



Fast and Interactive Ray-based Rendering

Dissertation

submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy (Ph.D.)

of the

Department of Computer Science,
Brunel University London

by

Thorsten Roth

Submission Date: 2020-09-24

Declaration

I hereby declare that this thesis is solely completed by the candidate, Thorsten Roth. The original research work has not been presented for the award of any other degree in the past. Some work in it has been published previously, which is stated in the text where relevant. All sources of material have been properly acknowledged and references have been provided.

Publications

This is a list of publications lead-authored and co-authored by the author of this thesis during the PhD time frame:

- (Weier, Maiero, et al. 2014a):
M. Weier, J. Maiero, T. Roth, A. Hinkenjann, S. Slusallek: *Enhancing Rendering Performance with View-Direction-Based Rendering Techniques for Large, High Resolution Multi-Display Systems*, 11. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR, 2014
- (Weier, Maiero, et al. 2014b):
M. Weier, J. Maiero, T. Roth, A. Hinkenjann, S. Slusallek: *Lazy Details for Large High Resolution Displays*, SIGGRAPH Asia, 2014, Poster
- (Sigitov, Roth, et al. 2015):
A. Sigitov, T. Roth, A. Hinkenjann: *Enabling Global Illumination Rendering on Large, High-Resolution Displays*, 8th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2015
- (Hinkenjann, Jato, et al. 2015):
A. Hinkenjann, O. Jato, J. Maiero, T. Roth, M. Weier: *High Quality Rendering and Visualization at the Institute of Visual Computing, Sankt Augustin, Germany*, IEEE VR 2015, Lab Presentation
- (Szeracki, Roth, et al. 2015):
S. Szeracki, T. Roth, A. Hinkenjann, Y. Li: *Boosting Histogram-Based Denoising Methods with GPU Optimizations*, 12. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR, 2015
- (Roth, Weier, et al. 2015):
T. Roth, M. Weier, J. Maiero, A. Hinkenjann, Y. Li: *Guided High-Quality Rendering*, 11th International Symposium on Visual Computing (ISVC), 2015
- (Jato, Weier, et al. 2016):
O. Jato, M. Weier, J. Maiero, D. Scherfgen, T. Roth, P. Frericks, O. Stefani, M. Bues, A. Hinkenjann, E. Kruijff: *OLIVE: Simulation within Human-Centric Lighting Environments*, Proceedings of the 13. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR, 2016
- (Weier, Roth, et al. 2016):
M. Weier, T. Roth, E. Kruijff, A. Hinkenjann, A. Pérard-Gayot, P. Slusallek, Y. Li: *Foveated Real-Time Ray Tracing for Head-Mounted Displays*, Computer Graphics Forum, 35-7, 2016, Okinawa, Japan
- (Roth, Weier, et al. 2016):
T. Roth, M. Weier, A. Hinkenjann, Y. Li, P. Slusallek: *An Analysis of Eye-Tracking Data in Foveated Ray Tracing*, Proceedings of the 2016 Workshop on Eye Tracking and Visualization
- (Frericks, Roth, et al. 2017):
P. Frericks, T. Roth, A. Hinkenjann, E. Kruijff: *A Framework for Cluster-based*

Rendering and Postprocessing, presented at the 10th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2017

- (Roth, Weier, et al. 2017):
T. Roth, M. Weier, A. Hinkenjann, Y. Li, P. Slusallek: *A Quality-Centered Analysis of Eye Tracking Data in Foveated Rendering*, Journal of Eye Movement Research, 10-5, 2017
- (Weier, Stengel, et al. 2017):
M. Weier, M. Stengel, T. Roth, P. Didyk, E. Eisemann, M. Eisemann, S. Grogorick, A. Hinkenjann, E. Kruijff, M. Magnor, K. Myszkowski, P. Slusallek: *Perception-Driven Accelerated Rendering*, Computer Graphics Forum, 36-2, 2017
- (Weier, Roth, et al. 2018a):
M. Weier, T. Roth, A. Hinkenjann, P. Slusallek: *Foveated Depth-of-Field Filtering in Head-mounted Displays*, ACM Transactions on Applied Perception, 15-4, 2018
- (Weier, Roth, et al. 2018b):
M. Weier, T. Roth, A. Hinkenjann, P. Slusallek: *Foveated Depth-of-field Filtering in Head-mounted Displays*, Proceedings of the 15th ACM Symposium on Applied Perception, SAP 2018, Vancouver, B.C., **received the best paper award**
- (Weier, Roth, et al. 2018c):
M. Weier, T. Roth, A. Hinkenjann, P. Slusallek: *Predicting the Gaze Depth in Head-mounted Displays using Multiple Feature Regression*, Proceedings of the 2018 ACM Symposium on Eye Tracking Research and Applications, ETRA 2018, 19:1–19:9, 2018
- (Roth, Weier, et al. 2019):
T. Roth, M. Weier, P. Bauszat, A. Hinkenjann, Y. Li: *Hash-based Hierarchical Caching for Interactive Previews in Global Illumination Rendering*, Computer Graphics and Visual Computing (CGVC 2019), Bangor University, Wales, United Kingdom, 2019
- (Roth, Weier, et al. 2020):
T. Roth, M. Weier, P. Bauszat, A. Hinkenjann, Y. Li: *Hash-based Hierarchical Caching and Layered Filtering for Interactive Previews in Global Illumination Rendering*, Computers, 9(1), 17, 2020, **featured as the cover story**

Abstract

Despite their age, ray-based rendering methods are still a very active field of research with many challenges when it comes to interactive visualization. In this thesis, we present our work on *Guided High-Quality Rendering*, *Foveated Ray Tracing for Head-Mounted Displays* and *Hash-based Hierarchical Caching and Layered Filtering*.

Our system for Guided High-Quality Rendering allows for guiding the sampling rate of ray-based rendering methods by a user-specified Region of Interest (RoI). We propose two interaction methods for setting such an RoI when using a large display system and a desktop display, respectively. This makes it possible to compute images with a heterogeneous sample distribution across the image plane. Using such a non-uniform sample distribution, the rendering performance inside the RoI can be significantly improved in order to judge specific image features. However, a modified scheduling method is required to achieve sufficient performance. To solve this issue, we developed a scheduling method based on sparse matrix compression, which has shown significant improvements in our benchmarks. By filtering the sparsely sampled image appropriately, large brightness variations in areas outside the RoI are avoided and the overall image brightness is similar to the ground truth early in the rendering process.

When using ray-based methods in a VR environment on head-mounted display devices, it is crucial to provide sufficient frame rates in order to reduce motion sickness. This is a challenging task when moving through highly complex environments and the full image has to be rendered for each frame. With our foveated rendering system, we provide a perception-based method for adjusting the sample density to the user's gaze, measured with an eye tracker integrated into the HMD. In order to avoid disturbances through visual artifacts from low sampling rates, we introduce a reprojection-based rendering pipeline that allows for fast rendering and temporal accumulation of the sparsely placed samples. In our user study, we analyse the impact our system has on visual quality. We then take a closer look at the recorded eye tracking data in order to determine tracking accuracy and connections between different fixation modes and perceived quality, leading to surprising insights.

For previewing global illumination of a scene interactively by allowing for free scene exploration, we present a hash-based caching system. Building upon the concept of linkless octrees, which allow for constant-time queries of spatial data, our framework is suited for rendering such previews of static scenes. Non-diffuse surfaces are supported by our hybrid reconstruction approach that allows for the visualization of view-dependent effects. In addition to our caching and reconstruction technique, we introduce a novel layered filtering framework, acting as a hybrid method between path space and image space filtering, that allows for the high-quality denoising of non-diffuse materials. Also, being designed as a framework instead of a concrete filtering method, it is possible to adapt most available denoising methods to our layered approach instead of relying only on the filtering of primary hitpoints.

Acknowledgements

First and foremost, I want to thank Jessica Chromik: You have been my main source of sanity and reason during most of my PhD. Through these years, I stumbled more than once; With all that has happened, it has been a very difficult and delightful experience at the same time, and good and bad, failure and success are sometimes so close.

I would like to thank my supervisors, André Hinkenjann and Yongmin Li, and all my dear colleagues at the Institute of Visual Computing, for fruitful and sometimes mind-boggling discussions, wandering minds, and the occasional (or regular) cup of coffee. I cannot express my gratitude sufficiently in regular words, but nonetheless I want to explicitly thank my colleague and friend Martin Weier for working on so many projects together with me throughout the years, through hardships and ease alike. It was both an honor and an insightful experience to work with you.

Thank you, Jens Maiero for many years of talking sense and nonsense, discussing ideas and frustrations, and for going on this journey together with me in 2013. I am very happy that we could not only start together, but will likely also both reach the finish line in the same year.


Exceptional thanks to all the people who have shared an office with me throughout the years: Katharina Stollenwerk (I'm seriously going to miss that diabolic laugh!), Oliver Jato (let's do some time sheets together!), Jessica Millberg, Anton Sigitov, Florian "Nougatring" Mannuß, Florian Bingel, Christoph Pomrehn, and Sandra Felsner. Thanks to Christina Trepkowski for helping me out when statistics were just too much for me! My deepest gratitude goes to Andreas Priesnitz for being an extremely supportive and understanding colleague when working together in projects unrelated to my PhD.

Thank you to all of my friends, especially (but in no particular order) to Denise Pommerening for just being there in difficult times and for some great trips, Mamdouh Maduar for being the awesome guy you are, Tobias Voß for over thirty years of friendship, Silvio Hentschel for your authenticity, open mind and for just being you, Julia Krämer and Ben for just being awesome, Diana Kurch for being a great and understanding friend even when we're far apart, and Stefanie Höller for many years of friendship. While we have not even met in person, Christoph Schempershofe, Anna-Lena Haider and Tobias Haider are a proof that gaming can make people grow really close over the years.

Thank you to Philipp Slusallek for inspiring ideas when I worked together with Martin, Pablo Bauszat for participating in bringing that HashCache idea to life when some random guy from the internet just asked you, and all my co-authors that I have not explicitly mentioned here. Thank you for your support.

Thanks to my family, especially my mother Solveig Roth, my uncle Helmut Brandt and my aunt Ingrid Brandt for supporting me.

Last but not least, thank you to our former dean Kurt-Ulrich Witt for inspiring me to start my PhD: "Wenn Sie es jetzt nicht machen, machen Sie es nie!" (*"If you don't do it now, you're never going to do it"*).

– dedicated to everyone special in my life 

Acronyms

ANOVA	analysis of variance
AABB	axis-aligned bounding box
BDPT	bidirectional path tracing
BRDF	bidirectional reflectance distribution function
BIH	bounding interval hierarchy
CDF	cumulative distribution function
CK	Country Kitchen
CPD	cycles per degree
CRF	caching resolution factor
CVA	comfortable viewing angle
DCNN	deep convolutional neural network
CNN	convolutional neural network
DGI	distance-guide image
DNN	deep neural network
FoV	field of view
FPS	frames per second
FRC	foveal region configuration
GI	global illumination
HMD	head-mounted display
HDR	high dynamic range
HVS	human visual system
LHRDW	large high-resolution display wall
LoD	level-of-detail
LUT	lookup table
MC	Monte Carlo
MLT	Metropolis light transport
MS-SSIM	multi-scale structural similarity
MVP Matrix	model-view-projection matrix

- OoI** object of interest
- PoR** point of regard
- PSNR** peak signal-to-noise ratio
- RMS** root mean square
- reIMSE** relative mean-square error
- RoI** region of interest
- SMC** sparse matrix compression
- SoA** Streets of Asia
- SPEM** smooth pursuit eye movement
- SPP** samples per pixel
- SURE** Stein's unbiased risk estimator
- VR** virtual reality

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Aims and Objectives	3
1.3	Thesis Overview	4
2	Background and Literature Review	6
2.1	Concepts of Ray-based Rendering	6
2.1.1	Ray Casting	7
2.1.2	Ray Tracing	7
2.1.3	Path Tracing	7
2.1.4	Bidirectional Path Tracing	11
2.1.5	Metropolis Light Transport	12
2.1.6	Standard Extensions of Ray-based Rendering	12
2.2	Other Ray-based Methods, Extensions and Optimizations	14
2.2.1	Photon Mapping	15
2.2.2	Radiosity	19
2.2.3	General Sampling Techniques	23
2.2.4	Machine Learning	25
2.2.5	Other Methods and Aspects	27
2.3	Perceptual Considerations and Gaze-Contingent Rendering	29
2.3.1	The Human Visual System: Basic Concepts and Limitations	29
2.3.2	Perceived Image Quality and Quality Assessment	33
2.3.3	Foveated/Gaze-Contingent Rendering	34
2.4	Denoising	37
2.5	Caching and Reprojection	39
2.6	Discussion and Conclusion	43
3	Guided High-Quality Rendering	44
3.1	Introduction	44
3.2	System Description	45
3.2.1	Building Blocks	45
3.2.2	Distance Measure	47
3.2.3	Interaction	51
3.2.4	Scheduling	53
3.2.5	Rendering	54
3.2.6	Filtering	54
3.2.7	Display	55
3.3	Results	55
3.3.1	Visual Quality	55
3.3.2	Benchmarks	57
3.4	Discussion and Conclusion	66

