendering with V, yielding $L_i(\mathbf{x}^k, \vec{\omega}^k)$, matches the *pixel constraint*, $L_i(\mathbf{x}^k, \vec{\omega}^k) = L_i^k(\mathbf{x}^k, \vec{\omega}^k)$. For example, one could modify the emission field of V to match a given appearance.

7.3.1 Volume Reconstruction

The difficulty in finding volume parameters that fulfill the user's input is that a single pixel constraint can influence many voxels and, inversely, two pixel constraints might imply changes on the same voxel. We illustrate this situation in Figure 7.2. Consequently, a perfect solution might not always be possible.

Instead, we seek to find a coefficient vector $\mathbf{a} := (\ldots, a^i, \ldots)^\top$ such that a linear combination $\sum_i a^i v^i(\mathbf{x})$ of some known basis functions v^i defines a property of the volume such that the constraint pixels are matched best. From a mathematical point of view, the basis functions could have global support. However, in practice, using a basis with spatially local support speeds up the reconstruction. One convenient choice for a basis are box functions associated to the volume's voxels, which corresponds to nearest-neighbor sampling of a 3D texture. Using triangle functions allows us to consider linearly interpolated solutions as well.

Next, we will show that the best match for light emission, single-scattering albedo, and extinction coefficient can be obtained by solving a linear system

$$\mathbf{o} = \mathbf{W}\mathbf{a}, \tag{7.1}$$

where $\mathbf{a}$ are the basis coefficients to be computed. The observations $\mathbf{o} := (\ldots, o^k, \ldots)^\top$ involve the constraint

pixel values L_{i}^k, and the entries of the matrix $\mathbf{W} := (\ldots, \mathbf{w}^{ik}, \ldots)$ are derived from single-scattering light transport in participating media with respect to the property that we seek to reconstruct.

7.3.2 Property Reconstruction

The volume-parameter reconstruction implies that Equation 2.14 needs to be linearized. In the following, we derive the entries of matrix $\mathbf{W}$ that enable the estimation of specific volume properties. The derivation is carried out for a single constraint pixel, i. e., for one row of matrix $\mathbf{W}$.

We isolate volume properties that can be linearly optimized. All other volume properties as well as the other scene parameters such as the lighting are assumed to be fixed and summarized into the coefficients of matrix $\mathbf{W}$ and the observation vector $\mathbf{o}$, respectively.

Emission

The derivation starts with Equation 2.14 in conjunction with Equations 2.15 and 2.16. Applying the constraints listed in Section 7.2, we obtain for a constraint pixel k:

$$L_{\mathrm{i}}^k(\mathbf{x}^k, \vec{\omega}^k) = T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_{\mathrm{s}}^k)\, L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \vec{\omega}^k) + \int_c T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, (L_{\mathrm{emit}}(\mathbf{x}_t) + \rho(\mathbf{x})\, L_{\mathrm{scat}}(\mathbf{x}_t, \vec{\omega}^k))\, \mathrm{d}t. \tag{7.2}$$

The integral is carried out along a straight ray c connecting the camera position $\mathbf{x}^k$ and the first visible surface or point on the background $\mathbf{x}_s^k$. We will later describe how refracted rays can be incorporated (Section 7.4.2).

As we assume single scattering, we can split the integral into emitted and scattered light. We unify the fixed surface and scattered radiance as well as the target radiance in o^k and isolate the contribution of emitted radiance that we intend to reconstruct:

$$\begin{aligned} o_{\mathrm{emission}}^k &:= \overbrace{L_{\mathrm{i}}^k(\mathbf{x}^k, \vec{\omega}^k)}^{\text{target radiance}} - \overbrace{T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_{\mathrm{s}}^k)\, L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}^k, \vec{\omega}^k)}^{\text{fixed surface radiance}} - \overbrace{\int_c T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, \rho(\mathbf{x})\, L_{\mathrm{scat}}(\mathbf{x}_t, \vec{\omega}^k)\, \mathrm{d}t}^{\text{fixed scattering radiance}} \\ &= \underbrace{\int_c T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, L_{\mathrm{emit}}(\mathbf{x}_t)\, \mathrm{d}t}_{\text{to-optimize-emission contribution}}. \end{aligned} \tag{7.3}$$

We represent $L_{\mathrm{emit}}(\mathbf{x}_t)$ by a linear combination of basis functions, whose integral can be computed:

$$\begin{aligned} o_{\mathrm{emission}}^k &= \int_c T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t) \left(\sum_i L_{\mathrm{emit}}^i\, \upsilon^i(\mathbf{x}_t) \right) \mathrm{d}t \\ &= \sum_i L_{\mathrm{emit}}^i \left(\int_c T_{\mathrm{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\mathrm{t}}(\mathbf{x}_t)\, \upsilon^i(\mathbf{x}_t)\, \mathrm{d}t \right) \\ &=: \left\langle \vec{L}_{\mathrm{emit}} \cdot \mathbf{w}^k \right\rangle. \end{aligned} \tag{7.4}$$

The coefficients of the above equation are defined by an integral corresponding to *single* ray passing through the volume. Combining all equations defined by the constraint pixels, we obtain a linear system. For the optimization of emission, the coefficient vector is $\mathbf{a}_{\mathrm{emission}} = \vec{L}_{\mathrm{emit}} = (L_{\mathrm{emit}}^1 \ \ldots \ L_{\mathrm{emit}}^M)^\top$ with M as the number of unknowns and basis functions.

Albedo

For albedo optimization, we can establish a similar linearization as for emission. Similar to Equation 7.3, we obtain:

$$\begin{aligned} o^k_{\text{albedo}} &:= \overbrace{L^k_{\text{i}}(\mathbf{x}^k, \vec{\omega}^k)}^{\text{target radiance}} - \overbrace{T_{\text{r}}(\mathbf{x}^k, \mathbf{x}^k_{\text{s}})\, L_{\text{o}}(\mathbf{x}^k_{\text{s}}, \vec{\omega}^k)}^{\text{fixed surface radiance}} - \overbrace{\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, L_{\text{emit}}(\mathbf{x}_t)\, \mathrm{d}t}^{\text{fixed emitted radiance}} \\ &= \underbrace{\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, L_{\text{scat}}(\mathbf{x}_t, \vec{\omega}^k)\, \rho(\mathbf{x}_t)\, \mathrm{d}t}_{\text{to-optimize-scattering contribution}}. \end{aligned} \tag{7.5}$$

With single scattering, the radiance $L_{\text{scat}}(\mathbf{x}_t, \vec{\omega}^k)$ is independent of the single-scattering albedo $\rho(\mathbf{x}_t)$, and we can solve for the latter. More precisely, representing the field of ρ by a linear combination of basis functions, we obtain:

$$\begin{aligned} o^k_{\text{albedo}} &= \sum_i \rho^i \left(\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, L_{\text{scat}}(\mathbf{x}_t, \vec{\omega}^k)\, \upsilon^i(\mathbf{x}_t)\, \mathrm{d}t \right) \\ &=: \left\langle \vec{\rho} \cdot \mathbf{w}^k \right\rangle. \end{aligned} \tag{7.6}$$

Again, the coefficients are defined via an integral that translates to volume rendering involving the known properties of the volume and the coefficient vector is $\mathbf{a}_{\text{albedo}} = \vec{\rho}$ in this case.

Emission & Albedo

The combination of emission and albedo can be jointly optimized because L_{emit} and ρ are linearly independent, i.e., entering as two different summands in Equation 7.2. We employ the same derivations as before and obtain:

$$\begin{aligned} o^k_{\text{joint}} &:= \overbrace{L^k_{\text{i}}(\mathbf{x}^k, \vec{\omega}^k)}^{\text{target radiance}} - \overbrace{T_{\text{r}}(\mathbf{x}^k, \mathbf{x}^k_{\text{s}})\, L_{\text{o}}(\mathbf{x}^k_{\text{s}}, \vec{\omega}^k)}^{\text{fixed surface radiance}} \\ &= \underbrace{\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, L_{\text{emit}}(\mathbf{x}_t)\, \mathrm{d}t}_{\text{to-optimize-emission contribution}} + \underbrace{\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, \rho(\mathbf{x})\, L_{\text{scat}}(\mathbf{x}_t, \vec{\omega}^k)\, \mathrm{d}t}_{\text{to-optimize-scattering contribution}}. \end{aligned} \tag{7.7}$$

Just as for the individual reconstructions, we represent the emission and albedo by a linear combinations of basis functions:

$$\begin{aligned} o^k_{\text{joint}} &= \sum_i L^i_{\text{emit}} \left(\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, \upsilon^i(\mathbf{x}_t)\, \mathrm{d}t \right) + \sum_j \rho^j \left(\int_c T_{\text{r}}(\mathbf{x}^k, \mathbf{x}_t)\, \sigma_{\text{t}}(\mathbf{x}_t)\, L_{\text{scat}}(\mathbf{x}_t, \vec{\omega}^k)\, \upsilon^j(\mathbf{x}_t)\, \mathrm{d}t \right) \\ &=: \left\langle [\vec{L}_{\text{emit}}, \vec{\rho}]^\top \cdot [\mathbf{w}^k_{\text{emission}}, \mathbf{w}^k_{\text{albedo}}]^\top \right\rangle. \end{aligned} \tag{7.8}$$

As such, the coefficient vector $\mathbf{a}_{\text{joint}}$ is a simple stacking of $\vec{L}_{\text{emit}}$ and $\vec{\rho}$ and the matrix is a horizontal partition of the emission and albedo weights, respectively. While the basis functions for emission and albedo can be different from a mathematical point of view, it is practical if they coincide.

Extinction

The estimation of the extinction coefficient is more challenging as it is not only in the scattering integral (Equation 7.2), but also in the transmittance T_{r}, which makes it non-linear. In the following, we will see that we can derive a linear system by performing the optimization in log space. Furthermore, the extinction coefficient can only be reconstructed if we assume that the volume's outgoing radiance (whether caused by emission,

scattering, or both) is constant, i. e., $L_\mathrm{m}(\mathbf{x},\vec{\omega}) = L_\mathrm{m}(\mathbf{x}',\vec{\omega}') = const.\ \forall \mathbf{x}',\vec{\omega}'$. In this case, our problem becomes similar to computed tomography. We will discuss the implication of the constraint in Section 7.3.3. Again, we begin with general volume rendering (Equation 2.14) and expand transmittance T_r as introduced in Section 2.4.1:

$$L_\mathrm{i}^k(\mathbf{x}^k,\vec{\omega}^k) = \overbrace{e^{-\int_c \sigma_\mathrm{t}(\mathbf{x}_t)\,\mathrm{d}t}}^{\text{transmittance}} L_\mathrm{o}(\mathbf{x}_\mathrm{s}^k,\vec{\omega}^k) + \int_c \overbrace{e^{-\int_c \sigma_\mathrm{t}(\mathbf{x}_t)\,\mathrm{d}t}}^{\text{transmittance}} \sigma_\mathrm{t}(\mathbf{x}_{t'})\, L_\mathrm{m}(\mathbf{x}_{t'},\vec{\omega}^k)\,\mathrm{d}t'. \tag{7.9}$$

First, we consider the first summand (attenuated surface radiance) and, as before, employ a linear combination of basis functions to represent the field of the extinction coefficients:

$$e^{-\int_c \Sigma \sigma_\mathrm{t}^i \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t} L_\mathrm{o}(\mathbf{x}_\mathrm{s}^k,\vec{\omega}^k) = \left(\prod_i e^{-\sigma_\mathrm{t}^i \int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t}\right) L_\mathrm{o}(\mathbf{x}_\mathrm{s}^k,\vec{\omega}^k). \tag{7.10}$$

Concerning the second summand (the volume radiance integral in Equation 7.9), we exploit the assumption of a constant outgoing radiance and remove L_m from the integral. The remainder can be integrated, yielding:

$$\begin{aligned} L_\mathrm{m} \int_c e^{-\int_c \sigma_\mathrm{t}(\mathbf{x}_t)\,\mathrm{d}t}\, \sigma_\mathrm{t}(\mathbf{x}_{t'})\,\mathrm{d}t' &= L_\mathrm{m}\,(1 - \overbrace{e^{-\int_c \sigma_\mathrm{t}(\mathbf{x}_t)\,\mathrm{d}t}}^{\text{transmittance}}) \\ &= L_\mathrm{m}\,(1 - e^{-\int_c \Sigma \sigma_\mathrm{t}^i \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t}). \end{aligned} \tag{7.11}$$

Mathematically, the transformation can be shown by decomposing σ_t into a piecewise-constant approximation and splitting the outer integral accordingly [Max, 1995]. Then each integral can be solved analytically and the result recombined. This proof is valid for any Riemann-integrable extinction-coefficient function. It also follows logically; the above remainder is the probability that a ray from the camera interacts with the volume, so it is one minus the probability that the ray passes the volume (which corresponds to transmittance). Again, we represent the field of the extinction by a linear combination of basis functions:

$$\begin{aligned} L_\mathrm{i}^k(\mathbf{x}^k,\vec{\omega}^k) &= L_\mathrm{o}(\mathbf{x}_\mathrm{s}^k,\vec{\omega}^k) \prod_i e^{-\sigma_\mathrm{t}^i \int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t} + L_\mathrm{m}\left(1 - \prod_i e^{-\sigma_\mathrm{t}^i \int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t}\right) \\ &= L_\mathrm{m} + (L_\mathrm{o}(\mathbf{x}_\mathrm{s}^k,\vec{\omega}^k) - L_\mathrm{m}) \prod_i e^{-\sigma_\mathrm{t}^i \int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t}. \end{aligned} \tag{7.12}$$

As before, we unify all fixed terms in o^k. Finally, we apply a logarithm to obtain a linear system in σ_t^i:

$$\begin{aligned} o_\text{extinction}^k &:= \overbrace{\ln\left(\frac{L_\mathrm{i}^k(\mathbf{x}^k,\vec{\omega}^k) - L_\mathrm{m}}{L_\mathrm{o}(\mathbf{x}_\mathrm{s}^k,\vec{\omega}^k) - L_\mathrm{m}}\right)}^{\text{fixed terms}} = \ln\left(\prod_i e^{-\sigma_\mathrm{t}^i \int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t}\right) \\ &= \sum_i \ln(e^{-\sigma_\mathrm{t}^i \int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t}) = \sum_i -\sigma_\mathrm{t}^i \left(\int_c \upsilon^i(\mathbf{x}_t)\,\mathrm{d}t\right). \end{aligned} \tag{7.13}$$

Gathering all extinction coefficients σ_t^i in a vector $\vec{\sigma}_\mathrm{t}$, the coefficient vector of the linear system becomes $\mathbf{a}_\text{extinction} = \vec{\sigma}_\mathrm{t}$.

7.3.3 Discussion of Volume Reconstruction

In all three cases that we discussed previously, we begin with the integral form of the radiative transfer equation that contains the volume property of interest. The target field is represented as a linear combination of basis functions, which allows us to move the coefficients out of the integral, and, hereby, to isolate the unknowns. The same process is applied in computed tomography [Kak and Slaney, 1988], which corresponds to our reconstruction of the extinction coefficient. However, different from medical computed tomography, we have to

deal with *inconsistent* user input, i. e., for which there may be no solution that would satisfy $\mathbf{o} = \mathbf{W}\mathbf{a}$. Therefore, we opt for a solution in the least squares sense $\mathbf{W}^\top \mathbf{o} = \mathbf{W}^\top \mathbf{W}\mathbf{a}$, which means that the *optimal* solution in our sense minimizes the quadratic function $||\mathbf{W}\mathbf{a} - \mathbf{o}||^2$.

Extinction describes the overall shape of a volume, but it is hard to optimize for. Emission and albedo are sufficient to change the appearance of an existing volume. Extinction cannot simultaneously be estimated in combination with emission or albedo, since it involves the solution of a complex non-linear problem, which is linearized by going into log-space. However, it is possible to begin with the reconstruction of extinction assuming L_m is constant. Hence, one can solve for the *shape* of the volume in a first step. Based on this result, one can then add scattered environmental light and refine the appearance of the volume by estimating albedo and emission, hereby lifting the constraint that L_m is constant.

7.3.4 Light Reconstruction

Alternatively to modifying the volume properties, one might also consider changing the environmental illumination. In the following, we explain how to add this additional optimization possibility, which aims at optimizing only the lighting of the scene in order to match the desired volume appearance as closely as possible. The volume properties emission, albedo, and extinction are assumed to be spatially varying, but constant in this situation.

As introduced in Chapter 2, light transport behaves linearly when employing the models typically used in rendering and one can formulate the complete light transport (including bounces) as a linear light transport operator T [Veach and Guibas, 1997] applied to the emitted radiance L_env from the environment. In a discrete setting, we can consider that L_env is a linear combination of a finite number M of basis functions that model the light sources and can, hence, be described as an M-dimensional coefficient vector $\vec{L}_\mathrm{env}$. Correspondingly, the transport operator is a matrix $\mathbf{T}$ of size $N \times M$, where N is the number of pixels in the images used as optimization constraints and, as before, the desired number of observation pixels $\mathbf{o}$ provided by the user. The condition that the volume properties are constant during this optimization implies that T is constant as well. Reconstruction then consists in determining an optimal emitted radiance from the environment $\vec{L}_\mathrm{env}$ that best satisfies the following equation in a least-squares sense:

$$\mathbf{o} = \mathbf{T}\,\vec{L}_\mathrm{env}. \tag{7.14}$$

To determine $\mathbf{T}$, we need to evaluate Equation 2.14, just as before. It is important to note that $\vec{L}_\mathrm{env}$ represents only coefficients for predefined basis functions. In our case, we consider directional light sources that represent a sampled environment map. But also other basis functions, e.g., describing local light sources, could be used in the optimization framework. The previous equation is similar to the ones we derived earlier and, as we will show in the next section, we can apply a similar optimization scheme. Nonetheless, we will also see that a few additional steps need to be added in order to make the solution efficient (Section 7.4.3).

7.4 Implementation

In order to ensure a quick feedback to the user, we map our optimization to the GPU via compute shaders in OpenGL. We first describe the details for the optimization of the volume parameters (Section 7.4.1). We then introduce extensions to the method and additional acceleration details (Section 7.4.2).

The light optimization (Section 7.4.3) is solved in a similar way, but requires some additional precomputation steps in order to speed up the optimization and to make it practical. In both cases, this mapping is not direct, as

special care is needed regarding memory and multi-threading management.

7.4.1 Volume Reconstruction

The matrix $\mathbf{W}$ is large (total count of constraint pixels $\times$ number of basis functions), the linear system is ill-conditioned, and, finally, we seek a physically plausible, i. e., a non-negative solution.

Fortunately, $\mathbf{W}$ is sparse because each row is derived by a ray passing through the volume, intersecting only a low number of basis functions v^i. Still it would be too much to keep all in memory. Instead, we implicitly solve the system by performing a conjugate gradient minimization [Shewchuk, 1994] of the quadratic function $||\mathbf{Wa}-\mathbf{o}||^2$. All necessary steps of the conjugate gradient method are carried out on the GPU and involve 3D textures to represent the vectors. Operations on these vectors are implemented as shaders.

We employ the conjugate gradient method due to its fast convergence, but, for clarity, we illustrate the required operations by describing a standard gradient descent which yields the same minimum. The main ingredients are:

1. the computation of the gradient $\mathbf{W}^\top(\mathbf{Wa}-\mathbf{o})$,
2. an update of the current solution $\mathbf{a}$ by adding a scaled version of the gradient,
3. an iteration of the previous two steps.

Performing the update is straightforward, but the computation of the gradient is not.

To understand $\mathbf{W}^\top(\mathbf{Wa}-\mathbf{o})$, we examine its elements step by step. The matrix $\mathbf{W}$ encodes volume rendering (e.g., for Equation 7.4): $\mathbf{Wa} = \int_c T_\mathrm{r}(\mathbf{x}^k,\mathbf{x}_t)\,\sigma_\mathrm{t}(\mathbf{x}_t)\left(\sum_i a^i\, v^i(\mathbf{x}_t)\right)$. Hence, $\mathbf{Wa}$ is determined by rendering a volume with properties $\mathbf{a}$, yielding multiple images. Next, computing $\mathbf{c} := (\ldots,c^k,\ldots)^\top := \mathbf{Wa}-\mathbf{o}$ is an image operation. Applying $\mathbf{W}^\top$ to $\mathbf{c}$ is the back-projection step that distributes the residual error over all voxels, a data scattering operation: each value c^k is associated to a constraint pixel k. It needs to be scattered to those voxels that are traversed by ray-marching the ray associated with pixel k, weighted according to the voxel's influence on k. In other words, we perform a ray marching that is very similar to the rendering computation with $\mathbf{W}$.

In fact, both operations are so similar, that almost the same code is used. In rendering, which implements the multiplication by $\mathbf{W}$, we use ray marching to sum the voxel contributions along the ray:

$$\text{outRadiance} \mathrel{+}= \text{weight} \cdot \text{texRead}(\mathbf{a}, \text{rayPos}).$$

The variable 'weight' includes the transmittance between the current marching position rayPos and the ray origin due to the volume's extinction coefficient, as well as the value of the basis function at the current position. To implement the multiplication with $\mathbf{W}^\top$, this single line is exchanged by

$$\text{texWriteAdd}(\mathbf{a}, \text{rayPos}, \text{weight} \cdot c^k).$$

Reusing the remainder of the code is also beneficial as the weight values match up perfectly, and, consequently, the implicitly constructed matrices $\mathbf{W}$ and $\mathbf{W}^\top$ agree with each other. This is also reflected in the pseudo code shown in Listing 7.1.

7.4.2 Optimizations and Extensions

Regularization is a standard way of stabilizing and controlling the optimization. Usually, additional quadratic terms, modeling prior knowledge about the solution space, are added to the quadratic error function to address numerical ill-conditioning. A Laplacian term $||\mathrm{L}\mathbf{a}||^2$ can smooth the overall volume. $||\mathbf{a}||^2$ minimizes the solution,

```
void main() {
#ifdef RENDERING
   vec4 outputRadiance = vec4(0);
#elif defined(BACKPROJECT)
   vec4 backProj = texelFetch(texProj, pixelIdx);
#endif

   // setup view ray
   Ray ray = getRay(pixelIdx, cameraData);
   vec2 startEndRay = intersect(ray, volumeBBox);
   float dSurface = texRead(texDepth, pixelIdx).r;
   startEndRay = min(startEndRay, dSurface.xx);
   advanceRay(ray, enterExitVolume.x);

   vec4 sumExt = vec4(0);
   // perform ray marching
   for(int i=0; i < (startEndRay.y-startEndRay.x)/rayStepSize; ++i) {
      // account for extinction
      vec4 extinction = texRead(volExtinction, ray);
      vec4 weight = exp(-sumExt)*(1-exp(extinction));
#ifdef RENDERING
      outputRadiance += weight * texRead(volEmission, ray);
#elif defined(BACKPROJECT)
      texWriteAdd(volEmission, ray, weight*backProj);
#endif
      sumExt += extinction;
      advanceRay(ray, rayStepSize);
   }
#ifdef RENDERING
   texWrite(imgRendering, pixelIdx, outputRadiance);
#endif
}
```

Listing 7.1: GLSL pseudo code: volume rendering and backprojection.

while $\|\mathbf{a}-1\|^2$ biases it towards one. Each of these regularizers is controlled by user-defined weights. Thus, when optimizing for albedo and emission, we can specify which element to favor and, e.g., minimize emission. In practice, we use weak weights (i.e., 2^{-9}), which proved sufficient for a stable solution.

Non-negative solution can be achieved by clamping negative coefficients to zero after each optimization iteration. While this *breaks* theoretically the global convergence as well as assumptions made by the conjugate-gradient optimization, it works well in practice. Optionally, one can 'restart' the conjugate-gradient method by computing the gradient. However, this was not necessary in any of our examples, nor did we experience

Constraint image weights are often desirable to give different parts of an image more importance than others. They are also handy when creating transitions, e.g., when editing only a part of the volume. These per-constraint pixel weights can be easily integrated into the conjugate gradient method and need to be multiplied with the vector $\mathbf{Wa}-\mathbf{o}$. An additional usage of weights is to steer the optimization process and favor certain parts. For example, one can consider putting more importance on the salient or important elements in the input images. The lowered weights give more room to the optimization of other views.

Visual hulls can be used to limit the domain of solution [Ihrke and Magnor, 2004], which increases quality and performance. In case of the emission/albedo optimization, we only consider voxels with non-zero extinction, as the other areas have no influence on the rendering. For estimating extinction itself, the input views of the user can be transformed into masks automatically, or, alternatively, the user can specify arbitrary masks (projections of the desired visual hull) as additional input to the optimization. Those masks do not need to align with any of the input images, the parts of the volume that are outside the visual hull are simply ignored during the optimization.

Refraction occurs for non-constant refractive indices and rays are bent during the traversal. Our reconstruction scheme can handle arbitrary integration curves c of known geometry (see Equation 7.2), which enables us to use more complex paths than a straight line. To compute the refractive ray paths in our volumetric setting, we resort to the Euler forward scheme of Ihrke et al. [2007]. Efficient rendering with varying index of refraction has been shown by Gutierrez et al. [2005]

Differing resolutions can be chosen for the different properties of the volume. In practice, albedo and emission can be of lower resolution without sacrificing too much quality. Hereby, computations are accelerated and memory usage is reduced. Although our software supports this possibility, we did not make use of it in the presented results.

Data scattering performance can be improved by following a few observations. While multiplying with $\mathbf{W}$ is efficient and directly parallelized as usual when stepping through the volume per pixel, the situation is different for multiplication by $\mathbf{W}^\top$. As we scatter data (`shader_image_load_store` allows for writing to texture), memory synchronization issues may occur. Rays of neighboring pixels will likely write to the same voxel. To avoid the resulting stall, we use an interleaved pattern of 6×6 pixels. In each round, only one ray of these sub-windows is shot, decreasing the number of conflicts and speeding up the computation by $\approx 10\%$. Further, the use of 16bit instead of 32bit textures leads to a speed-up of $\approx 25\%$ due to the reduced bandwidth.

Higher precision is obtained when using accurate ray traversal [Amanatides and Woo, 1987] instead of marching. For several basis-function choices, e.g., nearest-neighbor or linear (which we use), an accurate integral can be computed. This applies for the accumulated transmittance value during volume traversal as well as the weight of the basis function itself.

7.4.3 Light Reconstruction

We have seen in the previous section that the optimization of the environmental lighting can also be described as the least-squares solution of a linear system: $\mathbf{o} = \mathbf{T}\,\vec{L}_{\text{env}}$. As for the optimization of volume parameters, we will have to execute two matrix multiplications to converge to the optimal result; multiplying with $\mathbf{T}$ propagates the emitted radiance $\vec{L}_{\text{env}}$ to the constraint pixels, and multiplying with $\mathbf{T}^\top$ is a data scattering process to the light-source coefficients. In other words, the light-source coefficients take the role of the volume property to be optimized for and $\mathbf{T}$ replaces the matrix $\mathbf{W}$. While it is, hence, possible to directly execute the same scheme as for $\mathbf{W}$, such a direct solution would not be efficient.

The implicit construction of $\mathbf{T}$ is much more costly than it is for $\mathbf{W}$, so an on-the-fly evaluation is no option for acceptable performance. The reason is that $\mathbf{T}$ is not longer sparse – it encodes the complete light scattering in the scene. Consequently, we need precomputation strategies and store intermediate representations. We describe two different strategies; the first is to compute and store $\mathbf{T}$ entirely, but memory consumption grows linearly with the number of constraint pixels, the second strategy is slightly costlier during the optimization, but has a memory consumption linear in the size of the volume times the number of light coefficients. As in many cases, a few coefficients (around 16) are enough to capture the light's influence on the participating medium sufficiently, the memory cost is manageable.