4	Foveated Ray Tracing and Eye Tracking Data	68
4.1	Introduction	69
4.2	System Description	71
4.2.1	Ray Generation and Ray Tracing	73
4.2.2	Reprojection	74
4.2.3	Handling Reprojection Errors	76
4.2.4	Cache Update and Merging	77
4.2.5	Post-Processing	79
4.3	Experimental Evaluation: Benchmarks	79
4.4	Experimental Evaluation: User Study	82
4.4.1	Experimental Procedure and Design	83
4.4.2	Results	85
4.5	Experimental Evaluation: Analysis of Eye Tracking Data	87
4.5.1	Methods	87
4.5.2	Results	89
4.6	Discussion and Conclusion	97
5	Hash-based Hierarchical Caching and Layered Filtering	99
5.1	Introduction	99
5.2	Method	100
5.2.1	Cache Structure	101
5.2.2	Caching	104
5.2.3	Reconstruction	106
5.2.4	Layered Filtering	110
5.3	Results and Evaluation	114
5.3.1	Visual Quality	114
5.3.2	Performance	121
5.3.3	Memory Requirements	123
5.3.4	Comparison to State-of-the-Art	123
5.4	Discussion and Conclusion	125
6	Conclusion	127
6.1	Summary	127
6.2	Contributions	129
6.2.1	Guided High-Quality Rendering	130
6.2.2	Foveated Ray Tracing in Head-Mounted Displays	130
6.2.3	Hash-based Hierarchical Caching and Layered Filtering	130
6.3	Future Research and Impact of Technological Developments	131
6.3.1	Guided High-Quality Rendering	131
6.3.2	Foveated Ray Tracing in Head-mounted Displays	132
6.3.3	Hash-based Hierarchical Caching and Layered Filtering	133

List of Figures

2.1	A comparison of ray casting and ray tracing	8
2.2	Limiting recursion depth with transparent and reflective materials	9
2.3	Direct, indirect, and full global illumination	10
2.4	Bidirectional vs. standard path tracing, equal-time comparison	11
2.5	Bidirectional path tracing with 40 samples per pixel vs. Metropolis light transport with 250 mutations per pixel, equal-time comparison	12
2.6	The effect of (multiple) importance sampling	14
3.1	Building blocks of the proposed framework	46
3.2	Projection process of the ellipse onto the display system	48
3.3	Exemplary shape of the resulting distance-guide image	49
3.4	ART’s interaction device <i>FlyStick</i>	52
3.5	Schematic illustration of the relevant part of our LHRDW	56
3.6	Comparison between filtering methods and unfiltered image	56
3.7	Unfiltered vs. filtered images displayed on our LHRDW	57
3.8	Reference renderings of the scenes used for benchmarking	58
3.9	Computed samples and rendering times for various FoV settings	60
3.10	Tested RoI sizes for the Cornell Box scene	61
3.11	Tested region of interest (RoI) sizes for the Urban Sprawl scene	61
3.12	Tested RoI sizes for the Hairball scene	61
3.13	Rendering time for the Cornell Box scene with and without SMC	62
3.14	Rendering times for the Urban Sprawl scene with and without SMC	62
3.15	Rendering times for the Hairball scene with and without SMC	63
3.16	Relative rendering times for SMC-based scheduling (Cornell Box scene)	64
3.17	Relative rendering times for SMC-based scheduling (Urban Sprawl scene)	64
3.18	Relative rendering times for SMC-based scheduling (Hairball scene)	65
3.19	Relative rendering times for SMC-based scheduling (Hairball scene, alternative camera position)	65
3.20	Asymptotic error rate for various sampling probabilities	67
4.1	Effects of different configurations for the foveal region	68
4.2	The evolution of resolutions in modern HMDs	70
4.3	Building blocks of our reprojection pipeline	73
4.4	Foveal falloff function	74
4.5	Reprojection from frame $t - 1$ to frame t	75
4.6	Scenes used for benchmarks and user studies of our implementation	80
4.7	Average total rendering times foveated vs. RT	81
4.8	Influence of the FRC on FPS for Sponza (varying lighting)	82
4.9	Likert-scale ratings for Q1 (<i>The shown sequence was free of visual artifacts.</i>) for all scenes grouped by foveal region configurations	86

4.10	Likert-scale ratings of perceived visual artifacts for small, medium, large and full foveal regions	86
4.11	Tracking precision vs. fixation target's distance to the image center	88
4.12	CDF of measured fixation accuracy	91
4.13	Gaze deviation for all individual scenes, fixed and moving targets	92
4.14	Quality for all combinations of scenes and fixation modes	93
4.15	Quality ratings for fixation modes and FRCs (Rungholt)	94
4.16	Quality ratings for fixation modes and FRCs (Sponza)	94
4.17	Quality ratings for fixation modes and FRCs (Tunnel_Geom)	95
4.18	Quality ratings for fixation modes and FRCs (Tunnel_Maps)	95
4.19	Eccentricity-dependent mean quality measurements and eccentricity distributions for all scenes in free focus mode	96
5.1	Computation of the occupied grid cells and hashing	102
5.2	An overview of the caching process	104
5.3	Sampling parameters for a single path	105
5.4	An overview of the reconstruction process	107
5.5	Visual Comparison for jittering and denoising	109
5.6	Two dimensional example of the possible issue with spatial jittering	110
5.7	Layered filtering process	111
5.8	Unfiltered propagation of light bounces	113
5.9	Interleaved comparison: Layered vs. traditional filtering	114
5.10	relMSE and MS-SSIM for both presented scenes	116

List of Tables

4.1	An overview of the buffers used in the reprojection and accumulation process	72
4.2	Times in milliseconds for each stage of the pipeline in comparison to a full renderer showing the speedup of our approach	81
5.1	Data densities and memory requirements	125

Chapter 1

Introduction

1.1 Motivation

Visualizing the content of a scene described by its geometry, lighting and material information can be a challenging task. Even with today's hardware capabilities, the desired visual quality, geometric complexity and various other properties may lead to sub-par results, especially since visual quality expectations are ever increasing. Besides rasterization, which has been central to rendering especially in the consumer field, ray-based techniques have been a popular tool that has evolved over the last 52 years, with Appel (1968) first describing the basic *ray casting* algorithm in 1968, while Whitted (1980) introduced *ray tracing* in 1980.

Despite its age, ray-based rendering is still a very active field of research, partially because support for such methods has just been implemented in consumer hardware in recent years. Historically, it has grown from a method for finding the closest visible scene geometry for a specific camera perspective (basic ray casting) to a universal visualization tool that supports very basic rendering and photorealistic image synthesis alike. At the same time, the computational complexity of methods for generating photorealistic imagery is still challenging, as they rely on the physically-based process of global illumination (GI).

The most important concept in the field of GI is the rendering equation, presented by Kajiya (1986). It fully describes how light distributes on the surfaces in a scene after being emitted from an arbitrary number of light sources. Being formulated as an integral equation, its recursive nature makes it analytically untractable. Instead, numeric Monte Carlo (MC) methods are often used for computing solutions to the equation.

Such methods rely on an underlying stochastic process, where randomly generated rays are traced through a scene. These rays originate at the virtual camera and are then reflected at their intersection points with the scene's surfaces according to

material and illumination properties, generating additional rays recursively. Connecting consecutively generated rays creates a path, which is commonly referred to as a *sample* in GI rendering.

Two challenging phenomena occur when computing GI with MC methods: Due to the stochastic nature of the process, the generated images exhibit high-frequency noise, and reducing the noise becomes more and more computationally demanding the more samples are taken. With an asymptotic error of $\mathcal{O}(\sqrt{n^{-1}})$, where n is the number of samples, the remaining error declines rapidly at the start, but slows down rather quickly towards higher sample counts. Consequently, four times as many samples are required to reduce the noise of an image by 50%. This may become a major issue especially when such methods are used in performance-critical applications, where image quality is a crucial factor for judging certain image properties or for analyzing the displayed object's properties, respectively. This is the case in areas like design review or architectural visualization, and is certainly extendible to games, which pose the prime example of a field where the latest graphics hardware is utilized in the consumer market. However, determining the number of samples required to achieve a specific noise or error level without performing the actual rendering task is generally not possible.

While current 3D modeling software often also supports methods for preview rendering that do not rely on the full process of accurate global illumination computation, previews may not meet the specific requirements of providing a sufficient fidelity for the task at hand. Possible methods for improving the situation by also providing the means for selectively focusing the computational effort on specific image areas are discussed later in this thesis.

A field that has gained much interest over the last few years because of the increasingly wide availability of relevant consumer hardware is virtual reality (VR). Using ray-based rendering methods in this field is challenging, as high frame rates and low latencies are crucial when using devices such as head-mounted displays (HMDs) to avoid motion sickness. Recently, several methods from the field of gaze-contingent rendering have emerged, also relying on eye tracking devices, that are nowadays often integrated into HMDs. This allows for adapting the rendering process to the image regions that are actually focused by the user.

Another relevant field of research is given by caching algorithms for GI rendering. These methods exploit the fact that an MC-based GI renderer oftentimes generates data at many points in a scene for computing the energy along a specific path, but then dismisses this data completely. While there are approaches that reuse computed paths, which are further described in Chapter 2, data reuse is ensured by employing appropriate caching methods. The complexity and the way that various illumination effects are handled is completely dependent on the specific method,

though, and a variety of approaches are available both for offline rendering and interactive visualization.

Additionally, numerous methods have been introduced for filtering noisy image data. With recent approaches also focusing on temporal stability, their usefulness in interactive systems has increased over the years.

In this thesis, all of the three major fields described above are discussed and covered with appropriate methods for improving the general applicability of ray-based rendering methods.

1.2 Aims and Objectives

The aim of this thesis is to develop methods that support the applicability of ray-based rendering in different environments:

- (1) Previews of photorealistic rendering on large, high-resolution display systems
- (2) Interactive ray-based rendering in VR
- (3) Interactive preview rendering of MC-based GI with the possibility of free exploration, independent of the display system

In order to compute previews of photorealistic rendering on large display systems, we are going to analyse methods of selectively focusing on specific image regions that are essential to the user; this approach is mostly tailored towards design review or iterative modeling processes, where the user alternates between constructing a 3D model and reviewing it in its final environment.

As for interactive ray-based rendering on HMDs, we are going to use an eye tracking device which is integrated into the HMD to implement an appropriate method for adapting image resolution to the user's gaze. It is important for this method to provide a sufficient frame rate, while also maintaining a high resolution in order to avoid disturbances.

For interactive preview rendering of GI based on MC simulation, the basic idea is to develop a caching mechanism that provides the means for very fast reconstruction. One of this method's main goals is to also support non-diffuse materials, which is challenging to do in real-time applications. Available caching methods have to be reviewed to identify central aspects to improve upon.

1.3 Thesis Overview

Ray-based rendering methods, which form the foundation of the research presented in this thesis, are described together with their underlying concepts in Section 2.1. More precisely, we describe ray casting, ray tracing, path tracing, bidirectional path tracing and Metropolis light transport, and provide an overview of further developments in the field, as many of the general techniques in ray-based rendering can be used in conjunction with our own contributions.

Photon mapping and radiosity are two of the most important general techniques in GI rendering, that happen to be based on caching mechanisms, making them strongly related to our own work, which is why we give a relatively detailed overview of the research regarding these methods. General optimization of the sampling process in MC methods, more recent approaches based on machine learning and other techniques are also covered in Section 2.2.

In Section 2.3, we provide a brief overview of the most important properties of the human visual system (HVS) that have to be considered when designing a gaze-contingent rendering method. We also give a basic description of perceptual methods for quality assessment and an overview of research from the fields of foveated and gaze-contingent rendering. Denoising methods, which have become an invaluable tool in GI rendering, are reviewed in Section 2.4, while caching and reprojection methods are covered in Section 2.5.

In Chapter 3, we introduce our work on guided high-quality rendering, working towards the goal of supporting an iterative modeling/reviewing process, including GI rendering on high-resolution display systems by providing the means for focusing the computational effort on specific regions according to the user's requirements. Parts of the work presented in this chapter have been published in (Roth, Weier, et al. 2015).

Chapter 4 contains a thorough description of our foveated ray tracing system for HMDs, as well as an analysis of the outcome of our user study. In addition to letting the participants of the study rate the image quality achieved by our approach, we subsequently analysed the recorded eye tracking data together with the shown scenes' properties to give an idea of gaze behaviour in foveated rendering systems. The work presented in this chapter has been published in multiple articles, namely (Weier, Roth, et al. 2016), (Roth, Weier, et al. 2016), and (Roth, Weier, et al. 2017).