The first strategy works as follows. For each given light coefficient, we compute its influence on each constraint pixel, which corresponds directly to one column in $\mathbf{T}$. In other words, if all constraint pixels form a single image, we would simply render this image using an environmental lighting $\vec{L}^i_{\text{env}}$ for which only the ith coefficient is equal to one, all others zero. Each of these render steps will deliver one column of $\mathbf{T}$. In practice, any off-the-shelf renderer can be used, such as Optix [Parker et al., 2010]. During execution of the optimization steps, we can directly recover the needed coefficients from $\mathbf{T}$. This choice leads to a very efficient optimization, but $\mathbf{T}$ has size *light coefficients* $\times$ *constraint pixels*, which implies that memory consumption grows with the artistic input.

Alternatively, we propose to store *light volumes*; we determine the result of the light transport for each $\vec{L}^i_{\text{env}}$ to each voxel in the volume. In other words, we light the voxel volume with $\vec{L}^i_{\text{env}}$ and store the lit volume vol_i. To compute the influence of $\vec{L}^i_{\text{env}}$ on a given constraint pixel, one can shoot the corresponding ray through the volume and gather the weights from vol_i. Intuitively, this process is equivalent to associating each light-source coefficient to a light-source volume (the stored values can, in fact, be treated as an emission term). During the optimization, the multiplication with $\mathbf{T}$ is again a normal rendering, i.e., a ray traversal, during which information can be collected from all vol_i simultaneously and stored in the constraint pixels. The multiplication with $\mathbf{T}^\top$ then has to distribute the residuals to the light coefficients, just like in the case of volume optimization, only that the values are not stored in the volume, but the light coefficient vector.

Directly writing residuals to the light coefficient vector at each marching step for each pixel causes thread-synchronization issues. Even when accumulating the result along the entire ray trajectory and adding the result only at the end to $\vec{L}^i_{\text{env}}$, we would need to use atomics as multiple rays will write to the same coefficient. Hence, many threads need to write to the same memory location, especially when using only a few light coefficients, causing serialization and low performance. Therefore, we store the residuals into an intermediate cache (an image) and perform a summing using a mip-mapping procedure to gather instead of scatter all ray contributions. The highest level of the mip-map pyramid then contains the values for $\vec{L}^i_{\text{env}}$.

The flexibility and independence of the artist's input, makes the light-volumes approach the preferred solution. While the memory consumption is significant (light coefficients times light-volume resolution, for an

2 views 480x480 pixels each		volume resolution 32^3	64^3	128^3
ray marching steps / diagonal	64	47	64	143
	128	64	86	168
	256	98	126	217
	512	164	207	317

steps/diag.		64	128	256
vol. resolution		32^3	64^3	128^3
# of views 480x480 pixels each	2	47	86	217
	4	79	143	334
	6	113	203	454
	8	143	256	574

Figure 7.3: Timings in ms per iteration (six for convergence) for the volume optimization of the hand example. Left: volume resolution vs. ray marching steps (relative to volume diagonal), more steps estimate the integrals more accurately. Right: volume resolution vs. # of views. The resolution of the volume was fixed to 128^3.

10 views, 640x360 pixels each; 256 ray marching steps (relative to volume diagonal)

light precomputation		light volume resolution 32^3	64^3	128^3
light coeffs.	16	0.2	0.2	0.27
	64	0.78	0.81	1.07
	256	3.72	3.75	5.39

optimization iteration		light volume resolution 32^3	64^3	128^3
light coeffs.	16	0.62	0.63	0.63
	64	2.39	2.44	2.46
	256	10.28	10.28	10.52

memory consumption		light volume resolution 32^3	64^3	128^3
light coeffs.	16	2	16	128
	64	8	64	512
	256	32	256	2048

Figure 7.4: Timings in seconds / memory consumption in MB for City data set (10 views, each 640×360 pixels). The volume resolution of the underlying extinction volume was fixed to 128^3 and, correspondingly, the number of ray marching steps set to 256. The tables show number of light coefficients vs. light volume resolution. Left: time for precomputation of light volumes. Middle: time for a single optimization iteration (six for convergence). Right: memory consumption of the light volumes. The overall speed scales directly with the number of light coefficients. The light-volume resolution has a minor impact on the performance as it is further dominated by the number of ray marching steps, which need to correspond to the resolution of the underlying extinction volume. This behavior is equal to the volume optimization.

isotropic phase function), it is usually acceptable. It would further be possible to reduce GPU memory usage if really needed. One strategy is to perform multiple ray traversals; during each traversal, only some vol_i are kept in the actual GPU memory, occupying only a small amount of memory at once. Nonetheless, exchanging the vol_i set would be costly.

7.5 Results and Discussion

While the target images for the optimization can be arbitrary, a typical workflow is to render the volume with its current parameters (e.g., in the beginning with constant emission/albedo parameters) and modify the renderings to reflect the desired output. In all our examples we used Adobe Photoshop® to produce the desired target images. Other image editing tools, such as proposed by Levin et al. [2004] or Ma and Xu [2012], would also be possible.

We ran the optimizations on a machine equipped with an Intel x5650 and a Nvidia 560Ti with fast execution times, even for medium-sized voxel grids. For light optimization, the computation cost scales linearly with the number of light coefficients. A few are usually sufficient and lead to acceptable computation times.

For the volume optimization, the execution times for different numbers of ray-marching steps, voxel

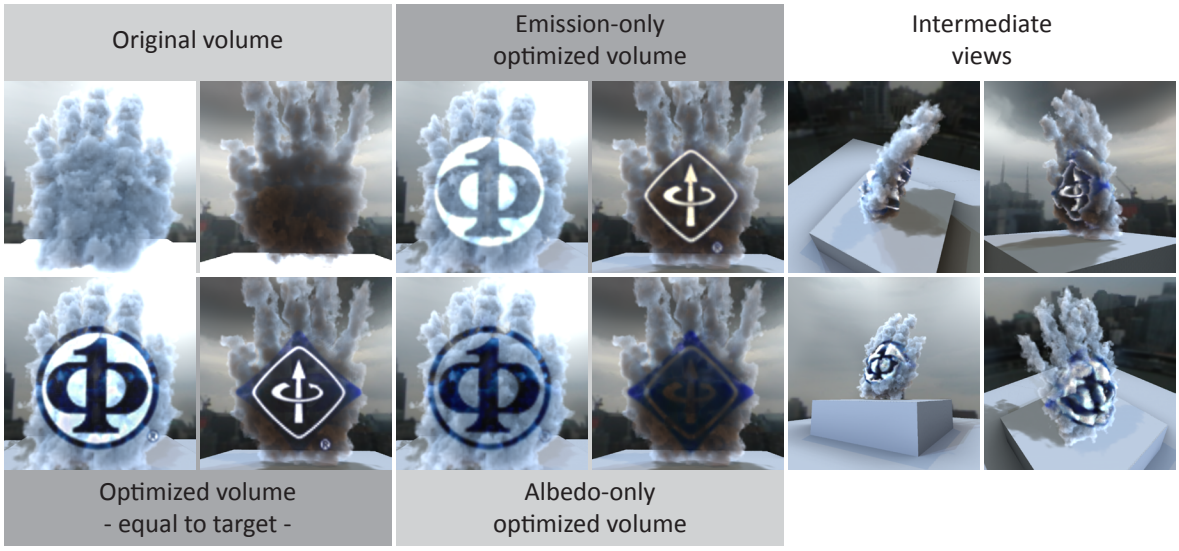


Figure 7.5: Logos on a smoke hand. Left: original appearance above the optimized result (emission and albedo). The images of the reconstruction are indistinguishable from the target input views. Middle: the images on the top use only emission, hence cannot fully match the desired appearance as light is only added. The other two use only albedo, this time, light can only removed (albedo is limited to the range $[0;1]$). Right: Intermediate views appear plausible.

resolution, and input images using the volume optimization are given in Figure 7.3. In practice, twice the number of ray-marching steps for the length of the diagonal as the number of voxels along an axis is a good choice. The process converges after three to five conjugate-gradient steps. For 128^3 voxels, 256 ray marching steps, 460K constraint pixels (2 views), the result (Figure 7.5) is computed in less than two seconds. With 2.3M constraint pixels (10 views) the computation takes seven seconds (Figure 7.8).

For the light optimization, the execution times are given in Figure 7.4. In general, good results can be achieved with a low number of light coefficients. Optimizing for 16 coefficients on a 128^3 volume with 2.3M constraint pixels (10 views), takes only around three seconds.

In the following, we illustrate our method on several examples and describe the intent of each of the stylizations. It took a user roughly 2 minutes to produce the smoke hand, 20 min for the volcano, 15 min for the city cloud, and 4 min for the refraction.

Smoke Hand shows a particular case, where the goal is to add imprinted logos in the volume. These were added to the user-provided views. When optimizing albedo and emission, the result is a indistinguishable close match to the input (Figure 7.5, left) and intermediate views look appealing and plausible (right). One can also modify emission or albedo only (center), in which case the result deviates from the input and can no longer match it perfectly. Emission can only lighten, albedo only darken the appearance (following physical constraints). Depending on the realism and desired material, these could be adequate choices as well.

Volcano Stylization demonstrates the effectiveness of our system, using the volume optimization. The scene shows a static smoke cloud from a volcano (Figure 7.6). The stylization intends to make the volcano appear active with ongoing dynamics.

Six input images were used (480×480 pixel, Figure 7.6, right-most column) to change the appearance of the smoke, resulting in appealing renderings from any viewing angle. A lower number of target images is insufficient for a good appearance all around the object. In general, only parts that are specified in at least a

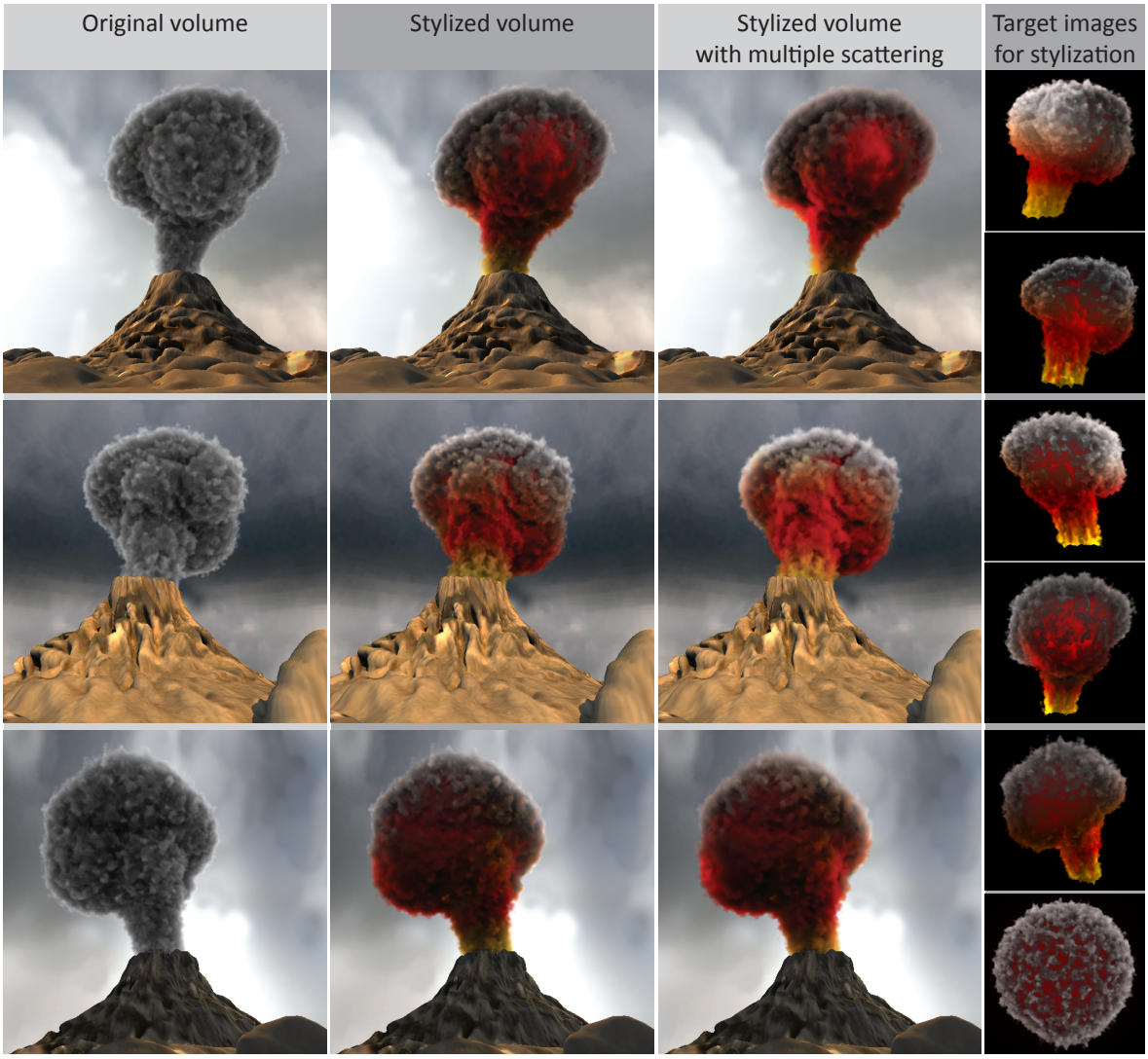


Figure 7.6: Stylization of the static smoke cloud of a volcano; original volume (left), intermediate views of the result after optimization (middle), rendering with multiple scattering (unlimited bounces) of the optimized volume (right). While we optimize only for single scattering, the results remain suitable for a full light transport simulation, leading only to a minor loss in contrast. To account for the additional energy, we globally scaled the emission and incoming light such that the resulting images have similar brightness. The right-most column shows all user-drawn images.

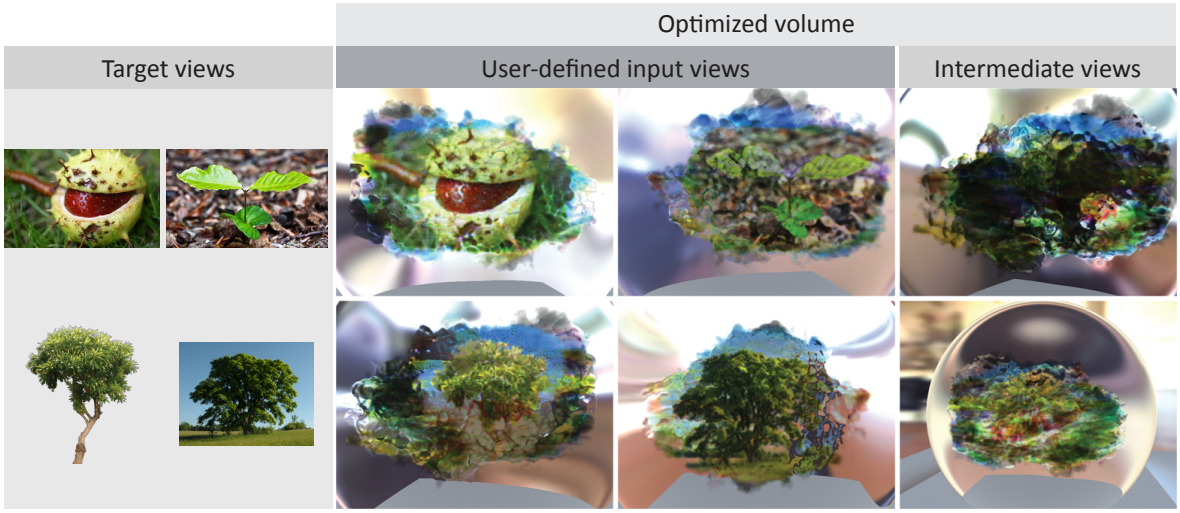


Figure 7.7: Reconstruction of extinction in a refractive volume to absorb light of the background. Left: target images. We additionally alpha-masked them for the optimization. Middle columns: results for target views. Right column: intermediate views.

single view change, all other parts (e.g., smoke inside the volcano, occluded by the volcano itself) remain mostly unaltered. Consequently, edges can appear between constrained and unconstrained regions. Also, as constraint pixels only affect voxels along their corresponding ray, insufficient views may induce stripe patterns into the volume. However, applying a weak Laplacian regularizer introduces diffusion between neighboring voxels, reducing both stripe and edge artifacts greatly without too much smoothing of the intricate detail.

The effect of rendering the stylized volume under multiple scattering is shown in the right column of Figure 7.6. The appearance remains pleasing, although multiple scattering was not included in our optimization process. It acts like a blur that reduces details, but the resulting rendering remains consistent.

Extinction and Refraction is shown in Figure 7.7. Here, we illustrate the use of our extinction optimization to show the construction of complex shapes and also to illustrate the compatibility of our solution with bent rays due to continuous refraction. The user provided four input views (growth of a tree) and our reconstruction computed extinction coefficients (per RGB channel) that attenuate the background light such as to match the provided images. In the example, the volume does not scatter or emit any light, i. e., $L_o = 0$. In this reconstruction process, the shape of the volume is implicitly defined by the volume of varying refractive index. The refraction indices were generated using a random process restricted to the inside of a glass sphere. We used per-pixel weights to concentrate the importance on the main parts of the user images, which leads to a less constrained boundary and a more variable shape. Due to the refraction, the intermediate views appear randomly colored. We imagine, such a combination of refraction and extinction could e.g. be used in a game, where the user has to find the correct viewpoint to see a certain image, in order to obtain hints to solve a puzzle.

Cloud Stylization illustrates the expressiveness of our system. The scene shows a cloud hovering over a city and the goal is to achieve a “frightening” look when on one side of the cloud and a calm appearance on the other side.

The user modified ten views (640×360 pixel, a selection is shown in Figure 7.8, left) by contrast enhancement and blending with a red mask. The resulting input images were *inconsistent* in parts, as each was independently designed. Yet, our least-squares solutions ensure a valid reconstruction that closely matches the desired scene appearance and extrapolates well.

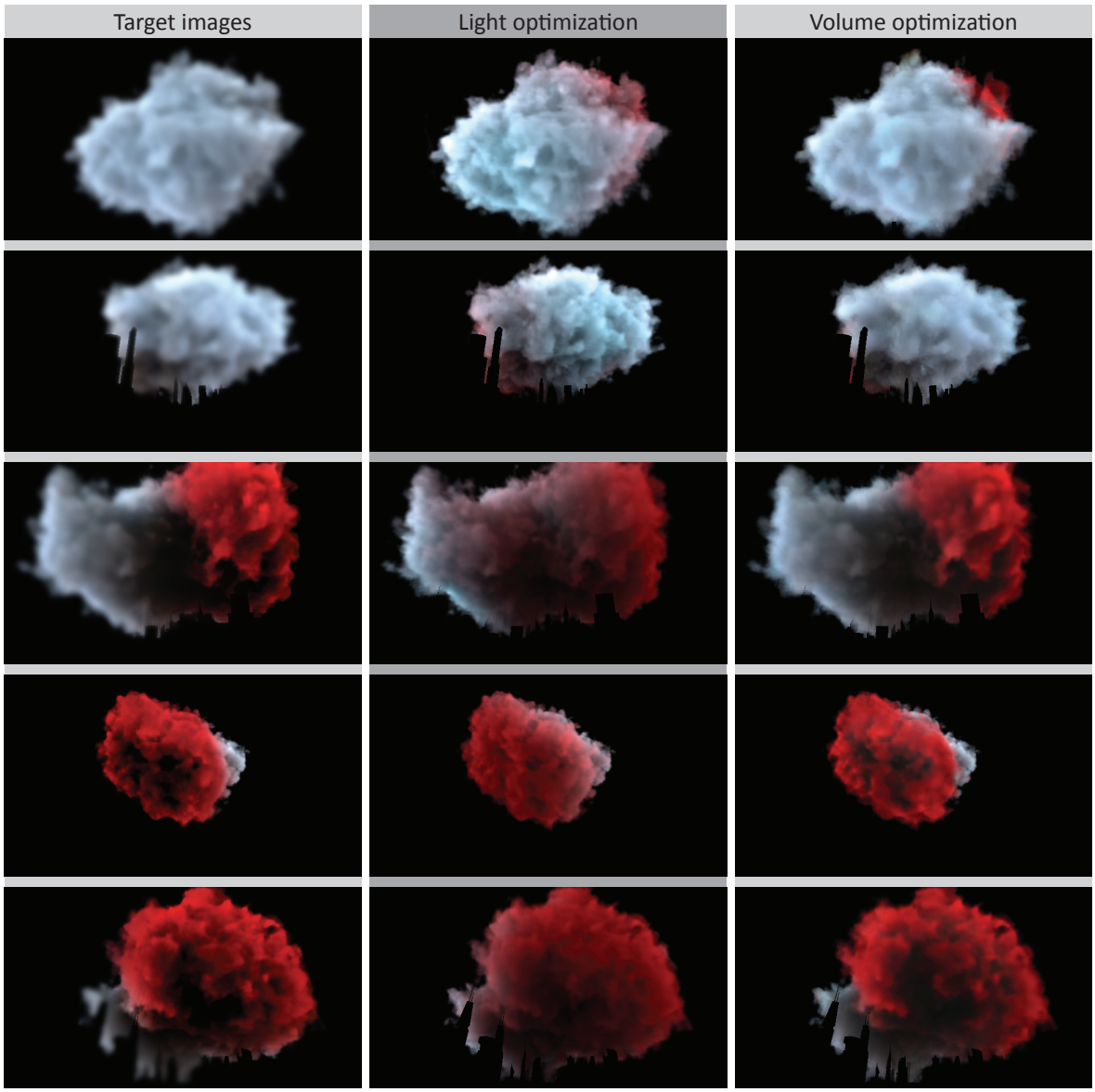


Figure 7.8: Comparison of the reconstruction results (subtracted background): user-drawn input images (left), results after light optimization (middle), results after volume optimization (right).

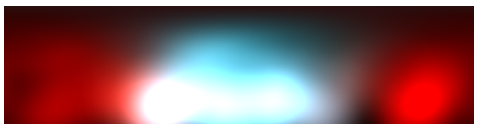


Figure 7.9: Optimized environment map in latitude-longitude format for the stylization shown in Figure 7.10. The lower half is omitted as the center directions of the light sources were chosen to be on the upper hemisphere only.



Figure 7.10: Cloud stylization via environmental lighting. The volume's original appearance (left) is changed due to a new environment map (right). All shown views are different from the target images.

We applied two different optimization strategies. First, we modified the environmental illumination (Figure 7.10). As light basis functions, we used 128 uniformly-distributed disc lights restricted to the upper hemisphere that cover a constant solid angle with a soft falloff. The resulting environment map (all light sources splat into the image) is illustrated in Figure 7.9.

It is important to note that an isotropic volume acts as a low-pass filter; from the light source to the camera, as well as from the camera to the light sources. Hence, the matching to the target images cannot be perfect. High-frequency information, such as the logos in Figure 7.5, cannot be reproduced in this case. Consequently, the resulting environmental lighting is also of a low frequency, justifying the low number of light coefficients. In practice, as few as 16 coefficients are often sufficient (see Figure 7.9).

The volume optimization on the city data set is able to reconstruct higher frequencies because the volume is modified locally (Figure 7.8). In fact, when allowing for emission, light optimization is theoretically redundant. It can be useful to include, e.g. when certain volume properties are not supposed to change or if emission is to be avoided. It is also a good means for rapid preview, as even a single view can be used without introducing any potential high-frequency artifacts stemming from a very low number of input images during the property reconstruction. Further, optimizing for light sources has a more global character as each light source affects the entire volume, which may be desired. It also naturally connects the appearance of the volume to the rest of the scene, when lit with the same environmental illumination (Figure 7.11).



Figure 7.11: Pair-wise comparison: additionally applying volume optimization (right in pair) after light optimization (left in pair) can add details to the reconstruction and hereby match the desired appearance more closely. Contrast can be locally decreased (top), but also increased (middle, bottom) according to the target images. For both optimizations, the same target images were used.

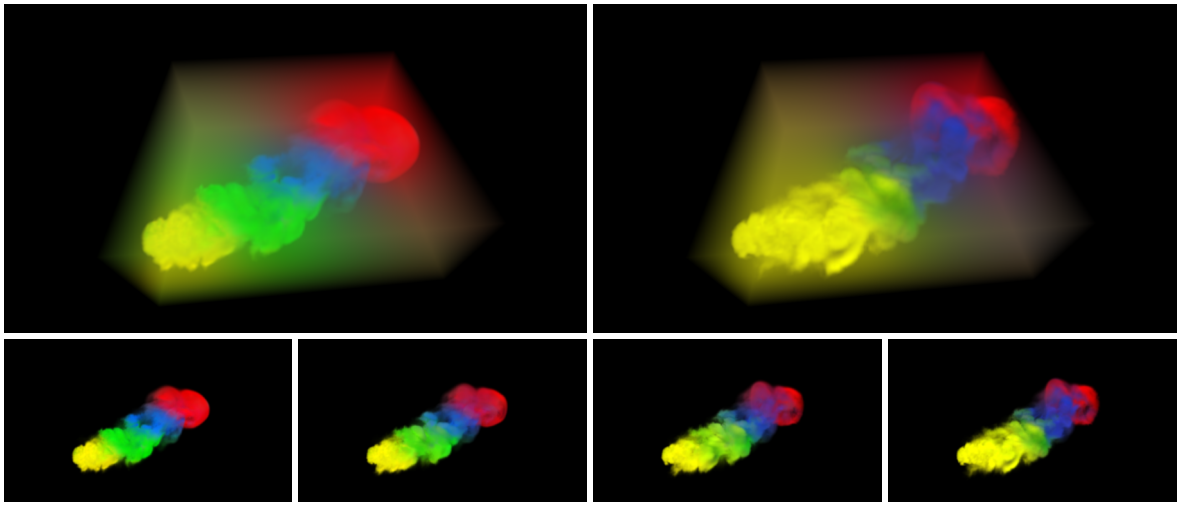


Figure 7.12: Diffusion for animated smoke. Applying diffusion helps with an animated extinction coefficient. While our animation support is limited to a simple blend, diffusion avoids undefined results as the entire space is covered. Top row: visualization of the reconstructed emission/albedo coefficient with a globally scaled extinction coefficient. Bottom row: four frames of a small smoke animation for which the first and last frame were stylized and volume parameters reconstructed.

7.5.1 Animation

While our optimization targets static volumes, a solution for animated and dynamic volumes would be of interest. Here, we analyze a simple extension, but we leave a full solution of this problem to future work.

If the volumetric data set is static, one way of incorporating a changing environment (e.g., light conditions and time-based constraints) would be to find the best solution (in a least-squares sense) over time, which can be achieved by our approach by extending the constraints system.

The more interesting case is dynamic volumes. Here, a straightforward solution is to use keyframing and optimization for multiple volumes at different time steps that are linearly blended together to produce in-between frames. Yet, this simple solution can lead to problems; if the extinction coefficient is zero at a location in two neighboring key frames, only random values are derived at this location. If the volume's animation now passes through this region, the result could be arbitrary. Using the Laplace regularizer as a diffusion process can solve this issue as it extends the emission/albedo definitions into regions with a extinction coefficient of value zero (see Figure 7.12). We refer the reader to the supplementary video of the original paper for a better impression.

A more advanced optimization of dynamic volumes requires an understanding of the volume's underlying dynamics and is beyond the scope of this work. For example, one could couple the optimized emission/albedo volume and the animated extinction volume by a flow field. The latter could directly come from a fluid simulation or be a reconstructed volumetric flow. Hereby, one could link voxel values over time, according to the flow. Currently, we can only add such constraints to the same voxel over time, making the definition of key frames only slightly more flexible. The additional flow information would allow more expressive and appealing results, but it would also lead to a computational and memory overhead.