Our hash-based caching system for GI rendering is described in Chapter 5, together with a novel layered filtering framework and an exemplary cross-bilateral filtering approach suited to the specific kind of visual artifacts exhibited by our approach. This work has been published in (Roth, Weier, et al. 2019) and (Roth, Weier, et al.

2020).

We integrate and discuss the presented ideas, give an overview of our contributions in this thesis and provide an outlook towards possible future research in Chapter 6.

Chapter 2

Background and Literature Review

In this chapter, we describe the foundations of our work and analyse the available research in the relevant fields. First of all, we introduce the fundamental methods of ray-based rendering in order to provide some basic understanding of the involved computations and challenges. After evaluating the relevant research regarding the concepts of ray-based rendering, we take a closer look at the human visual system (HVS) and the according properties that are important when synthesizing images. On the one hand, there are properties that may lead to adjustments of the according rendering system. On the other hand, there are properties that could also be described as *flaws* or *imperfections* that can be exploited to accelerate rendering or focus on the most important aspects of the generated images' appearance. This also leads to post-processing of the generated data, more specifically, denoising methods, that try to remove artifacts from the imagery that result from stochastic rendering approaches or general aliasing phenomena. Eventually, we give an overview of direction-independent and direction-dependent caching methods in global illumination (GI) rendering.

2.1 Concepts of Ray-based Rendering

There are various different approaches for image synthesis in the field of computer graphics. Until recently, rasterization has been the most widespread approach when it comes to synthesizing images from scene geometry, especially in the consumer field. One of the reasons for this is that rasterization has been supported by widely available hardware since the late 1990s, while ray-based techniques were largely bound to software implementations. With the appearance of freely programmable GPUs in the late 2000s, ray-based rendering started to gain momentum, with hardware support now being available in current-generation graphics hardware.

In this section, we first introduce the most basic ray-based rendering techniques: ray casting, which supports only local illumination, Whitted-style ray tracing, which ad-

ditionally supports specular GI, and path tracing, which additionally supports glossy and diffuse GI. Subsequently, several more sophisticated methods and approaches that have been developed over the years are described. As our own work presented in this thesis deals with surface-based data exclusively, we mostly omit research that is oriented towards participating media and volume rendering.

2.1.1 Ray Casting

The most basic ray-based rendering technique is *ray casting*, a term that has been coined in the early 1980s by Roth (1982). In this early work, ray casting was used as the basis for a modeling system for constructive solid geometry. The author describes the concept of generating primary rays (rays that originate at the virtual camera position and intersect the image plane at the according pixel coordinates) and intersecting them with the scene geometry, which forms the foundation of all subsequent research in the field. It has to be noted that even though Roth (1982) was the first to use the actual term *ray casting*, it was Appel (1968) who first described the algorithm already 14 years earlier in his 1968 paper.

2.1.2 Ray Tracing

The first logical extension of ray casting was introduced by Whitted (1980), later forming the – nowadays ubiquitous – term *ray tracing*. More specifically, this kind of ray tracing is referred to as *Whitted-style ray tracing*. The main difference between ray casting and ray tracing is the fact that additional rays may be cast from intersections with the scene geometry in order to achieve realistic physical effects like refraction, reflection, transmission or shadows. Figure 2.1 illustrates the main difference between these two concepts. It is important to note that this basic kind of ray tracing only supports perfectly specular reflections and refractions. A physically correct visualization of diffuse and glossy objects requires more sophisticated methods. The effect of varying recursion depths in a full GI renderer is shown in Figure 2.2.

2.1.3 Path Tracing

The most basic approach to provide a physically-based visualization of diffuse and glossy GI is to apply another extension to ray tracing, namely *path tracing*. For path tracing, multiple paths of rays are traced per pixel. If a ray intersects an object that has been assigned a non-specular material, such as a lambertian (perfectly diffuse) material, a stochastic process decides which direction is chosen for the newly cast ray.

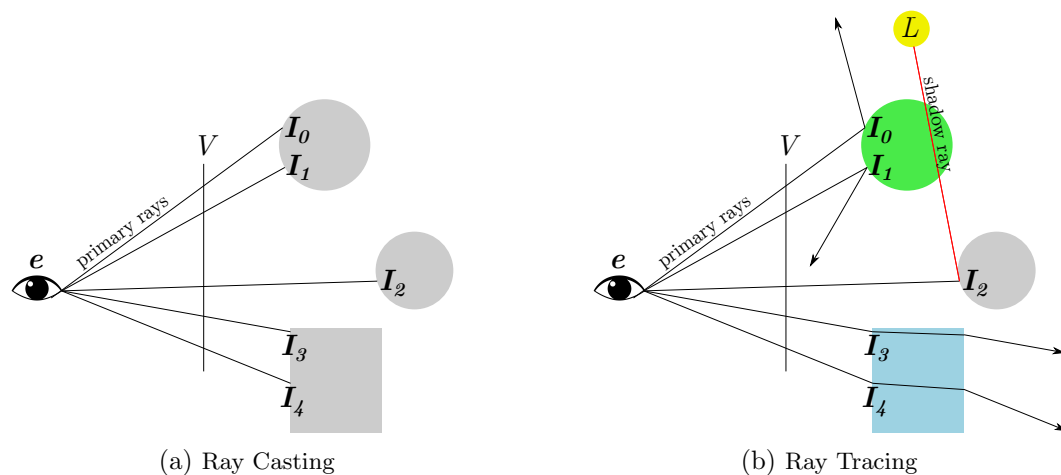


Figure 2.1: A comparison of ray casting and ray tracing. In ray casting, rays are cast into the scene which are then intersected with the scene geometry in order to find the closest hitpoint. Color information or local shading may be used to color the surfaces in the scene. In ray tracing, real specular transmissions/refractions and reflections are possible by casting additional rays according to the local geometry and material information. The green object represents a specular surface, where rays are reflected symmetrically to the local surface normal. The blue object represents a transparent material: rays are refracted according to the material's refractive index and the incident angle when entering and exiting the geometric object. The grey material represents a diffuse surface. At the intersection point I_2 , a shadow ray is cast towards the light source L in order to determine if the surface point is within a shadowed area or if it is illuminated by the light source. In this case, the ray is blocked by the green object, so there is no direct path towards the light source. Note that this kind of shadow computation is limited to blocker objects that are either opaque or have a refractive index of 1, as basic ray tracing cannot handle shadows from transmissive objects that refract light.

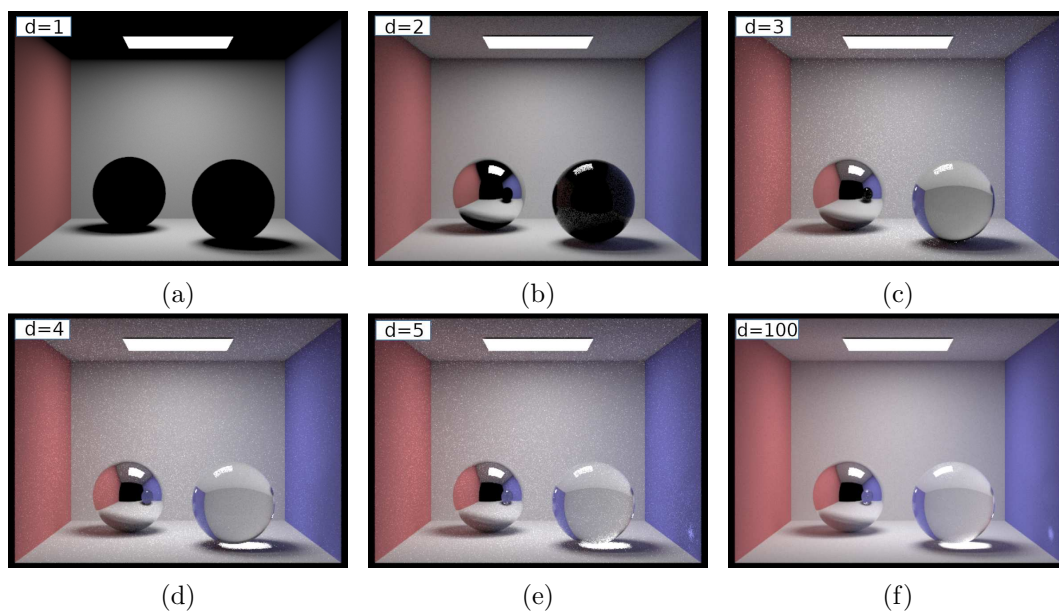


Figure 2.2: The effect of limiting recursion depth d with transparent and reflective materials. For $d = 1$, no reflections or transparencies are visible and the according objects appear black instead. Walls are only illuminated directly from the light source. For $d = 2$, indirect illumination becomes visible on diffuse materials (walls) and reflections become visible, while transmission is still not apparent because rays have to intersect the transparent sphere two times. For $d = 3$, indirect illumination becomes visible in the reflection of the left sphere, while the transparency of the right sphere can also be observed. For $d = 4$, caustics appear for the right sphere, which are also reflected in the sphere for $d = 5$ so they appear on the right wall. The full global illumination is shown for $d = 100$, also rendered with a higher number of samples. Images courtesy of (Zhang 2016)

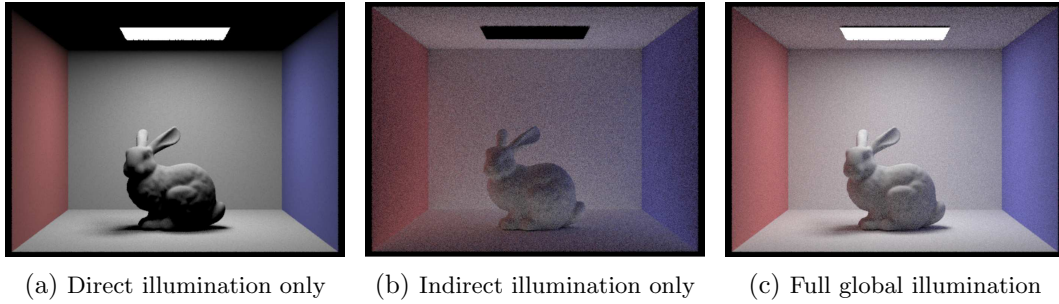


Figure 2.3: Direct, indirect, and full global illumination for the Stanford bunny in a Cornell Box. Images courtesy of (Zhang 2016)

This process is repeated recursively until the maximum recursion depth is reached, a light source is hit or the ray does not intersect any scene geometry (which means that it hits the background – which may also serve as a light source – at infinite distance). The light that is emitted from light sources is then propagated by also accounting for the specific material properties along the path. Repeating this process progressively leads to an approximation of Kajiya (1986)’s *rendering equation* that was presented in:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} f_r(\mathbf{x}, \omega_i)(\omega_i \cdot \vec{n})d\omega_i. \quad (2.1)$$

Here, L_o is the outgoing radiance transported along the direction ω_o from a position \mathbf{x} . L_o is the sum of the emitted radiance L_e and the radiance that is integrated over the hemisphere Ω and then reflected along ω_o according to a local bidirectional reflectance function f_r , where the latter determines the amount of radiance that is reflected towards ω_o when the incident direction is ω_i . While it is possible to also account for wavelength and time in the rendering equation, these parameters are usually omitted unless they are important for the specific method description. Figure 2.3 shows the effect of also employing indirect illumination in the rendering process, but also exhibits a pretty significant amount of noise that results from the stochastic rendering process.

The process of path tracing that has been shortly described above is based on the underlying concept of Monte Carlo (MC) simulation. As the rendering equation generally does not have an analytic solution, it is necessary to leverage numerical integration methods. However, basic approaches that are used for low-dimensional problems, such as Gaussian quadrature, have a very bad convergence rate for high-dimensional, discontinuous integrals such as the rendering equation. The stochastic approach taken by MC methods helps in evaluating integrals without being dependent on the integrand’s dimensionality. While a basic MC approach is usually relatively simple to implement, the error rate of $\mathcal{O}(n^{-1/2})$ may lead to long rendering times until the stochastic noise disappears from an image (Pharr and Humphreys



Figure 2.4: Bidirectional path tracing (left) vs. standard path tracing (right), equal-time comparison. BDPT clearly shows a better overall convergence of the image, with the issues in standard path tracing mainly caused by the specular object on the table. Caustics converge much quicker with BDPT, while a large amount of fireflies and noticeable noise are present when using standard path tracing in such a scenario. Images courtesy of (Veach and Guibas 1995b)

2010, pp. 637–638).

2.1.4 Bidirectional Path Tracing

bidirectional path tracing (BDPT) has been introduced by LaFortune (1993) as well as Veach and Guibas (1995a). In his PhD thesis, Veach (1997, pp. 219–247) gives additional information on BDPT and also describes it within the newly developed *path integral framework* thoroughly. Bidirectional path tracing is an approach that fully adapts the idea of sampling from more than one probability distribution to the path tracing process. Specifically, its basic principle is to start paths not only at the camera, but also at emitting surfaces (light sources), and combine these paths by joining them together. Varying the number of vertices per each of these subpaths makes it possible to obtain a number of sampling techniques. Combining the samples from the various techniques is then done using the multiple importance sampling approach explained below. One of the main advantages of bidirectional path tracing is its ability to handle phenomena such as caustics – which are difficult for standard path tracing – pretty well. Figure 2.4 shows an example of a scene that is difficult for standard path tracing, but can be handled well by BDPT.



Figure 2.5: Bidirectional path tracing with 40 samples per pixel (SPP) (left) vs. Metropolis light transport with 250 mutations per pixel (right), equal-time comparison. The placement of the light source in this scene poses a challenge for path tracing, as light only comes through the slightly open door (according to the author, around 0.1% of the light from the adjacent room can enter through the door). The clear improvements in the MLT-based rendering mainly come from MLT’s ability to maintain path segments that connect the two rooms through the gap, while BDPT generates completely new subpaths with each iteration. Images courtesy of (Veach 1997)

2.1.5 Metropolis Light Transport

Another method for solving the rendering equation is Metropolis light transport (MLT), proposed by Veach and Guibas (1997). By combining the Metropolis sampling method (Metropolis, Rosenbluth, et al. 1953) with the path integral framework, an importance algorithm is derived for path space. As MLT is based on mutations of generated paths, its key advantage is the possibility of local path space exploration by carefully adjusting the probability of mutation acceptance. In addition, the cost for computing a full sample is relatively small: Performing a mutation by adding a vertex to a path, for example, leads to only two additional rays for a full new sample. When important paths are found, steering the mutation probabilities makes it possible to also explore the nearby paths, which is fundamentally important for the efficient visualization of effects like caustics. While the presence of a mutation strategy may remind one of genetic algorithms, it is important to note that – among other differences – genetic algorithms rather deal with optimization problems, while Metropolis methods deal with a sampling problem, as stated by Veach (1997, p. 337). Figure 2.5 shows a scene that is difficult for BDPT, but can be rendered efficiently in the same time when using MLT.

2.1.6 Standard Extensions of Ray-based Rendering

The rendering systems employed in this thesis follow the theoretic foundations of basic ray tracing (Chapter 4) and straightforward path tracing (Chapters 3 and 5), but also include some optimizations such as next event estimation, multiple importance

sampling, russian roulette or low discrepancy sequences for improved convergence behaviour. We give an overview of some of the most popular modifications and implementation techniques for variance reduction in ray-based rendering below to clearly show what the actual renderers used in this thesis are capable of.

Russian Roulette

Russian roulette is a probabilistic approach for limiting recursion depth based on the current throughput of a transport path. This essentially avoids the choice of a fixed maximum recursion depth for an image (which also leads to biased results) and instead moves the issue to a local decision made per sample, based on the potential energy contribution of the currently reached recursion depth. However, choosing such an individual depth limit per path will still lead to a biased rendering result, as the energy that would have been contributed by the additional vertices of the path is essentially lost. In order to create an unbiased result, russian roulette works as follows: Given a probability p for continuing a path, if the decision is positive, the next contribution will be multiplied with the inverse of the probability p^{-1} . With the number of samples going towards infinity, the result is a correctly weighted contribution of later vertices of a path even though they may only rarely be sampled at all.

Next Event Estimation

Next event estimation is a method that explicitly samples one or more light sources at each vertex along a path. This effectively creates a multiple of the paths that would be generated without next event estimation. It does so by only adding a single ray per hitpoint and light source, as it removes the necessity to wait for the original stochastic process to randomly hit a light source at some point, and rather creates explicit paths potentially ending at a light source.

(Multiple) Importance Sampling

Importance sampling is a technique for reducing variance in the rendered image. It relies on the fact that sampling the surrounding of a point on a surface with a specific material may be largely improved by distributing the samples in a way that is similar to the surface material's reflection distribution. While simply relying on the surface distribution works better the closer a material is to perfect specularity, it becomes inefficient when a small light source is sampled with a distribution similar to perfectly diffuse (lambertian) reflectance distribution. To improve this situation,

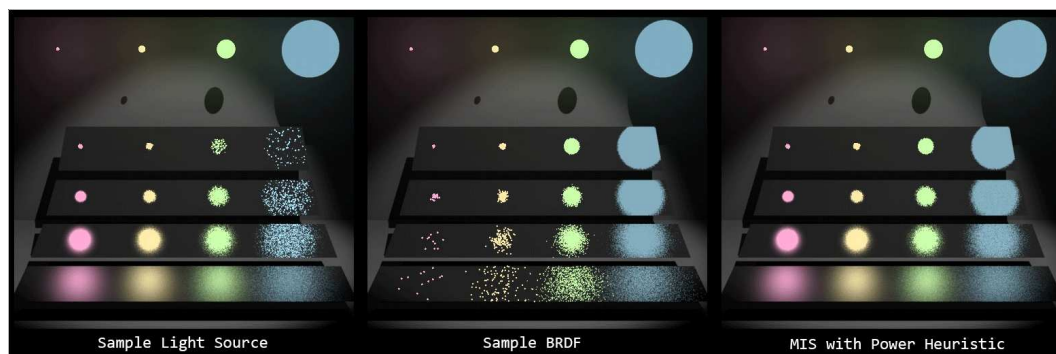


Figure 2.6: Rendering a combination of glossy surfaces and spherical area light sources at varying scales. **(left)** Sampling directions are chosen by uniformly sampling from the directions subtended by the area light. **(center)** The direction is chosen purely based on the local material properties. **(right)** A combination of both methods, weighted using the power heuristic. Images courtesy of (Veach and Guibas 1995b)

importance sampling can be extended to multiple importance sampling. This allows for making sampling decision based on two probability distributions. For example, in the next event estimation process, an additional random decision is introduced that chooses between generating a point directly on the light source based on its emittance distribution and connecting the current vertex to that point, and generating a ray based purely on the current surface’s reflection distribution. In order to create an overall low-variance estimator, these sampling strategies have to be combined with appropriate methods like the balance, cutoff, power, and maximum heuristics. A thorough description of multiple importance sampling and the possible weighting heuristics are provided by Veach and Guibas (Veach and Guibas 1995b; Veach 1997). Figure 2.6 shows the effect of only using a single probability distribution in various combinations of materials and light sources, and the improvement that is achieved by relying on multiple importance sampling.

2.2 Other Ray-based Methods, Extensions and Optimizations

The rendering systems used as the foundation of our work are relatively straightforward in their implementation. Yet, it is important to have an understanding of other approaches of extending and optimizing ray-based rendering methods in order to estimate the value of the contributions made in this thesis. Also, efficient implementations of the extended methods presented below may in turn also lead to implications for our own techniques, because the methods proposed in this thesis can usually also be used in conjunction with more sophisticated underlying rendering methods. However, an efficient implementation of the advanced rendering methods

on the GPU is often far more difficult than for basic path tracing, which is why much research is still based on the straightforward approach that can be parallelized with relative ease.

In this section, we give an overview of the research regarding some of the most popular methods in global illumination, i.e., photon mapping and radiosity. These are also relevant because they follow a caching scheme – something we introduce ourselves in Chapter 5. Subsequently, we look into research regarding sampling techniques, after which we provide some insight into the latest work regarding machine learning, as it has gained a lot of momentum over the last few years, with its concepts also being applicable for many rendering tasks. An overview of the relevant research that does not fit the mentioned fields specifically, but is still important for ray-based rendering in general, is given in the last subsection.

2.2.1 Photon Mapping

Similar to our HashCache method presented in Chapter 5, photon mapping is a two-pass global illumination method based on the eponymous concept of photon maps, serving as a cache which GI is reconstructed from. It was first presented by Jensen (1996). We dedicate a separate section to this popular technique as the involved photon tracing step could also be used to fill our HashCache system presented in Chapter 5. The basic idea of this technique is to first compute two photon maps by shooting photons from emissive surfaces in a scene and storing these at the intersected surface positions. The two photon maps are used for separate illumination phenomena. More specifically, there is a caustics photon map that requires a high resolution in order to directly visualize caustics, which tend to vary at a high frequency spatially, and a global photon map that is used to store lower frequency illumination. Using distribution ray tracing, the scene is rendered by reconstructing illumination from the photon maps. In addition to visualizing the light transport this way, the information stored in the photon maps can be used to optimize sampling directions (Jensen 1995). Also, the distribution ray tracing step should not require a high recursion depth, as the low-frequency photon map serves as an estimate of the radiance on all surfaces that are not too glossy. To compute this estimate at an intersection point \mathbf{x} , a density estimation is computed by locating the n closest photons to \mathbf{x} and integrating the information carried by these photons into the rendering equation.

Peter and Pietrek (1998) extend photon mapping by a third pass that employs particle tracing to compute a global importance map, which is then used to compute the photon maps more efficiently. This means that the photon density in important image regions (i.e., regions that exhibit high-frequency content) is increased by a

factor of up to 8, which allows for a far better reconstruction of effects like caustics. Suykens and Willems (2000) work on two important issues with the photon mapping approach, which are memory requirements and estimating how many photons are required for the representation of light transport in a specific scene. The authors introduce a local required density criterion that determines whether photons need to be stored at the current intersection point, which reduces memory requirements by lowering the amount of unimportant or superfluous stored photons. Caustic and global photon maps show a reduction in the number of photons between 50 and 80 percent. By providing such an estimate, a combination with Peter's and Pietrek's work may lead to the possibility of determining specific termination criteria while still maintaining high visual quality.

Ma and McCool (2002) propose a hash-based method that allows for computing the local neighborhood of an intersection point in the photon maps. Their technique relies on subdividing a space-filling curve into blocks that contain spatially coherent photons. With the sub-linear access time and the block-based structure, their approach already allows for an efficient implementation on GPUs. However, it has to be noted that the block-wise storage of photons will not deliver the same visuals as querying the original photon map. Instead, this is an approximate nearest neighbor method. However, it could be employed in methods that rely on a density estimation step similar to basic photon mapping.

The order of computations for the final gathering step in photon mapping is essentially reversed in Havran, Herzog, et al. (2005)'s work. The technique employs two trees for storing not only the photon positions, but also the position of final gathering rays. There is a notable performance improvement (speedup between 1.3 and 3.3, depending on the scene) that the authors mainly attribute to the logarithmic search complexity. Image quality is not affected and additional techniques for memory reduction of importance sampling can be combined with the method.

Interestingly, Steinhurst, Coombe, et al. (2005) explain that besides the memory requirements, memory bandwidth will also be a major challenge on the way to a hardware implementation of real-time photon mapping. Their approach to reducing the required bandwidth consists of reordering the nearest neighbor queries and using cache conscious data structures. Using a Hilbert curve for reordering the spatial queries, the total required bandwidth could be reduced by 99.2%, but with a very high amount of required intermediate storage. However, with just 1MB of storage, the required bandwidth could still be reduced by one order of magnitude. Since to our knowledge most methods do not rely on the order of spatial queries, this reordering approach can be considered as being rather universal when it comes to combining it with other optimizations of photon mapping.

Hachisuka, Ogaki, et al. (2008) present a progressive method based on photon map-

ping (therefore called *progressive photon mapping*). Their multi-pass algorithm relies on multiple photon tracing passes in order to progressively improve the accuracy of the GI solution. To achieve this, a new radiance estimate is used that converges to ground truth when adding more photons. In contrast to standard photon mapping, it is not necessary to store the full photon map. Instead, a limited amount of memory is already enough for computing a solution of arbitrary accuracy.

Spencer and Jones (2009) improve the final gathering step, which is fundamental to the photon mapping approach, by evaluating the photons over a surface hierarchically. An irradiance estimate is computed by using the gather rays' footprints rather than relying on local photon density. The presented approach leads to vastly reduced noise, but may exhibit visual differences to the ground truth, such as overly soft shadows. Hachisuka and Jensen (2009) extend progressive photon mapping to efficiently handle distribution effects like depth-of-field. Therefore, *stochastic progressive photon mapping* is introduced, a method which shares photon statistics within an area instead of using a purely point-based approach. The authors demonstrate how the method provides the same robustness as progressive photon mapping, while distribution effects are handled efficiently. Hachisuka and Jensen (2010) bring progressive photon mapping to the GPU, while Chen, Wang, et al. (2011) propose an improvement to stochastic progressive photon mapping by extending it with a Metropolis-Hastings algorithm that – just like it improves the distribution of paths in Metropolis light transport – can improve the distribution of photons by exploiting local coherence. Their approach cuts the relative error for various scenes in half or performs even better. In the most difficult test scene the number of visible light paths computed by the new method is improved to more than 2000 times the amount that is visible when uniform sampling is used (30.9% vs. 0.014%), with the worst case being the Cornell Box scene (92.8% vs. 82.1%).