7.6 Conclusion

In this chapter, we presented a novel approach to stylize heterogeneous participating media using a goal-based design approach. We showed that a desired scene appearance can be defined via a number of target views that our approach then optimizes for. We considered two scenarios to solve for a specific volume appearance: the modification of volume properties and the modification of environmental illumination. We illustrated the usefulness, expressiveness, and variety of our system with a collection of different examples.

The goal-based design approach is implemented as inverse rendering, which was used for scene manipulations before, often leading to a non-linear general-purpose optimization. We identified constraints that simplify the optimization to a linear problem, allowing us to optimize millions of parameters. Nonetheless, the process is still computationally involved and memory demanding. Instead of using standard solvers, we showed that basic optimization operations correspond to rendering tasks – algebraic steps are implemented as rendering shaders executed on the GPU. This solutions not only allows for the use of efficient rendering and light-transport-caching techniques, but also avoids the explicit storage of the light-transport matrix. However, such a high-performance inverse rendering comes at the expense of a customized optimization implementation.

7.6.1 Future Work

Just as for our rendering techniques, it would be interesting to investigate an extension to multiple scattering for appearance manipulations. Unfortunately, the resulting optimization is non-linear, a circumstance that we explicitly avoided in our work. However, the recent work of Hašan and Ramamoorthi [2013] gives hope that a manipulation with multiple scattering is feasible by using precomputations and some approximations. Another strategy could be the use of a ping-pong scheme that optimizes only part of the parameters while keeping the others fixed and then alternating the sets. The sub-problem can be simplified to a linear problem, providing the desired speed-up, although a global convergence is not guaranteed. This technique has been successfully employed for geometry manipulation, but also the reconstruction of multi-layered light attenuators [Baran et al., 2012].

Finally, we think that the specifics of our goal-based design metaphor can be further improved. Currently, the user has to accurately specify the desired pixel values, i.e., radiance values. Investigating other inputs that are at higher level would be interesting, e.g., relative color values instead of absolute ones or local sharpness. Further future work could investigate alternatives to the per-pixel input, e.g, specifying an appearance in regions. However, it is not obvious how this should map to actual constraints for which the volume is optimized. Lastly, an automatic derivation of constraints for intermediate views not specified by the user could be addressed. Effectively, it would correspond to a regularization of the optimization. Therefore, finding high-level regularization parameters (e.g., controlling the spatial frequency of the volume parameters) is an interesting topic.

8

Conclusion

In this dissertation, we addressed the topic of scene appearance to create believable and appealing worlds. We explored methods that target the problems of both rendering and scene manipulation, which can restrict artistic design.

We set the criterion of *effectivity* for the stylization and appearance manipulation of volumetric scattering. We think that we reached this goal due to two reasons. First, our methods address two distinct, highly interesting volumetric phenomena: the stylization of light shafts, an effect heavily used in games and movies; as well as the manipulation of appearance of heterogeneous media, which was only poorly supported by artistic tools. Second, we proposed dedicated design solutions for both phenomena. Introducing specialized rendering techniques for the editing of light shafts enabled us to select suitable parameters that provide artistic control. In contrast to this direct approach, we showed that a goal-based (or image-based) design approach can be effective for highly complex scene elements such as heterogeneous participating media. Based on inverse rendering, the approach provides a high-level user interaction with the volume parameters. By optimizing for a target appearance that is specified by images, the approach hides any complex parameter interplay. The user is left with a simple interface, providing a goal-driven editing experience.

Efficiency was the another main goal for all our algorithms. The requirement is evident for interactive rendering, but also stylization techniques need to provide feedback as quickly as possible to the user. We saw that plausible and appealing results can be achieved based on the assumption of single scattering – a compromise, as multiple scattering is very difficult to evaluate efficiently. Important scattering phenomena such as light shafts are covered, and heterogeneous media such as clouds look plausible. Even feature films currently support at most two volumetric bounces; more bounces are still prohibitive due to the immense scale of their scenes, despite the available rendering power. But we addressed efficiency in the implementations, too. We designed our algorithms to allow for a highly parallel execution on the GPU, and discussed implications of synchronizations that cannot be avoided, such as in the single-scattering filtering and the inverse rendering approach. For all our techniques, we relied on modern OpenGL techniques and provided code samples where appropriate. Although the GPU provided us with fast executions, our findings are of algorithmic nature and not bound to hardware.

In the following we give some concluding remarks on the problems considered in this dissertation.

Interactive Rendering The proposed rendering algorithms target real-time applications, for which performance is as important as rendering accuracy. Determining visibility between points, or from a point to the next surface, is one of the most basic and most important operations in rendering. Our methods target higher-order effects, but the integration of visibility is the main challenge.

Our methods confirm that outfactoring visibility from shading or scattering allows for an efficient aggregation

of visibility, while the remaining shading or scattering can be solved analytically. The individual components need to be recombined by a product – an approximation, as shading and visibility are not separable. With the example of bent normals and cones, we have seen that easy-to-compute statistics can help to improve upon such a simple product. Alternatively, we showed an approximation for our filter-based scattering that avoids an outfactoring of visibility and computes a weighted filtering.

Statistics are similar to basis functions, which are usually employed for compression purposes. We utilized them for the linearization of the visibility function. The linearization allowed us to evaluate visibility for entire rays instead of single points – effectively reducing the complexity of single-scattering computations. While the linearization has been used for surface shadows before, we have seen that a generalization to rays requires a suitable space. With the help of rectified shadow maps, our scattering computations are implemented as filter operations in image space.

Image-based solutions have proved to be highly efficient on today's GPUs. They avoid expensive queries against scene geometry – an important property, since the amount of scene content is constantly increasing. Both presented rendering techniques solely operate in image space, utilizing textures that contain geometric scene information but need to be computed for lighting anyway such as a G-buffer or a shadow map. Moreover, the efficiency and simplicity of image-based techniques motivated us to implement our occluder manipulations for the stylization of light shafts with image operations.

Appearance Manipulation We showed that stylization of volumetric scattering can be achieved using specialized rendering techniques. They manipulate the light transport to provide control over indirect effects such as light shafts that only occur due to special configurations in the scene layout. The resulting manipulations exhibit physically incorrect behavior, but we showed that they allow for highly expressive stylization, an interesting property for visually expressing a desired mood. The use of interactive rendering techniques for stylization has the advantage that instant feedback is provided, easing the interaction of users with the system. However, it also requires integrating the dedicated rendering procedures into the rendering pipeline, and performance constraints need to be considered.

The manipulation of parameters of scene elements, such as a heterogeneous medium, can seem more desirable. Once volume parameters are obtained, no special treatment is required, and the medium can be rendered by any physically-based renderer. The data can also be stored, processed further, and reused in any scene. However, these advantages come at the expense of the fact that the actual manipulation of heterogeneous media is much more challenging and needs to be performed in a preprocess.

Finally, our stylization techniques have in common that appearance manipulations are indicated by the user at a *global* scope, i.e., manipulations are not limited to a single light ray or voxel as in Nowrouzezahrai et al. [2011]. Hereby, our methods consider the aggregated effect of scattering that is observed by the virtual camera. An example is the reconstruction of medium parameters which are derived such that the aggregated radiance along view rays is optimal. This global manipulation does not mean that changes cannot be detailed; we have shown that the method can handle sharp features in the desired appearance. The global scope also applies to the manipulation of the light shafts, where our transfer functions define pixel colors based on aggregated scattering properties such as average visibility. Occluder manipulations are global as well; e.g., the proposed hole filling and silhouette enhancement are applied to the entire scene. Nonetheless, some locality can be desirable; for instance, we can limit the hole creation to specific objects.

8.1 Future Work

We already listed future work specific to our proposed techniques in the respective chapters and will only pose some more general challenges in interactive rendering and appearance manipulation.

Interactive Rendering We proposed two rendering techniques specific to surface rendering and scattering in participating media, respectively. Future work will have to address light transport between surfaces and media. Surfaces and participating media exchange energy once global illumination is considered. However, most interactive global-illumination algorithms are limited to surface rendering; and multiple-scattering techniques often ignore the scene's geometry. Light propagation volumes [Kaplanyan and Dachsbacher, 2010; Billeter et al., 2012] are an exception since they model any light transport by a volumetric representation; however, accuracy is low. While a conceptual separation of surface rendering and scattering in media is desirable, only a unified approach can model full global illumination.

The above surface-media interaction issue showcases a phenomenon in interactive rendering: the techniques are highly specific. We believe that visibility is a major reason for these specialized algorithms. Our techniques do not target the computation of visibility, but they demonstrate two different solutions for determining visibility. Standard geometry rasterization offers poor flexibility in determining visibility; queries for visibility of two arbitrary points are not possible. In consequence, many techniques employ other representations and methods to approximate visibility. Examples include point-based geometry, voxels, or image-based visibility as used for our screen-space bent normals and cones. In consequence, we have a zoo of techniques with strongly varying performance and quality, and partial incompatibility. A unified approach for determining visibility would lead to comprehensive interactive rendering techniques.

A potential game changer and solution to the above could be the switch from rasterization to ray casting. While we are waiting for the transition for quite some time now (with ray tracing being claimed to be interactive since the mid 2000s), it seems certain that it will happen eventually. While Monte-Carlo rendering offers more generality and is practically as well as mathematically more elegant, the performance advantage of rasterization still dominates in interactive rendering. Major challenges in a switch to ray casting include coherence in execution and memory access, a reuse of intermediate results (such as visibility or transmittance), and predictability of the rendering quality, as these aspects contradict with Monte-Carlo strategies that rely on statistically independent samples.

Stylization and Appearance Manipulation We consider our work on appearance manipulation as a starting point to the artistic stylization of scenes containing volumetric elements. It is clear that future scenes will increase in complexity; additional high-level manipulation techniques are required – a fact that motivated our work, but cannot be considered solved. While our example scenes target the scale of current computer games, they are far from the level of detail of scenes used in movies.

An important aspect only touched upon by our work, as well as concurrent work, is the stylization of animated scenes. The core of our techniques focuses on the stylization of static scenes with extensions to handle animations; e.g., by allowing for a time-dependent change of parameters. We believe that it would be highly interesting to investigate stylization that uses the time dimension as a principal component.

Both of our stylization and manipulation approaches focus on the stylization of participating media, largely ignoring the remainder of the scene. Future work could investigate a stronger coupling between solid scene objects and volumetric scattering to ensure a consistent appearance of the entire scene. Obviously, this problem

is not bounded and any number of parameters could be optimized (object materials, light sources, volumes, etc.); finding a suitable sub-problem and the corresponding set of parameters to optimize will be one of the main challenges.

Finally, we proposed specific rendering techniques that support physically incorrect manipulation of the light transport, despite the recent trend of physically-based rendering. We argue that there is and will be a need for highly expressive manipulations that cannot be achieved when obeying physical rules. Nonetheless, a challenge for future techniques is to find a good tradeoff between the physical characteristics of the light transport and desired expressiveness.

Bibliography

Agarwal, S., Ramamoorthi, R., Belongie, S., and Jensen, H. W. Structured importance sampling of environment maps. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 22(3):605–612, 2003. DOI: 10.1145/882262.882314.

Aila, T. and Laine, S. Alias-free shadow maps. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 161–166. Eurographics Association, 2004. DOI: 10.2312/EGWR/EGSR04/161-166.

Amanatides, J. and Woo, A. A fast voxel traversal algorithm for ray tracing. *Computer Graphics Forum*, 87(3): 3–10, 1987.

Annen, T., Mertens, T., Bekaert, P., Seidel, H.-P., and Kautz, J. Convolution shadow maps. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 51–60. Eurographics Association, 2007. DOI: 10.2312/EGWR/EGSR07/051-060.

Annen, T., Mertens, T., Seidel, H.-P., Flerackers, E., and Kautz, J. Exponential shadow maps. In *Proc. of Graphics Interface (GI)*, pages 155–161. Canadian Information Processing Society, 2008.

Appel, A. Some techniques for shading machine renderings of solids. In *Proc. of the Spring Joint Computer Conference*, pages 37–45. ACM, 1968. DOI: 10.1145/1468075.1468082.

Atcheson, B., Ihrke, I., Heidrich, W., Tevs, A., Bradley, D., Magnor, M., and Seidel, H.-P. Time-resolved 3D capture of non-stationary gas flows. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 27(5): 132:1–132:9, 2008. DOI: 10.1145/1409060.1409085.

Baboud, L., Eisemann, E., and Seidel, H.-P. Precomputed safety shapes for efficient and accurate height-field rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18:1811–1823, 2011. DOI: 10.1109/TVCG.2011.281.

Baran, I., Chen, J., Ragan-Kelley, J., Durand, F., and Lehtinen, J. A hierarchical volumetric shadow algorithm for single scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 29(6):178:1–178:10, 2010. DOI: 10.1145/1882261.1866200.

Baran, I., Keller, P., Bradley, D., Coros, S., Jarosz, W., Nowrouzezahrai, D., and Gross, M. Manufacturing layered attenuators for multiple prescribed shadow images. *Computer Graphics Forum (Proc. of Eurographics)*, 31 (2pt3):603–610, 2012. DOI: 10.1111/j.1467-8659.2012.03039.x.

Barla, P., Thollot, J., and Markosian, L. X-Toon: An extended toon shader. In *Proc. of International Symposium on Non-photorealistic Animation and Rendering (NPAR)*, pages 127–132. ACM, 2006. DOI: 10.1145/1124728.1124749.

Bavoil, L., Sainz, M., and Dimitrov, R. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH Talks*, 2008. DOI: 10.1145/1401032.1401061.

Ben-Artzi, A., Overbeck, R., and Ramamoorthi, R. Real-time BRDF editing in complex lighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 25(3):945–954, 2006. DOI: 10.1145/1141911.1141979.