Knaus and Zwicker (2011) present a new formulation to progressive photon mapping by using a probabilistic approach that does not require storing the local photon statistics and supports arbitrary kernel adjustments for computing the radiance estimate. In addition, volumetric photon mapping is also supported.

Hachisuka, Jarosz, et al. (2012) give an overview of the state-of-the-art in photon density estimation. The presented methods include basic photon mapping, (probabilistic) progressive photon mapping, participating media and some more concrete information on the implementation of photon mapping methods in readily available rendering systems. Based on analyzing the radiance estimate in progressive photon mapping, Kaplanyan and Dachsbacher (2013) introduce a locally adaptive extension that minimizes the error in an image by balancing noise and bias. To achieve this, photon mapping is reformulated as a regression problem, which enables the authors to estimate a pixel's measurement. In the same year, Mara, Luebke, et al. (2013)

performed an analysis of various photon mapping approaches, leading to the conclusion that tiled, deferred photon gathering in a compute shader is the best variant when it comes to performance and quality.

Kang, Wang, et al. (2016) present a survey of the state-of-the-art of photon mapping algorithms and the most important challenges in the field. According to the authors, at the time of writing their survey article, progressive photon mapping was a very important field of research. However, they also mention the drawback of having to adapt the global photon map to the progressive approach in order to visualize it directly.

Driven by the gradient-domain extensions that have been proposed for MC-based rendering methods, Hua, Gruson, et al. (2017) improved density-estimation-based approaches like photon mapping. In the following year, Gruson, Hua, et al. (2018) also included volumetric effects such as participating media in this adaptation.

An approach that uses the recently introduced hardware ray tracing capabilities for rendering caustics with photon mapping in image space is presented by Kim (2019). A major drawback of this method is that it does not perform well when the camera is placed close to a surface that exhibits caustics, as the required photon density in such a case would be way higher than currently manageable. Zhu, Xu, et al. (2020) present a novel photon mapping algorithm that is based on deep learning. The authors train a deep neural network in order to give good estimates on photon density by extracting local photon properties and using these as an input to their network. Their evaluation shows that the proposed method can generate accurate results of scenes that include caustics with one to two orders of magnitude less photons than required by stochastic or adaptive progressive photon mapping. By separating caustics from the global photon map and providing additional features, Zeng, Wang, et al. (2020) achieve very promising results from denoising renderings generated with stochastic progressive photon mapping.

It can be certainly said that photon mapping has come a long way since its original publication, and it still is one of the most efficient tools for rendering accurate caustics. Current developments in the field of image processing and machine learning may have a significant influence on the development of image quality in photon mapping approaches, as various modern denoising methods have already vastly improved the image quality of MC-based methods like path tracing. However, high-frequency effects like caustics remain challenging for such denoising methods, since guessing the photon distribution in such complicated effects with their very specific, scene-dependent appearance is rather challenging.

2.2.2 Radiosity

Radiosity is the first popular algorithm for computing the interreflection between diffuse surfaces (diffuse global illumination), presented by Goral, Torrance, et al. (1984). Like photon mapping, it can be seen as a caching method, but only for diffuse global illumination. It is derived from methods used in thermal engineering and allows for an observer-independent precomputation of light transport. The original article on Radiosity published by Goral et al. is also the origin of the infamous Cornell Box, as it is used here to compare the computed light transport against reality (i.e., a physically constructed Cornell Box).

The hemi-cube method (Cohen and Greenberg 1985) is essentially an optimized approach for determining the form-factors between surface patches. These form-factors are necessary in order to determine the energy exchanged between surfaces in the radiosity method. By projecting other patches onto each patch's hemi-cube, a matrix of form-factors for all patches in the scene is computed, which is then used to compute a radiosity solution.

A more general radiosity method that is not limited to diffuse interreflection is introduced by Immel, Cohen, et al. (1986). In addition to the diffuse intensities, the direction-dependent intensities are computed and stored for each non-diffuse surface patch. Essentially, this can be viewed as a kind of surface light field computation. When rendering the scene, directional intensities of the global hemi-cubes are interpolated bilinearly to obtain a smooth result. Obviously, storing the directional intensities per patch results in prohibitively high memory requirements for real-world scenes, especially at the time of publication. This becomes even worse with the glossiness of utilized materials approaching perfect specularly.

One possible solution for alleviating the long preprocessing time of radiosity methods is to modify the original approach by adding a progressive refinement approach. The first implementation of such a method is presented in Cohen, Chen, et al. (1988). This can be considered one of the most important publications when it comes to bringing global illumination methods to interactive applications. The authors restructure the radiosity algorithm in order to perform the computation of form-factors on-the-fly instead of requiring a full $\mathcal{O}(n^2)$ preprocessing step. They update the radiosity for the entire set of patches that are available all at once, while the patches are processed in an order sorted by their energy contribution. To provide an acceptable visual result early on, global illumination of a scene is estimated by its geometry and materials and then used as an ambient term initially. While Cohen, Chen, et al. (1988) still relied upon the hemi-cube method, Wallace, Elmquist, et al. (1989) presented an alternative method to form-factor computation that relies on a ray tracing approach. Around the same time, Baum, Rushmeier, et al. (1989)

proposed an analytical method for form-factor computation that avoids the visual errors that could result from applying the hemi-cube approach in a progressively refining algorithm. Their hybrid approach employs the hemi-cube as long as the underlying assumptions regarding proximity, visibility and aliasing are not violated, and switches to the analytical method when violation occurs.

Heckbert (1990) separates material properties into diffuse and specular reflectivity in order to compute the specular component with a combination of light tracing and path tracing, and the diffuse component with a radiosity method. Separating the illumination phenomena resembles some properties of the approach we take in our implementation of the HashCache method described in Chapter 5. In this article, Heckbert (1990) also introduces his path notation based on regular expressions.

Hanrahan, Salzman, et al. (1991) propose *hierarchical radiosity*, a method to represent the form-factor matrix in a hierarchical way, done by an adaptive, user-guided subdivision of patches based on an error-bound. Their tests show a reduction of the number of surface interactions by up to two orders of magnitude, while maintaining the precision of the resulting image. Lischinski, Tampieri, et al. (1992) present a method that explicitly considers discontinuities in scene illumination to interpolate between the respective regions. Combined with a progressive radiosity framework, their algorithm makes it possible to accurately render fine details.

In their article on *wavelet radiosity*, Gortler, Schröder, et al. (1993) show that hierarchical radiosity is an actual instance of more general, wavelet-based methods. The proposed approach provides a novel view on the radiosity methods developed so far. Based upon this, a set of new algorithms for computing radiosity is presented that run in $\mathcal{O}(n)$ time. The presented results are based on dubbed flatlets and multi-wavelets and show accurate results. However, the authors emphasize that their technique still requires additional understanding regarding various properties of the employed wavelets and their behaviour in the radiosity context. While hierarchical radiosity is based on subdividing surfaces, it cannot utilize preexisting hierarchies present in the scene geometry, which makes the algorithm inefficient for geometrically complex scenes.

Smits, Arvo, et al. (1994) propose a technique for clustering surfaces by estimating the energy that is transferred between clusters and maintaining an error bound for these interactions, bringing the initial complexity of $\mathcal{O}(n^2)$ down to $\mathcal{O}(n \log n)$ for both computational and storage complexity. The reported speedups show that the performance can be improved by two orders of magnitude for scenes of moderate complexity when compared to the standard hierarchical radiosity algorithm. One year later, Neumann (1995) presented radiosity methods that are nearly linear in runtime complexity. To achieve such a behaviour, the author employ MC methods for solving the system of equations, approximating the gathering and shooting

at a lower computational cost than when using the exact solutions. Additionally, importance sampling is employed throughout the sampling process. Keller (1996) improves this method by using low discrepancy sequences to create the technique of *quasi-monte carlo radiosity*.

A new method for bounding the transfer error of radiosity is presented by Gibson and Hubbard (1996). The authors apply their approach by using volumes as a representation of clustered small surfaces. It is notable that the orientation of the surfaces is still accounted for, which makes it easier to compute an accurate solution when compared to neglecting surface orientation. In the article's conclusion, the authors already make clear that perceptual considerations will be an important measure for determining image quality and termination criteria for progressive algorithms like newer radiosity implementations.

To our knowledge, Keller (1997) was the first to propose an algorithm that could compute a radiosity solution within a few seconds: Instant radiosity. The presented approach avoids the use of finite element methods and scene discretization. Instead, it relies on jittered low discrepancy sequences, the quasi-random walk method and the utilization of the available graphics hardware. In the same year, Gibson and Hubbard (1997) showed how perceptually-based measures relating to brightness perception and adaptation can be used in the computation of radiosity solutions, making it possible to reduce the computational effort that is put into regions of the scene that hardly benefit from it.

Building upon Neumann (1995)'s earlier work, Bekaert, Neumann, et al. (1998) eventually combine MC radiosity and hierarchical radiosity to create the technique of hierarchical MC radiosity. According to the authors, storage requirements are reduced to about 20% of the original hierarchical radiosity algorithm, while the computational effort can be reduced by one order of magnitude.

Durand, Drettakis, et al. (1999) introduce a novel global visibility data structure called the *visibility skeleton* that allows for the computation of exact form-factors between polygons and vertices created during a subdivision process, while also supporting general visibility computations. Discontinuity meshing is extended by using a perceptually-based ranking strategy, leading to a good adaptation of the hierarchical subdivision to shadow variations. At the same time, the square table of form-factors is replaced by a linked structure of polygons and vertices. Scenes that are illuminated by multiple light sources or even almost exclusively by indirect lighting can be handled quite well with the presented method, which is a clear improvement compared to earlier work.

By utilizing the evolving graphics hardware, Coombe, Harris, et al. (2005) developed a radiosity approach that could compute a full radiosity solution for smaller scenes

within a time frame of less than one second, including adaptive scene subdivision. Segovia, Iehl, et al. (2006) propose an extension of Keller (1997)'s instant radiosity similar to the extension that bidirectional path tracing is to regular path tracing. This means that a combination of multiple estimators is employed in order to generate a low variance estimate within a short time frame. Even with multiple light sources and glossy materials, their method could already achieve interactive frame rates on graphics hardware that was commonly available in 2006.

An approach that is based on utilizing hardware shadow maps for computing the indirect illumination of virtual point lights is presented by Laine, Saransaari, et al. (2007). Their method is suitable for a single bounce of GI and generates the virtual point lights by casting rays from the primary light source. They also provide a method that makes the virtual point lights reusable by maintaining their distribution in an incremental way, which leads to only few shadow maps having to be rendered for each frame as long as no rapid lighting changes are present. The result is an algorithm that delivers real-time frame rates even when hundreds of virtual point lights are employed.

The observation of diffuse indirect illumination often exhibiting only low spatial frequencies is leveraged by Nichols, Shopf, et al. (2009)'s image space radiosity approach. By bringing the radiosity computations to image space, readily available hardware support for MipMaps is used for an implicit quadtree implementation. Virtual point lights are computed using a multiresolution splatting technique combined with lightcuts in image space. The presented method is able to generate results at interactive frame rates that are similar to a much larger number of regularly sampled virtual point lights.

Another extension of instant radiosity is presented by Hedman, Karras, et al. (2016). They state their main goal to be the distribution of virtual point lights in a way that enables temporally coherent illumination as well as a high-quality radiosity solution. To achieve this, the authors developed an adaptive distribution of virtual point lights that also maintains temporal coherence. In addition, the number of virtual point lights whose position is changed between frames is limited by a heuristic sampling technique. The presented approach provides the means to compute global illumination for interactive environments with dynamic light sources and complex occlusion properties without having to deal with a large amount of temporal flickering.

Like photon mapping, radiosity has come a long way since its original publication. While it has originally been a purely preprocessing-based method, advancements have made it suitable for interactive scene exploration. Current advancements in GPU development, i.e., the availability of hardware support for ray tracing, will likely influence the use of radiosity-related methods in productive environments and entertainment content such as games.

2.2.3 General Sampling Techniques

Generally, the sampling process in ray-based rendering techniques can be steered or influenced in various ways, which we look into in this section. Already in 1955, Kahn (1955) published his research on variance reduction techniques for MC methods, which were completely unrelated to photorealistic image synthesis, which obviously did not exist back then. These techniques are importance sampling, russian roulette/splitting, combining analytic and probabilistic methods, correlation and regression, systematic sampling and stratified sampling. Looking at rendering and post-processing techniques today, all of these approaches have been used in the synthesis of photorealistic imagery.