Billeter, M., Sintorn, E., and Assarsson, U. Real time volumetric shadows using polygonal light volumes. In *Proc. of High Performance Graphics (HPG)*, pages 39–45. Eurographics Association, 2010.

Billeter, M., Sintorn, E., and Assarsson, U. Real-time multiple scattering using light propagation volumes. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 119–126. ACM, 2012. DOI: 10.1145/2159616.2159636.

Blinn, J. F. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics (Proc. of SIGGRAPH)*, 16(3):21–29, 1982. DOI: 10.1145/965145.801255.

Blinn, J. F. and Newell, M. E. Texture and reflection in computer generated images. *Computer Graphics (Proc. of SIGGRAPH)*, 10(2):266–266, 1976. DOI: 10.1145/965143.563322.

Bousseau, A., Chapoulie, E., Ramamoorthi, R., and Agrawala, M. Optimizing environment maps for material depiction. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 1171–1180, 2011. DOI: 10.1111/j.1467-8659.2011.01975.x.

Buchholz, B. and Boubekeur, T. Quantized point-based global illumination. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, 31(4):1399–1405, 2012. DOI: 10.1111/j.1467-8659.2012.03135.x.

Bunnell, M. Dynamic ambient occlusion and indirect lighting. In *GPU Gems*, volume 2, pages 223–233. Addison Wesley, 2005.

Burke, D., Ghosh, A., and Heidrich, W. Bidirectional importance sampling for direct illumination. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 147–156. Eurographics Association, 2005. DOI: 10.2312/EGWR/EGSR05/147-156.

Calderon, S. and Boubekeur, T. Point morphology. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 33 (4):45:1–45:13, 2014. DOI: 10.1145/2601097.2601130.

Catmull, E. E. A subdivision algorithm for computer display of curved surfaces. PhD thesis, The University of Utah, 1974.

Cerezo, E., Pérez, F., Pueyo, X., Seron, F. J., and Sillion, F. X. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328, 2005. DOI: 10.1007/s00371-005-0287-1.

Chandrasekar, S. *Radiative Transfer*. Dover Pub., 1960.

Chen, J., Baran, I., Durand, F., and Jarosz, W. Real-time volumetric shadows using 1D min-max mipmaps. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 39–46. ACM, 2011. DOI: 10.1145/1944745.1944752.

Christensen, P. H. Point-based approximate color bleeding. Technical report, Pixar Technical Notes, 2008.

Clarberg, P., Jarosz, W., Akenine-Möller, T., and Jensen, H. W. Wavelet importance sampling: Efficiently evaluating products of complex functions. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3): 1166–1175, 2005. DOI: 10.1145/1073204.1073328.

Colbert, M., Pattanaik, S., and Křivánek, J. BRDF-shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE Computer Graphics and Applications*, 26(1):30–36, 2006. DOI: 10.1109/MCG.2006.13.

Cole, F., DeCarlo, D., Finkelstein, A., Kin, K., Morley, K., and Santella, A. Directing gaze in 3D models with stylized focus. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 377–387, 2006. DOI: 10.2312/EGWR/EGSR06/377-387.

Cook, R. L. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, 1986. DOI: 10.1145/7529.8927.

Cook, R. L., Porter, T., and Carpenter, L. Distributed ray tracing. *Computer Graphics (Proc. of SIGGRAPH)*, 18 (3):137–145, 1984. DOI: 10.1145/964965.808590.

Crow, F. C. Shadow algorithms for computer graphics. *Computer Graphics (Proc. of SIGGRAPH)*, 11(2): 242–248, 1977. DOI: 10.1145/965141.563901.

Dachsbacher, C. and Stamminger, M. Reflective shadow maps. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 203–231. ACM, 2005. DOI: 10.1145/1053427.1053460.

Dachsbacher, C. and Stamminger, M. Splatting indirect illumination. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 93–100. ACM, 2006. DOI: 10.1145/1111411.1111428.

Debevec, P. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 189–198. ACM, 1998. DOI: 10.1145/280814.280864.

DeCoro, C., Cole, F., Finkelstein, A., and Rusinkiewicz, S. Stylized shadows. In *Proc. of International Symposium on Non-photorealistic Animation and Rendering (NPAR)*, pages 77–83. ACM, 2007. DOI: 10.1145/1274871.1274884.

Deering, M., Winner, S., Schediwy, B., Duffy, C., and Hunt, N. The triangle processor and normal vector shader: A VLSI system for high performance graphics. *Computer Graphics (Proc. of SIGGRAPH)*, 22(4):21–30, 1988. DOI: 10.1145/378456.378468.

d'Eon, E. A better dipole. Technical report, 2012.

d'Eon, E. and Irving, G. A quantized-diffusion model for rendering translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 30(4):56:1–56:14, 2011. DOI: 10.1145/2010324.1964951.

d'Eon, E., Luebke, D., and Enderton, E. Efficient rendering of human skin. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 147–157. Eurographics Association, 2007. DOI: 10.2312/EGWR/EGSR07/147-157.

Dobashi, Y., Yamamoto, T., and Nishita, T. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Proc. of Graphics Hardware*, pages 99–107. Eurographics Association, 2002. DOI: 10.2312/EGGH/EGGH02/099-108.

Dobashi, Y., Iwasaki, W., Ono, A., Yamamoto, T., Yue, Y., and Nishita, T. An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. *ACM Trans. Graph.*, 31(6): 145:1–145:10, 2012. DOI: 10.1145/2366145.2366164.

Donnelly, W. and Lauritzen, A. Variance shadow maps. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 161–165, 2006. DOI: 10.1145/1111411.1111440.

Donner, C. and Jensen, H. W. Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1032–1039, 2005. DOI: 10.1145/1073204.1073308.

Donner, C. and Jensen, H. W. A spectral BSSRDF for shading human skin. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 409–417. Eurographics Association, 2006. DOI: 10.2312/EGWR/EGSR06/409-417.

Donner, C., Weyrich, T., d'Eon, E., Ramamoorthi, R., and Rusinkiewicz, S. A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 27(5): 140:1–140:12, 2008. DOI: 10.1145/1409060.1409093.

Dou, H., Yan, Y., Kerzner, E., Dai, Z., and Wyman, C. Adaptive depth bias for shadow maps. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 97–102. ACM, 2014. DOI: 10.1145/2556700.2556706.

Eisemann, E. and Décoret, X. Fast scene voxelization and applications. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 71–78. ACM, 2006. DOI: 10.1145/1111411.1111424.

Eisemann, E. and Décoret, X. Single-pass GPU solid voxelization for real-time applications. In *Proc. of Graphics Interface (GI)*, pages 73–80. Canadian Information Processing Society, 2008.

Eisemann, E., Schwarz, M., Assarsson, U., and Wimmer, M. *Real-time shadows*. AK Peters (CRC Press), 2011. DOI: 10.1201/b11030.

Elek, O., Ritschel, T., Dachsbacher, C., and Seidel, H.-P. Principal-ordinates propagation for real-time rendering of participating media. *Computers & Graphics*, 45:28–39, 2014. DOI: 10.1016/j.cag.2014.08.003.

Engelhardt, T. and Dachsbacher, C. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 119–125. ACM, 2010. DOI: 10.1145/1730804.1730823.

Farrell, T. J., Patterson, M. S., and Wilson, B. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo. *Med. Phys.*, 4(12):879–888, 1992.

Fernando, R., Fernandez, S., Bala, K., and Greenberg, D. P. Adaptive shadow maps. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 387–390. ACM, 2001. DOI: 10.1145/383259.383302.

Frisvad, J. R., Hachisuka, T., and Kjeldsen, T. K. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics (TOG)*, 34(1):5:1–5:12, 2014. DOI: 10.1145/2682629.

Fuchs, M., Raskar, R., Seidel, H.-P., and Lensch, H. P. A. Towards passive 6D reflectance field displays. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 27(3):58:1–58:8, 2008. DOI: 10.1145/1360612.1360657.

Georgiev, I., Křivánek, J., Davidovič, T., and Slusallek, P. Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 31(6):192:1–192:10, 2012. DOI: 10.1145/2366145.2366211.

Georgiev, I., Křivánek, J., Hachisuka, T., Nowrouzezahrai, D., and Jarosz, W. Joint importance sampling of low-order volumetric scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 32(6): 164:1–164:14, 2013. DOI: 10.1145/2508363.2508411.

Ghosh, A. and Heidrich, W. Correlated visibility sampling for direct illumination. *The Visual Computer*, 22 (9–11):693–701, 2006. DOI: 10.1007/s00371-006-0055-x.

Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., and Levin, A. Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 32(6):162:1–162:13, 2013. DOI: 10.1145/2508363.2508377.

Green, P., Kautz, J., Matusik, W., and Durand, F. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 7–14. ACM, 2006. DOI: 10.1145/1111411.1111413.

Green, P., Kautz, J., and Durand, F. Efficient reflectance and visibility approximations for environment map rendering. *Computer Graphics Forum (Proc. of Eurographics)*, 26(3):495–502, 2007. DOI: 10.1111/j.1467-8659.2007.01072.x.

Greger, G., Shirley, P., Hubbard, P., and Greenberg, D. The irradiance volume. *IEEE Computer Graphics and Applications*, pages 32–43, 1998. DOI: 10.1109/38.656788.

Guo, H., Mao, N., and Yuan, X. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(12):2106–2114, 2011. DOI: 10.1109/TVCG.2011.261.

Gutierrez, D., Muñoz, A., Anson, O., and Seron, F. J. Non-linear volume photon mapping. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 291–300. Eurographics Association, 2005. DOI: 10.2312/EGWR/EGSR05/291-300.

Gutierrez, D., Seron, F. J., Lopez-Moreno, J., Sanchez, M. P., Fandos, J., and Reinhard, E. Depicting procedural caustics in single images. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 27(5):120:1–120:9, 2008. DOI: 10.1145/1409060.1409073.

Habel, R., Christensen, P. H., and Jarosz, W. Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, 32(4):27–37, 2013. DOI: 10.1111/cgf.12148.

Hachisuka, T., Pantaleoni, J., and Jensen, H. W. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 31(6):191:1–191:10, 2012. DOI: 10.1145/2366145.2366210.

Hanrahan, P. and Krueger, W. Reflection from layered surfaces due to subsurface scattering. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 165–174. ACM, 1993. DOI: 10.1145/166117.166139.

Hargreaves, S. Deferred shading. GDC 2004, D3D Tutorial Day, 2004.

Harris, M., Sengupta, S., and Owens, J. D. *GPU Gems 3*, chapter 39. Parallel Prefix Sum (Scan) with CUDA, pages 851–876. Addison-Wesley, 2007.

Hašan, M., Pellacini, F., and Bala, K. Direct-to-indirect transfer for cinematic relighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 25(3):1089–1097, 2006. DOI: 10.1145/1141911.1141998.

Hašan, M., Pellacini, F., and Bala, K. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 26(3), 2007. DOI: 10.1145/1276377.1276410.

Hašan, M. and Ramamoorthi, R. Interactive albedo editing in path-traced volumetric materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(2):11:1–11:11, 2013. DOI: 10.1145/2451236.2451237.

Hawkins, T., Einarsson, P., and Debevec, P. Acquisition of time-varying participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):812–815, 2005. DOI: 10.1145/1073204.1073266.

Heckbert, P. S. Fundamentals of texture mapping and image warping. Master's thesis, U.C. Berkeley, 1989.

Heidrich, W. and Seidel, H.-P. Realistic, hardware-accelerated shading and lighting. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 171–178, 1999. DOI: 10.1145/311535.311554.

Hendrickx, Q., Scandolo, L., Eisemann, M., and Eisemann, E. Adaptively layered statistical volumetric obscurance. In *Proc. of High Performance Graphics (HPG)*. ACM, 2015.

Herholz, S., Schairer, T., Schilling, A., and Straßer, W. Screen space spherical harmonic occlusion. In *Proc. of Vision, Modeling and Visualization (VMV)*. Eurographics Association, 2012. DOI: 10.2312/PE/VMV/VMV12/071-078.

Holländer, M., Ritschel, T., Eisemann, E., and Boubekeur, T. Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 1233–1240, 2011. DOI: 10.1111/j.1467-8659.2011.01982.x.

Holroyd, M., Baran, I., Lawrence, J., and Matusik, W. Computing and fabricating multilayer models. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 30(6):187:1–187:8, 2011. DOI: 10.1145/2070781.2024221.

Huang, J., Boubekeur, T., Ritschel, T., Holländer, M., and Eisemann, E. Separable approximation of ambient occlusion. In *Eurographics Short Papers*, 2011. DOI: 10.2312/EG2011/short/029-032.

Ihrke, I. and Magnor, M. Image-based tomographic reconstruction of flames. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 365–373. Eurographics Association, 2004. DOI: 10.1145/1028523.1028572.

Ihrke, I. and Magnor, M. Adaptive grid optical tomography. *Graphical Models*, 68(5):484–495, 2006. DOI: 10.1016/j.gmod.2006.08.001.

Ihrke, I., Goldlücke, B., and Magnor, M. Reconstructing the geometry of flowing water. In *International Conference on Computer Vision (ICCV)*, pages 1055–1060, 2005. DOI: 10.1109/ICCV.2005.202.

Ihrke, I., Ziegler, G., Tevs, A., Theobalt, C., Magnor, M., and Seidel, H.-P. Eikonal rendering: Efficient light transport in refractive objects. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 26(3), 2007. DOI: 10.1145/1276377.1276451.

Jansen, J. and Bavoil, L. Fourier opacity mapping. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 165–172. ACM, 2010. DOI: 10.1145/1730804.1730831.

Jarosz, W. Efficient Monte Carlo methods for light transport in scattering media. PhD thesis, UC San Diego, 2008.

Jarosz, W., Zwicker, M., and Jensen, H. W. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):557—-566, 2008. DOI: 10.1111/j.1467-8659.2008.01153.x.

Jarosz, W., Nowrouzezahrai, D., Sadeghi, I., and Jensen, H. W. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (TOG)*, 30(1):5:1–5:19, 2011a. DOI: 10.1145/1899404.1899409.

Jarosz, W., Nowrouzezahrai, D., Thomas, R., Sloan, P.-P. J., and Zwicker, M. Progressive photon beams. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 30(6):181:1–181:12, 2011b. DOI: 10.1145/2024156.2024215.

Jensen, H. W. and Buhler, J. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):576–581, 2002. DOI: 10.1145/566654.566619.

Jensen, H. W. and Christensen, N. J. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995. DOI: 10.1016/0097-8493(94)00145-O.

Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. A practical model for subsurface light transport. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 511–518. ACM, 2001. DOI: 10.1145/383259.383319.

Jimenez, J., Sundstedt, V., and Gutierrez, D. Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception*, 6(4):23:1–23:15, 2009. DOI: 10.1145/1609967.1609970.

Kajiya, J. T. The rendering equation. *Computer Graphics (Proc. of SIGGRAPH)*, 20(4):143–150, 1986. DOI: 10.1145/15922.15902.

Kajiya, J. T. and Von Herzen, B. P. Ray tracing volume densities. *Computer Graphics (Proc. of SIGGRAPH)*, 18 (3):165–174, 1984. DOI: 10.1145/964965.808594.

Kak, A. C. and Slaney, M. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988. DOI: 10.1137/1.9780898719277.

Kaplanyan, A. and Dachsbacher, C. Cascaded light propagation volumes for real-time indirect illumination. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 99–107. ACM, 2010. DOI: 10.1145/1730804.1730821.

Kautz, J. and McCool, M. D. Approximation of glossy reflection with prefiltered environment maps. In *Proc. of Graphics Interface (GI)*, pages 119–126, 2000.

Kawai, J. K., Painter, J. S., and Cohen, M. F. Radioptimization: Goal based rendering. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 147–154. ACM, 1993. DOI: 10.1145/166117.166136.

Keller, A. Instant radiosity. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 49–56. ACM Press/Addison-Wesley Publishing Co., 1997. DOI: 10.1145/258734.258769.

Keller, A. and Heidrich, W. Interleaved sampling. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*, pages 269–276, 2001. DOI: 10.2312/EGWR/EGWR01/269-276.

Kerr, W. B., Pellacini, F., and Denning, J. D. BendyLights: Artistic control of direct illumination by curving light rays. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 1451–1459, 2010. DOI: 10.1111/j.1467-8659.2010.01742.x.

Kim, T.-Y. and Neumann, U. Opacity shadow maps. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*. Eurographics Association, 2001. DOI: 10.2312/EGWR/EGWR01/177-182.

Klehm, O., Kol, T. R., Seidel, H.-P., and Eisemann, E. Stylized scattering via transfer functions and occluder manipulation. In *Proc. of Graphics Interface (GI)*, pages 115–121. Canadian Information Processing Society, 2015.

Kollig, T. and Keller, A. Efficient illumination by high dynamic range images. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 45–50. Eurographics Association, 2003. DOI: 10.2312/EGWR/EGWR03/045-051.

Kontkanen, J. and Laine, S. Ambient occlusion fields. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 41–48, 2005. DOI: 10.1145/1053427.1053434.

Kontkanen, J., Tabellion, E., and Overbeck, R. S. Coherent out-of-core point-based global illumination. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 1353–1360, 2011. DOI: 10.1111/j.1467-8659.2011.01995.x.

Kroes, T., Schut, D., , and Eisemann, E. Smooth probabilistic ambient occlusion for volume rendering. In *GPU Pro 6*. A K Peters (CRC Press), 2015.

Kulla, C. and Fajardo, M. Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, 31(4):1519–1528, 2012. DOI: 10.1111/j.1467-8659.2012.03148.x.

Křivánek, J., Georgiev, I., Hachisuka, T., Vévoda, P., Šik, M., Nowrouzezahrai, D., and Jarosz, W. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 33(4):103:1–103:13, 2014. DOI: 10.1145/2601097.2601219.

Lafortune, E. P. and Willems, Y. D. Bi-directional path tracing. In *Proc. of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, 1993.

Laine, S., Saransaari, H., Kontkanen, J., Lehtinen, J., and Aila, T. Incremental instant radiosity for real-time indirect illumination. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 277–286, 2007. DOI: 10.2312/EGWR/EGSR07/277-286.

Landis, H. Production-ready global illumination. In *ACM SIGGRAPH Courses*, pages 87–102, 2002.

Langer, M. and Zucker, S. Shape-from-shading on a cloudy day. *Journal of the Optical Society of America A (JOSAA)*, 11(2):467–478, 1994. DOI: 10.1364/JOSAA.11.000467.

Lanman, D., Wetzstein, G., Hirsch, M., Heidrich, W., and Raskar, R. Polarization fields: Dynamic light field display using multi-layer lcds. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 30(6): 186:1–186:10, 2011. DOI: 10.1145/2070781.2024220.

Lauritzen, A. and McCool, M. Layered variance shadow maps. In *Proc. of Graphics Interface (GI)*, pages 139–146. Canadian Information Processing Society, 2008.

Lecocq, P., Marvie, J.-E., Sourimant, G., and Gautron, P. Sub-pixel shadow mapping. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 103–110. ACM, 2014. DOI: 10.1145/2556700.2556709.

Lee, S., Eisemann, E., and Seidel, H.-P. Depth-of-field rendering with multiview synthesis. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5):134:1–134:6, 2009. DOI: 10.1145/1618452.1618480.

Levin, A., Lischinski, D., and Weiss, Y. Colorization using optimization. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 23(3):689–694, 2004. DOI: 10.1145/1015706.1015780.

Linţu, A., Lensch, H. P. A., Magnor, M., El-Abed, S., and Seidel, H.-P. 3D reconstruction of emission and absorption in planetary nebulae. In *Proc. of Volume Graphics*, pages 9–16. Eurographics Association, 2007. DOI: 10.2312/VG/VG07/009-016.

Lokovic, T. and Veach, E. Deep shadow maps. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 385–392. ACM Press/Addison-Wesley Publishing Co., 2000. DOI: 10.1145/344779.344958.

Loos, B. J. and Sloan, P.-P. Volumetric obscurance. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 151–156, 2010. DOI: 10.1145/1730804.1730829.

Ma, L.-Q. and Xu, K. Efficient antialiased edit propagation for images and videos. *Computers & Graphics (Proc. of Virtual Environments and Applications)*, 36(8):1005–1012, 2012. DOI: 10.1016/j.cag.2012.08.001.

Malmer, M., Malmer, F., Assarsson, U., and Holzschuch, N. Fast precomputed ambient occlusion for proximity shadows. *Journal of Graphics Tools (JGT)*, 12(2):59–71, 2007. DOI: 10.1080/2151237X.2007.10129241.

Mardia, K. V. and Jupp, P. E. *Directional Statistics*. John Wiley and Sons, 2000.

Martin, T. and Tan, T.-S. Anti-aliasing and continuity with trapezoidal shadow maps. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 153–160. Eurographics Association, 2004. DOI: 10.2312/EGWR/EGSR04/153-160.

Mattausch, O., Igarashi, T., and Wimmer, M. Freeform shadow boundary editing. *Computer Graphics Forum (Proc. of Eurographics)*, 32:175–184, 2013. DOI: 10.1111/cgf.12037.

Max, N. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 1(2):99–108, 1995. DOI: 10.1109/2945.468400.

Max, N. L. Atmospheric illumination and shadows. *Computer Graphics (Proc. of SIGGRAPH)*, 20(4):117–124, 1986. DOI: 10.1145/15886.15899.

Max, N. L. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, 1988. DOI: 10.1007/BF01905562.

McGuire, M. Ambient occlusion volumes. In *Proc. of High Performance Graphics (HPG)*, pages 47–56, 2010.

McGuire, M., Osman, B., Bukowski, M., and Hennessy, P. The alchemy screen-space ambient obscurance algorithm. In *Proc. of High Performance Graphics (HPG)*, pages 25–32. ACM, 2011. DOI: 10.1145/2018323.2018327.

McNamara, A., Treuille, A., Popović, Z., and Stam, J. Fluid control using the adjoint method. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 23(3):449–456, 2004. DOI: 10.1145/1015706.1015744.

Miller, G. Efficient algorithms for local and global accessibility shading. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 319–326, 1994. DOI: 10.1145/192161.192244.

Mitchell, K. *GPU Gems 3*, chapter 13. Volumetric light scattering as a post-process, pages 275–285. Addison-Wesley, 2007.

Mitra, N. J. and Pauly, M. Shadow art. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5): 156:1–156:7, 2009. DOI: 10.1145/1618452.1618502.

Mittring, M. Finding next gen: CryEngine 2. In *ACM SIGGRAPH Courses*, pages 97–121, 2007. DOI: 10.1145/1281500.1281671.

Mittring, M. A bit more deferred – CryEngine 3. In *Triangle Game Conference*, 2009.

Muñoz, A., Echevarria, J. I., Seron, F. J., and Gutierrez, D. Convolution-based simulation of homogeneous subsurface scattering. *Computer Graphics Forum*, 30(8):2279–2287, 2011. DOI: 10.1111/j.1467-8659.2011.02034.x.

Nalbach, O., Ritschel, T., and Seidel, H.-P. Deep screen space. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 79–86. ACM, 2014a. DOI: 10.1145/2556700.2556708.

Nalbach, O., Ritschel, T., and Seidel, H.-P. Deep screen space for indirect lighting of volumes. In *Proc. of Vision, Modeling and Visualization (VMV)*, pages 143–150, 2014b. DOI: 10.2312/vmv.20141287.

Ng, R., Ramamoorthi, R., and Hanrahan, P. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 22(3):376–381, 2003. DOI: 10.1145/882262.882280.

Nguyen, C. H., Ritschel, T., Myszkowski, K., Eisemann, E., and Seidel, H.-P. 3D material style transfer. *Computer Graphics Forum (Proc. of Eurographics)*, 31(2pt2):431–438, 2012. DOI: 10.1111/j.1467-8659.2012.03022.x.

Nguyen, C. H., Scherzer, D., Ritschel, T., and Seidel, H.-P. Material editing in complex scenes by surface light field manipulation and reflectance optimization. *Computer Graphics Forum (Proc. of Eurographics)*, 32(2): 185–194, 2013. DOI: 10.1111/cgf.12038.

Nicodemus, F., Richmond, J., Hsia, J., Ginsberg, I., and Limperis, T. Geometrical considerations and nomenclature for reflectance. *National Bureau of Standards*, 1977.

Novák, J., Engelhardt, T., and Dachsbacher, C. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 119–124. ACM, 2011. DOI: 10.1145/1944745.1944765.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):60:1–60:11, 2012a. DOI: 10.1145/2185520.2185556.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. Progressive virtual beam lights. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, 31(4):1407–1413, 2012b. DOI: 10.1111/j.1467-8659.2012.03136.x.

Novák, J., Selle, A., and Jarosz, W. Residual ratio tracking for estimating attenuation in participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 33(6):179:1–179:11, 2014. DOI: 10.1145/2661229.2661292.

Nowrouzezahrai, D., Johnson, J., Selle, A., Lacewell, D., Kaschalk, M., and Jarosz, W. A programmable system for artistic volumetric lighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 30(4):29:1–29:8, 2011. DOI: 10.1145/2010324.1964924.

Oat, C. and Sander, P. V. Ambient aperture lighting. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 61–64, 2007. DOI: 10.1145/1230100.1230111.

Obert, J., Křivánek, J., Pellacini, F., Sykora, D., and Pattanaik, S. iCheat: A representation for artistic control of indirect cinematic lighting. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, 27(4): 1217–1223, 2008. DOI: 10.1111/j.1467-8659.2008.01260.x.

Obert, J., Pellacini, F., and Pattanaik, S. Visibility editing for all-frequency shadow design. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 1441–1449, 2010. DOI: 10.1111/j.1467-8659.2010.01741.x.

Okabe, M., Matsushita, Y., Shen, L., and Igarashi, T. Illumination brush: Interactive design of all-frequency lighting. In *Proc. of Pacific Graphics (PG)*, pages 171–180, 2007. DOI: 10.1109/PG.2007.34.

Papas, M., Jarosz, W., Jakob, W., Rusinkiewicz, S., Matusik, W., and Weyrich, T. Goal-based caustics. *Computer Graphics Forum (Proc. of Eurographics)*, 30(2):503–511, 2011. DOI: 10.1111/j.1467-8659.2011.01876.x.

Papas, M., Houit, T., Nowrouzezahrai, D., Gross, M., and Jarosz, W. The magic lens: Refractive steganography. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 31(6):186:1–186:10, 2012. DOI: 10.1145/2366145.2366205.

Papas, M., Regg, C., Jarosz, W., Bickel, B., Jackson, P., Matusik, W., Marschner, S., and Gross, M. Fabricating translucent materials using continuous pigment mixtures. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4):146:1–146:12, 2013. DOI: 10.1145/2461912.2461974.

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., and Stich, M. OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29:66:1–66:13, 2010. DOI: 10.1145/1833349.1778803.

Pegoraro, V. and Parker, S. G. An analytical solution to single scattering in homogeneous participating media. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):329–335, 2009. DOI: 10.1111/j.1467-8659.2009.01372.x.

Pellacini, F., Tole, P., and Greenberg, D. P. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):563–566, 2002. DOI: 10.1145/566654.566617.

Pellacini, F., Vidimče, K., Lefohn, A., Mohr, A., Leone, M., and Warren, J. Lpics: A hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24 (3):464–470, 2005. DOI: 10.1145/1073204.1073214.

Pellacini, F., Battaglia, F., Morley, R. K., and Finkelstein, A. Lighting with paint. *ACM Transactions on Graphics (TOG)*, 26(2), 2007. DOI: 10.1145/1243980.1243983.

Perlin, K. and Hoffert, E. M. Hypertexture. *Computer Graphics (Proc. of SIGGRAPH)*, 23(3):253–262, 1989. DOI: 10.1145/74334.74359.