Veach and Guibas (1995b) introduce multiple importance sampling as a technique for combining samples from various distributions with an optimized variance (based on several heuristics). This approach basically forms the foundation of bidirectional path tracing, later introduced by the same authors, where multiple distributions are combined all the time to generate new light transport paths. Owen (1998) introduces a new sampling technique called *Latin supercube sampling*. This approach aims for improving the outcome of high-dimensional MC simulations and is built upon the combination of Latin hypercube sampling and Quasi-Monte Carlo (QMC) methods. A grouping of input variables into subsets is used to apply QMC methods for lower dimensions to each of these subsets. This yields an effective QMC approach for high-dimensional problems, which works as well as Latin hypercube sampling in the worst case. Owen (2003) later gives an overview of Quasi-Monte Carlo methods in his 2003 SIGGRAPH course. He starts with basic, unoptimized MC (referred to by the author as *crude* Monte Carlo) as a basis, and then explains what various stratification approaches can do for the integration task. After introducing the terms of uniformity and discrepancy, the road towards Quasi-Monte Carlo is paved step-by-step with the details required for implementing such methods for an individual rendering system.

A multidimensional adaptive sampling strategy is presented by Hachisuka, Jarosz, et al. (2008). They focus on synthesizing images that exhibit distribution effects like soft shadows, motion blur, and depth of field at a high efficiency, i.e., with low noise levels at relatively few samples. Instead of simply adaptively sampling the image plane, the authors show how to adaptively sample the multidimensional space that is given by the rendering equation by putting more computational effort into regions where local contrast is high. The reconstruction of the final image is done by determining each sample's extent in the sampled space, aided by a structure tensor. Looking at the results, the proposed approach performs better than all other methods it is compared to (Mitchell, Metropolis, and low discrepancy sequences) in the exemplary scenes. Note however that the presented approach is tested solely

for ray tracing instead of path tracing, which vastly limits the dimensionality of the sample space.

Jin, Ihm, et al. (2009) propose a technique for a selective and adaptive supersampling process in ray tracing on many-core processors such as GPUs or modern CPUs. The presented approach combines image space attributes such as local contrast and object space attributes such as material properties in order to reduce aliasing artifacts as much as possible. The implementation is done for the GPU as well as the CPU and achieves reported speedups between 2 and 3 when compared to non-adaptive sampling, and an interactive frame rate between 1 and 4 frames per second (FPS) depending on the scene, which can definitely be seen as an achievement back in 2009.

A robust light transport method that unifies photon mapping and bidirectional path tracing by integrating photon mapping into the path integral framework is presented by Georgiev, Krivánek, et al. (2012b). The authors show that their approach retains the advantages of both techniques while also explaining the relative efficiency of photon mapping in specular-diffuse-specular transport paths.

Using the concept of high-dimensional manifolds, Jakob and Marschner (2012) propose a method that can vastly improve the handling of difficult specular and near-specular paths in Markov chain Monte Carlo rendering. Manifolds are used to model the space of specular paths and explore these in a way that is much more efficient than previous methods.

Li, Wu, et al. (2012) apply Stein’s unbiased risk estimator (SURE) to MC rendering in order to generate more effective reconstruction kernels and to treat adaptive sampling and reconstruction in an optimization framework. SURE makes it possible to estimate the reconstruction quality of arbitrary filter kernels, thus eliminating the limitation to symmetric kernels.

Lehtinen, Karras, et al. (2013) introduce a novel rendering algorithm for directly computing image gradients and reconstructing the final image from these gradients, based on Metropolis sampling. The authors show that their approach performs well when compared to state-of-the-art methods and they also present an analysis of the spectral properties of gradient-domain sampling compared to image space sampling.

In their 2015 article, Zwicker, Jarosz, et al. (2015) provide a survey on adaptive sampling and reconstruction methods for MC-based rendering methods. In their report, they distinguish between *a priori* and *a posteriori* methods. *A priori* methods rely on analyzing light transport equations and try to derive appropriate sampling rates or filter kernels, while *a posteriori* methods drive the sampling and reconstruction process by using statistical methods on sample sets.

Kettunen, Manzi, et al. (2015) describe how the estimation of image gradients can also be done for standard MC methods, reducing the error rate significantly in many cases. The technique also undergoes a frequency analysis, comparing it to standard MC rendering. Eventually, *gradient-domain path tracing* is the result of applying the suggested method to a real-world MC method, yielding much better results when compared to basic path tracing. By adding temporal finite differences and extending the reconstruction method to work in a three-dimensional spatio-temporal way, Manzi, Kettunen, et al. (2016) modify gradient-domain path tracing to improve the temporal coherence in rendering animation sequences.

Herholz, Elek, et al. (2016) use the Gaussian mixture model for representing the illumination and the reflectance factors in GI rendering, which they fit by using multiple optimization methods. The resulting product distribution is then used to appropriately parameterize importance sampling instead of only relying on a single one of these factors, as it is usually done. The authors demonstrate that their approach performs better than state-of-the-art methods regarding the error convergence rate in complex environments.

An example for the application of blue-noise is given by Georgiev and Fajardo (2016)'s work on *Blue-noise Dithered Sampling*. Instead of relying on the white-noise error distribution that results from randomly *decorrelating* pixels, the authors employ a dithering-based *correlation* that thresholds individual pixels with a mask based on blue-noise. This minimizes the low-frequency noise in the output signal, leading to a much more pleasant image especially at low sampling rates.

Christensen, Kensler, et al. (2018) present a framework for developing algorithms that generate progressive 2D sample point sequences. Additionally, they propose three such sample sequences, yielding different stratification properties regarding the two dimensions. Taking the prefix of arbitrary length of such a sequence will also yield a well-distributed sequence, which makes them well-suited for progressive rendering. A method for combining multiple MC-based rendering methods to compute a single image is presented by Bitterli and Jarosz (2019), making it possible to use straightforward path tracing as a base algorithm, while artifacts such as fireflies can be dealt with by a *metropolised* sampling scheme. This brings together the best of both worlds: The computational overhead for basic path tracing is low, complicated light transport paths can be sampled, and temporal flickering artifacts that often result from Metropolis-based techniques are eliminated.

2.2.4 Machine Learning

With increasing hardware capabilities and even specific support by GPUs, machine learning has been an important research topic in many fields for several years now,

including image synthesis. Vorba, Karlík, et al. (2014) propose a parametric mixture model which is progressively trained for scattering directions and light source sampling and can be integrated with state-of-the-art rendering methods. The authors modify the offline stepwise expectation maximization to create an online process that can potentially handle an arbitrary amount of samples. They describe how an unbiased guiding method based on density estimation of weighted particles and distribution caching can be used to improve rendering quality significantly. Most of the overhead that is generated by the presented technique is caused by cache queries, which may take almost up to half the full execution time for very simple scenes with basic path tracing.

Using a data structure similar to irradiance volumes, Dahm and Keller (2017) introduce a modified importance sampling approach that significantly improves the performance of light transport simulation. The presented method is based on uniformly sampling the scene and placing a hemisphere at each sampled point that stores the incident radiance for a discrete set of directions, which is then used to guide the importance sampling process when generating the next vertex for a path. Also, these hemispheres are progressively updated during the rendering process. The presented results are clearly better than basic path tracing and – for the shown scene – even improve upon MLT.

Methods to steer importance sampling with neural networks in primary sample space and in the general case have been proposed by Zheng and Zwicker (2018), and Müller, McWilliams, et al. (2019), respectively. Elek, Thomas, et al. (2019) propose an approach that tries to produce the best estimate in MC-based image synthesis on a per-pixel basis by considering the sample distributions for each pixel individually for varying scene configurations. The technique is based on identifying recurring patterns in these distributions and using them to train a model for variance reduction based on neural networks. The resulting estimates are improved when compared to simply computing the mean value of all samples, while the approach can also be combined with standard denoising methods. Sanzenbacher, Mescheder, et al. (2020) show how neural networks can be used to learn light transport for static and dynamic scenes in three spatial dimensions. Contrasting other approaches, their technique allows for predicting global illumination effects and also enables the user to modify the scene geometry interactively. Tewari, Fried, et al. (2020) present a survey on state-of-the-art methods that leverage machine learning methods to generate photorealistic imagery.

With increasing computational power and the available hardware support for machine learning techniques, a machine-learning-based approach to global illumination and, more specifically, its underlying sampling techniques, may certainly yield outcomes that would be hard to achieve with other approaches that are not based on

machine learning. However, it is very important to keep the possible challenges of such techniques in mind, such as the appropriate input selection or overfitting issues.

2.2.5 Other Methods and Aspects

While the methods described in this subsection do not specifically fit into any of the other subsections, they all have the common goal of accelerating global illumination rendering.

Stamminger, Scheel, et al. (2000) present suitable algorithms for interactive walk-throughs through global illumination solutions that also exhibit distribution effects like glossy materials. To achieve this, a novel representation of incident illumination is presented, using a hierarchical radiance clustering algorithm to achieve a finite element representation of outgoing radiance. At the same time, outgoing radiance is represented using an adaptive hierarchical basis that only stores radiant intensity instead of radiance and enables interactive viewing. The presented methods can be easily implemented as an extension to radiosity systems that only support diffuse GI. Instead of also representing incoming radiance using a costly finite-element representation, the authors present an algorithm they call *illumination samples* as the means to store non-diffuse illumination data. The authors describe the illumination samples method to be similar to photons in Jensen's photon mapping approach (Jensen 1996).

Instead of working with concrete illumination samples, the light transport within a scene can also be analyzed in a preprocessing step. Sloan, Kautz, et al. (2002) propose such a method for storing precomputed radiance transfer that achieves high frame rates when rendering low-frequency global illumination for diffuse and glossy objects and can even handle soft shadows, interreflections and caustics. Their main contribution is the concept of functions that are distributed over objects' surfaces and represent the transfer of incident lighting into transferred radiance. Rigid rotation of objects is possible, and an extension even allows for rigid movement. The evaluation shows that performance for the shown scenes is clearly interactive (between 1.7 and 129 FPS), with a preprocessing time between 8 minutes and 4.4 hours. While such a preprocessing time is too long to use the proposed method in an iterative workflow where previews of a model are required after slight changes of a 3D model, it is certainly adequate when the modeling process is finished and geometry and materials are finished.

Bala, Walter, et al. (2003) present a new representation of sparsely sampled global illumination, the *edge-and-point-image*. Their approach does not rely on any preprocessing or storage of the computed illumination of preceding frames, but instead works purely in image space in order to reconstruct a visually pleasant image from

a sparsely sampled one. Using appropriate interpolation methods, radiance can be reconstructed instead of computing complex shading per-pixel. The edges represented in the edge-and-point image are analytically computed and serve to avoid wrongly interpolated samples across discontinuities. Even soft shadows are suitably reconstructed with the border between umbra, penumbra and unshadowed regions being computed analytically. The presented system allows for interactive object manipulation and delivers a speedup of 20–60 over basic ray tracing while exhibiting a lower amount of aliasing. One of the main advantages of a system like Bala’s is its independence of temporal changes in a scene. In contrast to methods that reuse information from previous frames or knowledge about the radiance transfer in a scene, the method at hand can also be used in fully dynamic scenes.

A much more basic and universal approach is the development of advanced strategies for data structure construction that lead to improved traversal times. Improvements to the bounding interval hierarchy (BIH) that allow for better adaptation to non-uniformly tessellated geometry are presented by Stich, Friedrich, et al. (2009), leading to the popular *SBVH* data structure that is also used in NVIDIA’s OptiX framework. Also, this is the data structure we chose for all rendering components developed in this thesis, as it combines low construction times with a high traversal performance. As an alternative to pure offline computation of GI effects, Crassin, Neyret, et al. (2011) propose voxel cone tracing, a method that uses a sparse voxel octree and allows for real-time computation of glossy and diffuse GI. Their approach makes it possible to estimate the light transport along a bundle of rays within a cone by tracing only a single ray, which is a significant alleviation of the required computational effort. With two indirect light bounces, the presented method already achieves 25–70 FPS on an NVIDIA GeForce GTX 480. One drawback of the presented method is the necessity of maintaining the scene’s voxel representation, which may limit the scene size due to both computational and memory limitations.

Müller, Gross, et al. (2017) developed a method for light path guiding in standard path tracing, based on learning an approximate representation of a scene’s radiance field. The authors introduce a novel data structure for the according spatio-directional information, called the SD-tree, that serves to store and sample the incident radiance and consists of two parts: A binary tree for spatial subdivision that forms the upper nodes and a quadtree for directional information that forms the lower nodes. It is emphasized that the proposed method does not require any parameter tuning and yields results that are at least on-par with more complicated state-of-the-art methods, tested with difficult visibility situations, highly-detailed models and complex light transport. A light path guiding method like this could also be greatly interesting for building an optimized renderer for the methods contributed in this thesis.

A novel method for using sparse samples of the diffuse light field to interpolate radiance is presented by Silvennoinen and Lehtinen (2017). Their work essentially consists of a direct-to-indirect transport method that allows for accurately rendering imagery exhibiting diffuse indirect illumination effects. Because of a necessary preprocessing step, the proposed algorithm works with mostly static scenes, but allows for fully dynamic lighting, camera and (diffuse) material changes. Radiance is reconstructed by using a set of radiance probes that are distributed throughout the scene with global and local parts being dealt with independently. The sampled radiance field is used as an input to the local reconstruction operators, which yield the indirect radiance estimate. Radiance probe placement can be done with different approaches; a non-uniform way of this placement process is described by Wang, Khat, et al. (2019).

Tailored specifically towards open world scenes, Liu, Gao, et al. (2020) present an approximate GI method based on hybrid cone tracing that works with light maps for complex meshes and height-field meshes separately. The algorithm consists of two steps: First, the cone-scene-intersection is performed, followed by light sampling and sample accumulation. According to the authors, the results are visually pleasing, while memory requirements are around 12% of similar state-of-the-art methods like voxel cone tracing.

2.3 Perceptual Considerations and Gaze-Contingent Rendering

2.3.1 The Human Visual System: Basic Concepts and Limitations

Over the years, there have been various great general reference works in the field of human visual perception and the HVS. Hubel (1988)'s *Eye, Brain, and Vision*, Wandell (1995)'s *Foundations of Vision*, Adler, Kaufman, et al. (2011)'s *Adler's Physiology of the Eye*, and Goldstein (2013)'s *Sensation and Perception* form a great basis for the general understanding of human vision and perception.

There are some concepts of the HVS that are required to fully understand parts of the main chapters of this thesis, in which specific properties and weaknesses of human vision are exploited in order to accelerate and improve the process of image synthesis. To make it easier to understand these parts, we give an overview of basic mechanisms and properties that are important for the following chapters, based on the high-level model of the HVS that we developed in our state-of-the-art report (Weier, Stengel, et al. 2017), shown in Figure 2.7. This model contains the main stages that have to be passed by a visual stimulus before it is finally turned into

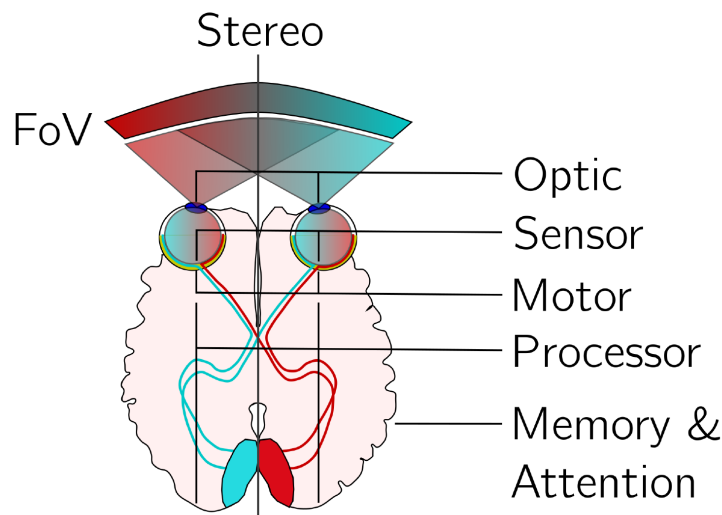


Figure 2.7: A high-level model of the HVS that describes the basic components responsible for visual perception.

a percept. Following the illustration of our high-level model from top to bottom, there are two locations (one for each eyeball) where light enters the HVS, which enables stereo vision. After passing the optical system, light reaches the retina (called the *sensor* in our model), which is responsible for passing a stimulus to the visual pathways. Via this connection, signals are passed to the visual cortex, that involves various parts of the brain responsible for processing and interpreting the signals and forming a percept, and thus an image perceived by the individual. Memory and attention also play an important role in this process. Below, we will shortly describe the most important properties of these elements in order to convey the knowledge necessary for understanding some key elements of the main parts of this thesis. For a more detailed description, see (Weier, Stengel, et al. 2017) and the reference works described at the beginning of this section.

There are various optical properties of the HVS that partially result from the position and shape of the eyes. With their eyes looking straight ahead, humans have a horizontal field of view (FoV) of approximately 190° , which increases to 290° when also accounting for eyeball rotation. It is important to note that perception is far from uniform over the whole visual field because of the spatial varying properties of the retina, which means that there are significant differences in perception between central and peripheral vision. Also, the resolution of the HVS is limited to approximately 60 cycles per degree (CPD) (Wandell 1995, p. 24), which could lead to aliasing when viewing specific patterns that contain higher frequencies. However, the optics in the cornea and lens also serve as a lowpass filter that has a cutoff frequency of approximately 60 CPD, which reduces aliasing artifacts significantly. Another important element among the optical elements is the pupil, which serves as an aperture. Mainly, it is responsible for adjusting the sharpness of the perceived

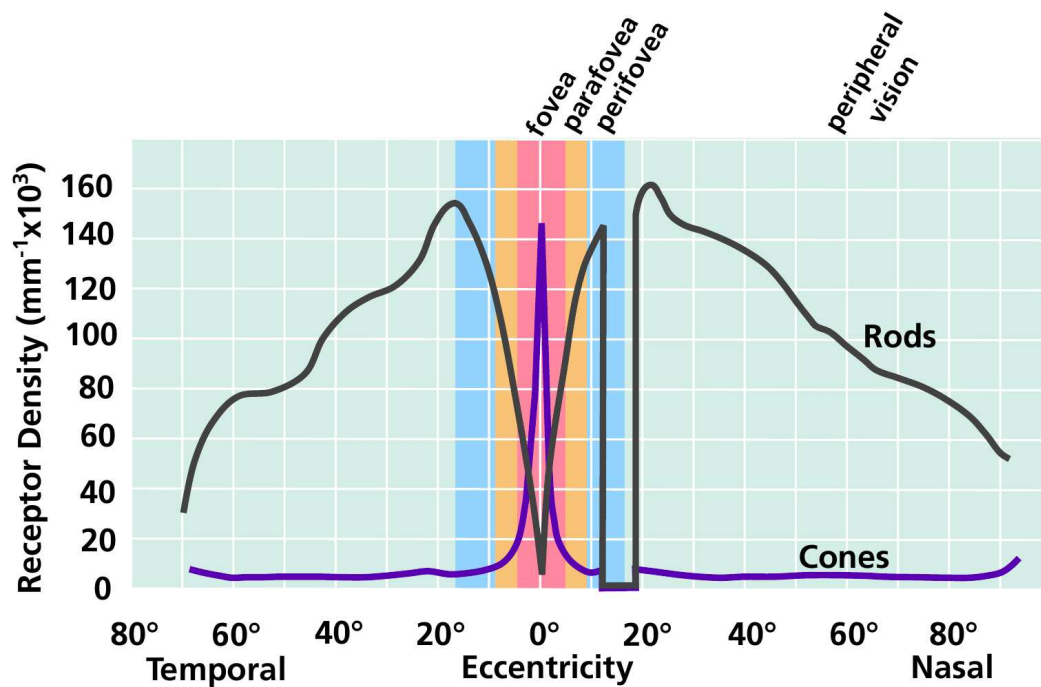


Figure 2.8: Photoreceptor distribution on the retina

image with its adjustable diameter (2–8mm), as it can only control about one order of magnitude of light intensity.

After traveling through the eye's optical system, light reaches the retina, which is the photosensitive layer of the human eye, consisting of two different kinds of photoreceptors, responsible for brightness and color sensation, respectively: Rods and cones. Rods are mainly important for brightness perception in scotopic vision, which occurs when only little light is available. This means that a visual percept in scotopic vision is mainly caused by the rods, leading to a monochromatic perception in very low light. Cones, on the other hand, are divided into S-, M- and L-cones, responsible for short, medium and long wavelengths (often described as blue, green and red). They enable color perception, also referred to as photopic vision.

As mentioned above, photoreceptors are not distributed uniformly over the surface of the retina, which is illustrated in Figure 2.8. The fovea is the retina's central area with a size that corresponds to approximately 5° of the visual field. As shown in the diagram, there are no rods in this area, but it is instead fitted with cones exclusively. At the same time, the density of cones on the retina drops significantly with increasing eccentricities (denoting the angular distance to the optical axis). In addition to the fovea, central vision consists of two more areas: The parafovea (between 5° and 9°) and the perifovea (between 9° and 17°) (Curcio, Sloan, et al. 1990). Areas outside *central vision* belong to *peripheral vision*. The density of cones

is crucial for *visual acuity*, also described as the “keenness of sight”, which is the highest inside the foveal area, while already being reduced by approximately 75% at an eccentricity of 6°. While depending largely on the receptor distribution, visual acuity is also influenced by a stimulus’ contrast. The limit of visual acuity can only be achieved with a high contrast image under photopic vision, such as reading a book in daylight.

Another element of perception that is influenced by the varying distribution of photoreceptors is color sensation. The foveal region mainly contains M- and L-cones, while S-cones are spread out much further, leading to a higher sensitivity to such wavelengths in peripheral vision. Retinal photoreceptors can also adapt to changes in light intensity, which also influences color perception and visual acuity (Ledda, Santos, et al. 2004). The HVS makes it possible to perceive brightness stimuli within a range of seven orders of magnitude, with varying properties such as reduced color sensation in scotopic vision. Just like visual acuity drops towards the periphery, depth perception is also reduced in peripheral vision, mainly because of the reduction of perceivable details and missing overlap between the two eyes’ stimuli.

Besides the adjustments that can be made by the eye’s optics and photoreceptor cells, external muscles also allow for highly precise changes of the eyeball’s orientation, mainly for following an object of interest (OoI) and projecting it onto the fovea in both eyes for highly detailed perception. Also, this allows for dynamic exploration of the environment and quickly switching attention between various objects without having to move one’s head too much. The extraocular muscles are also responsible for adjusting the lens in order to bring the current OoI into focus. A highly detailed psychophysical description of human eye movements is provided by Kowler (2011). Two very important types of movement of the eyeball that happen constantly are saccades and fixations. While saccades denote the motion that happens when rapidly jumping between multiple OoIs, also triggering saccadic suppression (a rapid decline in visual sensitivity), fixations denote the state where the gaze rests on an OoI and visual information is perceived and processed. It is noteworthy that fixations at larger eccentricities are usually not kept for a long time and instead head movement is induced. According to Defense (1999, p. 17), this eccentricity is referred to as the comfortable viewing angle (CVA) and is approximately 15°.

While the above description is by no means comprehensive, it conveys the information necessary for understanding the approaches and decisions made mainly in Chapters 3 and 4. For a more detailed description with concrete numbers and a large body of research related to accelerated rendering in the context of human perception, see our state-of-the-art report (Weier, Stengel, et al. 2017). In the following sections, we will give a condensed overview of the most important work that is related to our own research and the methods that we used.

2.3.2 Perceived Image Quality and Quality Assessment

Judging the visual quality of an image is a difficult task not only because of differences in perception between individuals, but also because of the complex processing in the HVS, or, more precisely, the visual cortex. While a numeric objective measure can easily be established, e.g., by computing the root mean square (RMS) error of individual color channels or luminance values compared to a reference image, this is not a reliable tool when it comes to *perceived* quality. Simply adding some random noise to an image will yield significant differences when using such an approach, while barely altering the image's appearance as a whole; the same would happen when simply adding or subtracting some constant values from each pixel.

Wang, Simoncelli, et al. (2003) and Wang, Bovik, et al. (2004) developed the method of multi-scale structural similarity (MS-SSIM) for obtaining structural similarity information that strongly relies on the assumption of the HVS' high adaptation for the extraction of structural scene information. This means that the ability of obtaining such a measure should yield fundamental information about the perceived image quality. The authors also present experiments that show the superiority of their approach when compared to single-scale methods when parameterized correctly. When it comes to the visual fidelity of natural images, Chandler and Hemami (2007) introduce the *visual signal-to-noise ratio (VSNR)*, which is a two-stage approach. Wavelet-based models of visual masking and summation are used for detecting distortions in natural images using contrast thresholds in the first stage. In the second stage, a multi-scale wavelet decomposition is employed to determine the significance of the present distortion by looking into the visual properties of perceived contrast and global precedence. However, the proposed method is limited to grayscale images, which means that purely chromatic errors in an image remain undetected.

Huynh-Thu and Ghanbari (2008) analyse the behaviour of peak signal-to-noise ratio (PSNR) as a quality measure when different content and video codecs are used and come to the conclusion that if these parameters are changed, then PSNR is not a valid quality measure anymore because of the reduced correlation between perceived quality and PSNR.

Mantiuk, Kim, et al. (2011) present a new visual model that works for all luminance conditions, which they base a novel visual metric for predicting visibility (discrimination) and quality (mean-opinion-score). The employed visual model is derived from new measurements of contrast sensitivity and it is calibrated against several relevant datasets and image quality databases. The evaluation shows the metric to achieve results at least on-par with MS-SSIM. Note that contrary to Chandler and Hemami (2007)'s work, this approach supports chromatic information instead of only working with grayscale images.