Perlin, K. An image synthesizer. *Computer Graphics (Proc. of SIGGRAPH)*, 19(3):287–296, 1985. DOI: 10.1145/325165.325247.

Pharr, M. and Green, S. *Ambient occlusion*, volume 1, pages 279–292. Addison Wesley, 2004.

Poulin, P. and Fournier, A. Lights from highlights and shadows. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 31–38. ACM, 1992. DOI: 10.1145/147156.147160.

Poulin, P. and Fournier, A. Painting surface characteristics. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*, pages 160–169. Springer Vienna, 1995. DOI: 10.1007/978-3-7091-9430-0_16.

Preim, B. and Botha, C. P. *Visual Computing for Medicine: Theory, Algorithms, and Applications*. Morgan Kaufmann Publishers Inc., 2 edition, 2013.

Raab, M., Seibert, D., and Keller, A. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods*, pages 591–605. Springer Berlin Heidelberg, 2006. DOI: 10.1007/978-3-540-74496-2_35.

Ragan-Kelley, J., Kilpatrick, C., Smith, B. W., Epps, D., Green, P., Hery, C., and Durand, F. The lightspeed automatic interactive lighting preview system. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 26(3): 25:1–25:11, 2007. DOI: 10.1145/1276377.1276409.

Ramamoorthi, R. and Hanrahan, P. An efficient representation for irradiance environment maps. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 497–500. ACM, 2001. DOI: 10.1145/383259.383317.

Ramamoorthi, R. and Hanrahan, P. Frequency space environment map rendering. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):517–526, 2002. DOI: 10.1145/566654.566611.

Reeves, W. T., Salesin, D. H., and Cook, R. L. Rendering antialiased shadows with depth maps. *Computer Graphics (Proc. of SIGGRAPH)*, 21(4):283–291, 1987. DOI: 10.1145/37402.37435.

Reiner, T., Kaplanyan, A., Reinhard, M., and Dachsbacher, C. Selective inspection and interactive visualization of light transport in virtual scenes. *Computer Graphics Forum (Proc. of Eurographics)*, 31(2pt4):711–718, 2012. DOI: 10.1111/j.1467-8659.2012.03050.x.

Ren, Z., Zhou, K., Lin, S., and Guo, B. Gradient-based interpolation and sampling for real-time rendering of inhomogeneous, single-scattering media. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 27(7): 1945–1953, 2008. DOI: 10.1111/j.1467-8659.2008.01343.x.

Ritschel, T., Grosch, T., Kim, M. H., Seidel, H.-P., Dachsbacher, C., and Kautz, J. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 27 (5):129:1–129:8, 2008. DOI: 10.1145/1409060.1409082.

Ritschel, T., Engelhardt, T., Grosch, T., Seidel, H.-P., Kautz, J., and Dachsbacher, C. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5):132:1–132:8, 2009a. DOI: 10.1145/1618452.1618478.

Ritschel, T., Grosch, T., and Seidel, H.-P. Approximating dynamic global illumination in image space. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 75–82, 2009b. DOI: 10.1145/1507149.1507161.

Ritschel, T., Okabe, M., Thormählen, T., and Seidel, H.-P. Interactive reflection editing. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5):129:1–129:7, 2009c. DOI: 10.1145/1618452.1618475.

Ritschel, T., Thormählen, T., Dachsbacher, C., Kautz, J., and Seidel, H.-P. Interactive on-surface signal deformation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29(4):36:1–36:8, 2010. DOI: 10.1145/1778765.1778773.

Ritschel, T., Eisemann, E., Ha, I., Kim, J. D. K., and Seidel, H.-P. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, 30(8):2258–2269, 2011. DOI: 10.1111/j.1467-8659.2011.01998.x.

Ritschel, T., Dachsbacher, C., Grosch, T., and Kautz, J. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1):160–188, 2012. DOI: 10.1111/j.1467-8659.2012.02093.x.

Ropinski, T., Praßni, J., Steinicke, F., and Hinrichs, K. Stroke-based transfer function design. In *Proc. of IEEE/EG Symposium on Volume and Point-Based Graphics*, pages 41–48, 2008. DOI: 10.2312/VG/VG-PBG08/041-048.

Rushmeier, H. E. and Torrance, K. E. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics (Proc. of SIGGRAPH)*, 21(4):293–302, 1987. DOI: 10.1145/37402.37436.

Rushmeier, H. E. Realistic image synthesis for scenes with radiatively participating media. PhD thesis, Cornell University, 1988.

Saito, T. and Takahashi, T. Comprehensible rendering of 3-D shapes. *Computer Graphics (Proc. of SIGGRAPH)*, 24(4):197–206, 1990. DOI: 10.1145/97880.97901.

Salvi, M. Rendering filtered shadows with exponential shadow maps. In *ShaderX 6.0*, pages 257–274. Charles River Media, 2008.

Schmid, J., Sumner, R. W., Bowles, H., and Gross, M. Programmable motion effects. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29(4):57:1–57:9, 2010. DOI: 10.1145/1778765.1778794.

Schmidt, T.-W., Novák, J., Meng, J., Kaplanyan, A. S., Reiner, T., Nowrouzezahrai, D., and Dachsbacher, C. Path-space manipulation of physically-based light transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4):129, 2013. DOI: 10.1145/2461912.2461980.

Schmidt, T.-W., Pellacini, F., Nowrouzezahrai, D., Jarosz, W., and Dachsbacher, C. State of the art in artistic editing of appearance, lighting, and material. In *Eurographics State of the Art Reports*. Eurographics Association, 2014. DOI: 10.2312/egst.20141041.

Schoeneman, C., Dorsey, J., Smits, B., Arvo, J., and Greenberg, D. Painting with light. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 143–146. ACM, 1993. DOI: 10.1145/166117.166135.

Schwarz, M. and Seidel, H.-P. Fast parallel surface and solid voxelization on GPUs. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 29(6):179:1–179:10, 2010. DOI: 10.1145/1882261.1866201.

Secord, A., Heidrich, W., and Streit, L. Fast primitive distribution for illustration. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*, pages 215–226. Eurographics Association, 2002. DOI: 10.2312/EGWR/EGWR02/215-226.

Sen, P., Cammarano, M., and Hanrahan, P. Shadow silhouette maps. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 22(3):521–526, 2003. DOI: 10.1145/882262.882301.

Shanmugam, P. and Arikan, O. Hardware accelerated ambient occlusion techniques on GPUs. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 73–80, 2007. DOI: 10.1145/1230100.1230113.

Shewchuk, J. R. An introduction to the conjugate gradient method without the agonizing pain. Technical report, 1994.

Sintorn, E., Eisemann, E., and Assarsson, U. Sample based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 1285–1292, 2008. DOI: 10.1111/j.1467-8659.2008.01267.x.

Sloan, P.-P., Kautz, J., and Snyder, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):527–536, 2002. DOI: 10.1145/566654.566612.

Song, Y., Tong, X., Pellacini, F., and Peers, P. Subedit: A representation for editing measured heterogeneous subsurface scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 28(3):31:1–31:10, 2009. DOI: 10.1145/1531326.1531337.

Spencer, B., Jones, M. W., and Lim, I. S. A visualization tool used to develop new photon mapping techniques. *Computer Graphics Forum*, 34(1):127–140, 2015. DOI: 10.1111/cgf.12464.

Stam, J. Multiple scattering as a diffusion process. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*, pages 41–50, 1995. DOI: 10.1007/978-3-7091-9430-0_5.

Stam, J. Stable fluids. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999. DOI: 10.1145/311535.311548.

Stamminger, M. and Drettakis, G. Perspective shadow maps. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):557–562, 2002. DOI: 10.1145/566654.566616.

Straßer, W. Schnelle Kurven- und Flächendarstellung auf graphischen Sichtgeräten. PhD thesis, Technische Universität zu Berlin, 1974.

Sun, B., Ramamoorthi, R., Narasimhan, S. G., and Nayar, S. K. A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1040–1049, 2005. DOI: 10.1145/1073204.1073309.

Sun, X., Zhou, K., Lin, S., and Guo, B. Line space gathering for single scattering in large scenes. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29(4):54:1–54:8, 2010. DOI: 10.1145/1778765.1778791.

Szirmay-Kalos, L., Umenhoffer, T., Tóth, B., Szécsi, L., and Sbert, M. Volumetric ambient occlusion for real-time rendering and games. *IEEE Computer Graphics and Applications*, pages 70–79, 2010. DOI: 10.1109/MCG.2010.19.

Timonen, V. Line-sweep ambient obscurance. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 97–105, 2013. DOI: 10.1111/cgf.12155.

Toth, B. and Umenhofer, T. Real-time volumetric lighting in participating media. In *Eurographics Short Papers*, 2009. DOI: 10.2312/egs.20091048.

Treuille, A., McNamara, A., Popović, Z., and Stam, J. Keyframe control of smoke simulations. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 22(3):716–723, 2003. DOI: 10.1145/882262.882337.

Vardis, K., Papaioannou, G., and Gaitatzes, A. Multi-view ambient occlusion with importance sampling. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D)*, pages 111–118. ACM, 2013. DOI: 10.1145/2448196.2448214.

Veach, E. and Guibas, L. J. Bidirectional estimators for light transport. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*, pages 147–162, 1994.

Veach, E. and Guibas, L. J. Optimally combining sampling techniques for monte carlo rendering. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 419–428. ACM, 1995. DOI: 10.1145/218380.218498.

Veach, E. and Guibas, L. J. Metropolis light transport. In *Annual Conference Series (Proc. of SIGGRAPH)*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997. DOI: 10.1145/258734.258775.

Veach, E. Robust Monte Carlo methods for light transport simulation. PhD thesis, Stanford University, 1997.

Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1098–1107, 2005. DOI: 10.1145/1073204.1073318.

Walter, B., Zhao, S., Holzschuch, N., and Bala, K. Single scattering in refractive media with triangle mesh boundaries. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 28(3):92:1–92:8, 2009. DOI: 10.1145/1531326.1531398.

Wang, B., Huang, J., Buchholz, B., Meng, X., and Boubekeur, T. Factorized point based global illumination. *Computer Graphics Forum (Proc. of EG Symposium on Rendering)*, pages 117–123, 2013. DOI: 10.1111/cgf.12157.

Wetzstein, G., Lanman, D., Heidrich, W., and Raskar, R. Layered 3D: Tomographic image synthesis for attenuation-based light field and high dynamic range displays. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 30(4):95:1–95:12, 2011. DOI: 10.1145/2010324.1964990.

Wetzstein, G., Lanman, D., Hirsch, M., and Raskar, R. Tensor displays: Compressive light field synthesis using multilayer displays with directional backlighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31 (4):80:1–80:11, 2012. DOI: 10.1145/2185520.2185576.

Weyrich, T., Peers, P., Matusik, W., and Rusinkiewicz, S. Fabricating microgeometry for custom surface reflectance. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 28(3):32:1–32:6, 2009. DOI: 10.1145/1531326.1531338.

Whitted, T. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980. DOI: 10.1145/358876.358882.

Williams, L. Casting curved shadows on curved surfaces. *Computer Graphics (Proc. of SIGGRAPH)*, 12(3): 270–274, 1978. DOI: 10.1145/965139.807402.

Wimmer, M., Scherzer, D., and Purgathofer, W. Light space perspective shadow maps. In *Rendering Techniques (Proc. of EG Symposium on Rendering)*, pages 143–151. Eurographics Association, 2004. DOI: 10.2312/EGWR/EGSR04/143-151.

Woodcock, E., Murphy, T., Hemmings, P., and Longworth, T. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Computing Methods to Reactor Problems*, 1965.

Wyman, C. and Ramsey, S. Interactive volumetric shadows in participating media with single-scattering. In *IEEE Symp. on Interactive Ray Tracing*, pages 87–92, 2008. DOI: 10.1109/RT.2008.4634627.

Wyman, C. Voxelized shadow volumes. In *Proc. of High Performance Graphics (HPG)*, pages 33–40. ACM, 2011. DOI: 10.1145/2018323.2018329.

Wyman, C. and Dai, Z. Imperfect voxelized shadow volumes. In *Proc. of High Performance Graphics (HPG)*, pages 45–52. ACM, 2013. DOI: 10.1145/2492045.2492050.

Yu, I., Cox, A., Kim, M. H., Ritschel, T., Grosch, T., Dachsbacher, C., and Kautz, J. Perceptual influence of approximate visibility in indirect illumination. *ACM Transactions on Applied Perception*, 6(4):24:1–24:14, 2009. DOI: 10.1145/1609967.1609971.

Zhang, F., Sun, H., Xu, L., and Lun, L. K. Parallel-split shadow maps for large-scale virtual environments. In *Proc. of Virtual Reality Continuum and Its Applications*, pages 311–318. ACM, 2006. DOI: 10.1145/1128923.1128975.

Zhao, S., Ramamoorthi, R., and Bala, K. High-order similarity relations in radiative transfer. *ACM Trans. Graph.*, 33(4):104:1–104:12, 2014. DOI: 10.1145/2601097.2601104.

Zhou, K., Hou, Q., Gong, M., Snyder, J., Guo, B., and Shum, H.-Y. Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. In *Proc. of Pacific Graphics (PG)*, pages 116–125. IEEE Computer Society, 2007. DOI: 10.1109/PG.2007.28.

Zhou, K., Ren, Z., Lin, S., Bao, H., Guo, B., and Shum, H.-Y. Real-time smoke rendering using compensated ray marching. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 27(3):36:1–36:12, 2008. DOI: 10.1145/1360612.1360635.

Zhukov, S., Iones, A., and Kronin, G. An ambient light illumination model. In *Rendering Techniques (Proc. of EG Workshop on Rendering)*, pages 45–56, 1998. DOI: 10.1007/978-3-7091-6453-2_5.

Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., and Yoon, S.-E. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proc. of Eurographics - State of the Art Reports)*, 34(2):667–681, 2015. DOI: 10.1111/cgf.12592.