Just like many other areas, machine learning has gained interest in the field of image classification. Rawat and Wang (2017) present a comprehensive review on the use of deep convolutional neural networks (DCNNs) in image classification, giving an overview of the history of these learning-based methods and an in-depth analysis of the important work and the challenges that have been identified. A deep neural network (DNN)-based method aimed specifically at quality assessment is proposed by Bosse, Maniry, et al. (2018). The novelty of their approach lies in the adaptability of the DNN they use, as it can be easily modified to work in no-reference as well as full-reference scenarios. There are no hand-crafted features or any other prior knowledge about the involved domains in their system and the presented evaluation results are superior to the state-of-the-art in both no-reference and full-reference modes.

2.3.3 Foveated/Gaze-Contingent Rendering

In this section, we give an overview of work that is relevant for the understanding of our gaze-contingent rendering method and the subsequent analysis of eye tracking data, as presented in Chapter 4. As described in Section 2.3.1, saccades are a very important type of movement in the HVS, and at the same time fixations are only comfortable up to a certain eccentricity. Bahill, Adler, et al. (1975) already described in 1975 their findings about the typical eccentricities of saccades in natural environments, which showed that an eccentricity of 15° is rarely exceeded. Instead, when the focus is shifted between objects that are further apart, head movement is usually involved. Noorlander, Koenderink, et al. (1983) analysed the sensitivity of contrast detection in the peripheral visual field and its relation to the actual size of a colored target. They found that, while color vision deteriorates largely if a stimulus is moved away from the fovea, suitably enlarging the target size leads to a colour discrimination comparable to the foveal area, as more cones are involved.

Details about the ability of motion detection and the relation to object size are discussed by McKee and Nakayama (1984). The authors show that relative motion detection is possible even when the changes are smaller than the retina's spatial resolution. Legge and Kersten (1987) analyse the ability of contrast discrimination in both central and peripheral vision, finding that, if scaled to the local contrast sensitivity, contrast discrimination is similar, which implies similar mechanisms of contrast coding in both areas.

An early gaze-based system for ray-based volume rendering is presented by Levoy and Whitaker (1990). While the authors did not employ eye tracking in their work, the number of rays as well as the number of samples that are computed along each ray are modified based on the eccentricity of the according direction. As such a

sparse sampling method leads to unsampled pixels, an additional reconstruction method is proposed that adapts the kernel size based on eccentricity.

Ohshima, Yamamoto, et al. (1996) provide a technique for gaze-aware LoD rendering, based on information from binocular vision, that allows for interacting with multiple objects in virtual environments. It is noteworthy that, amongst others, their approach relies on the discrimination between the characteristics of central and peripheral vision, which is central to our work presented in Chapter 4. The approach chosen by Murphy and Duchowski (2001) employs eye tracking for LoD-based rendering using a spatial degradation function that the authors derived from a user study. On average, their method delivers a speedup of 4 compared to basic rendering. Interestingly, research has shown that eye movement patterns may vary depending on various specifics like the cultural background of a person, as shown by Chua, Boland, et al. (2005). While the number of participants in a user study like ours limits the possibility of considering this observation, it is certainly worth considering in the sampling process and the evaluation of larger experiments, since experimental results may vary depending on a participant's cultural origin. Also according to Dorr, Martinetz, et al. (2010), eye movements and fixations are significantly more coherent when viewing a Hollywood movie compared to viewing a natural scene, where movements are way more subject-specific. Additionally, the author's found that presenting the same stimulus to one subject multiple times leads to more coherent viewing behaviour than presenting the same stimulus to multiple subjects and comparing their behaviour with each other. However, coherent viewing behaviour for one stimulus or set of stimuli does not imply the predictability of viewing behaviour for another stimulus.

One of the central challenges in the field of human-computer interaction, and even more so in gaze-contingent rendering, is latency. Loschky and Wolverton (2007) carried out a user study regarding the detection of eccentricity-based image blur, finding that in such a scenario update rates below up to 60ms are rarely detected. In the same year, Duchowski (2007)'s comprehensive work on eye tracking methodology was released, as well as an article on techniques for simulating arbitrary visual fields (such as the ones impaired by macular degeneration, glaucoma and similar processes) for video and still images (Duchowski and Çöltekin 2007). Published several years later, Kowler (2011) provides a psychophysical survey on many details of human eye motion, while Strasburger, Rentschler, et al. (2011) published a review/survey taking the connection between the drop in visual acuity and peripheral vision into focus.

A method for foveated rendering based on rasterization is presented by Guenter, Finch, et al. (2012). Based on the peripheral acuity falloff and eye tracking data, the proposed method renders three image layers at different resolutions, where the

highest resolution layer contains only the foveal region, the medium resolution layer contains a transitional region and the lowest resolution layer contains the full image. These three images are smoothly composited to avoid artifacts. While there is some redundancy in the rendering process here, speedups of 5–6 are achieved on a 1080p desktop display. A foveated ray tracing approach similar to our own system proposed in Chapter 4 is presented by Fujita and Harada (2014). While they use a static sampling pattern for reconstruction in conjunction with a nearest-neighbor approach, our own system relies on temporal reprojection, also allowing for temporal antialiasing.

A method for measuring the latency of gaze-contingent display systems directly has been developed by Saunders and Woods (2014), coming to the conclusion that when simulating visual impairments or similar content, latencies should be as low as possible, even leading to the recommendation of using high-speed CRT displays in such cases. An extension to the graphics pipeline that allows for varying sampling densities across the image plane is proposed by He, Gu, et al. (2014), being a natural fit for rasterization-based foveated rendering methods like the one by Guenter, Finch, et al. (2012).

With eye tracking devices in head-mounted displays (HMDs) not being widespread at the time, Stengel, Grogorick, et al. (2015) developed a solution for the integration of a binocular eye tracker into an HMD. In an emerging technologies installation, Patney, Kim, et al. (2016) presented prior work on foveated rendering as well as their own, novel technique (Patney, Salvi, et al. 2016) that reduces objectionable artifacts of previous methods while also still significantly reducing rendering cost. Stengel, Grogorick, et al. (2016) propose a method that focuses on the reduction of shading costs, as shading produces a large portion of the computational cost in today’s rendering systems. The authors approach this by only shading visible features and interpolating the remainder of the image while avoiding visible artifacts. In this process, they account for acuity falloff, eye motion, contrast perception and brightness adaptation. Their technique is integrated into a deferred shading pipeline with the interpolation mechanism being implemented by using a pull-push approach. Psychovisual experiments demonstrate the scene- and task-independence of the proposed approach, which reduces the number of shaded fragments by 50 to 80%. Pohl, Zhang, et al. (2016) present a technique for exploiting lens astigmatism in modern HMDs to improve rendering performance by approximately 20%. A method specifically tailored towards the needs of 3D artists creating 360° content is presented by Koskela, Immonen, et al. (2017). With their foveated rendering approach, the authors improve the workflow by removing much of the computational cost that originates from scene changes and put the main effort on the artist’s points of interest, which is highly related to our work presented in Chapter 3. Sampling is performed according to a newly introduced acuity model. Meng, Du, et al. (2018)

propose a closed-form, parameterized foveation method based on the distribution of photoreceptors in the retina. Their two-pass algorithm is based on rendering to a reduced resolution buffer in log-polar space based on a kernel transformation in the first pass, while the inverse transformation is done in the second pass, also including antialiasing. When rendering to a 4k display (2160p resolution), the authors report speedups around 3 while only losing a negligible amount of detail. Similarly, Koskela, Lotvonen, et al. (2019) also propose to render in visual-polar space before transforming the result to screen space.

2.4 Denoising

Denoising methods have become a central component of non-interactive and interactive photorealistic image synthesis alike, especially within the last decade. In this section, we will describe some of the most important techniques, including those that are based on machine learning, which have mainly emerged over the last five years. Denoising algorithms are also employed in our work presented in Chapters 3 and 5.

Buades, Coll, et al. (2005) provide a framework for the mathematical and methodological comparison of denoising methods as well as a novel filtering approach based on non-local means. With their approach, they try to solve the issue of prior methods that caused artifacts and removed fine details from denoised images when the input did not match the algorithm’s underlying assumptions. The same authors present a unified theory of denoising filters (Buades, Coll, et al. 2008). They discuss several principles that a denoising method should follow in order to achieve desirable results without impacting detail and structure of an image, while still reducing the amount of perceived noise significantly.

With their edge-avoiding, wavelet-based approach, Dammertz, Sewtz, et al. (2010) propose one of the first interactive methods for denoising images generated using MC methods. Shortly after that, Bauszat, Eisemann, et al. (2011) published their method based on *guided image filtering*. The main idea of their work is not to filter the path traced image, but instead use it as a guide to filter images containing noise-free scene characteristics like surface normals or depth. Like many other techniques, their approach is usually only applied to indirect illumination, since lower-frequency content can be handled much better by the presented approach. Sen and Darabi (2012) analyse the functional relationship between MC noise and the chosen random parameters by calculating the statistical dependency between the system’s input and output. Each sample is assigned an importance, which is reduced for samples strongly affected by MC noise. Using this importance as an auxiliary buffer for a cross-bilateral filter, noise can be reduced to levels that equal renderings with three

orders of magnitude more samples. A high-dimensional filtering method based on adaptive manifolds is proposed by Gastal and Oliveira (2012). With their approach of reducing the sample sets and interpolating remaining pixels, they already achieved framerates as high as 50 frames per second for a resolution of 10 megapixels in 2012. Rousselle, Knaus, et al. (2012) developed a technique that does not only denoise the image based on a non-linear filter, but also estimates residual errors on a per-pixel basis and uses this information to adaptively sample the image plane using a dual-buffer approach. In another article, Rousselle, Manzi, et al. (2013) propose a method for appropriately combining the noisy color buffer and noise-free feature buffers in order to efficiently filter noise from an image while avoiding the overblurring of structural details. The suggested framework is based on non-local means and cross-bilateral filtering, with the error estimation relying on the SURE method. Their approach shows significant improvements over preceding work and also includes adaptive sampling as well as space-time filtering. A histogram-based multiscale method for denoising is proposed by Delbracio, Musé, et al. (2014), with an optimized GPU implementation developed by Szeracki, Roth, et al. (2015). However, due to the necessity of building histograms over samples, this approach requires a relatively large amount of samples in order to avoid the appearance of rectangular artifacts in the result, which is prohibitive for interactive applications.

While other filtering approaches described so far work in image space, Keller, Dahm, et al. (2014) propose a method that smoothes the contribution of samples in path space before accumulation in image space. Essentially, our own filtering approach used in Chapter 5 is a hybrid method between path space and screen space, as the individual vertices of each path are filtered and propagated from bounce to bounce. Building upon the successful variety of feature-based denoising filters, Khademi Kalantari, Bako, et al. (2015) use multilayer neural networks to learn the relationship between image noise and ideal filter parameters with a nonlinear regression model. After an offline training phase, the network adjusts the filter parameters per pixel in order to approximate the ground truth as good as possible, also supporting various distribution effects.

Based on an analysis of existing approaches, Bitterli, Rousselle, et al. (2016) use their observations to design a new regression-based filter that only relies on auxiliary buffers in the fitting phase, but disregards those buffers when computing regression weights. In addition, the authors introduce a new general mean squared error estimator that is well-suited for the proposed filtering method, while also automatically parameterizing the regression kernel. Schied, Kaplanyan, et al. (2017) propose a variance-based filtering approach working with temporal accumulation in order to generate temporally stable image sequences. Using a hierarchical image space denoising filter makes it possible to distinguish between fine details and noise at multiple levels. Their method achieves great visual quality already at 1 SPP, as

temporal accumulation effectively increases the available sample count. Using gradient estimation, the proposed approach is further improved (Schied, Peters, et al. 2018).

Chaitanya, Kaplanyan, et al. (2017) improve denoising methods using deep convolutional networks by adding recurrent connections to reduce temporal artifacts. Their approach is applicable to image sequences generated with very low sample budgets, which can be denoised at interactive rates, while also automatically modeling the relationship between auxiliary buffers and image noise. Bako, Vogels, et al. (2017) propose a supervised learning approach that enables complex filtering kernels by using DCNNs, which may predict the noise-free values per-pixel directly by non-linearly combining features from the auxiliary buffers. Path space filtering is improved with a jittered spatial hashing method by Binder, Fricke, et al. (2018). As our own filtering approach is related to path space filtering, employing the according jittering method in our caching system described in Chapter 5 seemed like a natural fit and helped tremendously with reducing quantization artifacts.

The work by Lehtinen, Munkberg, et al. (2018) provides information about how clean images can be restored from noisy or corrupted input without having any uncorrupted, noise-free reference data. Their method is applied to MC noise, photographs and MRI scans. Based on these findings, Krull, Buchholz, et al. (2019) enhance the method by not requiring noisy image pairs or clean target images, which makes it possible to apply their approach in almost arbitrary situations. Koskela, Immonen, et al. (2019) propose a technique that computes regression-based filtering for individual blocks and works with 1 SPP input in real-time. Using QR factorization and stochastic regularization, their method produces high-quality images with a performance that is at least on-par with other state-of-the-art techniques. A survey of methods for MC denoising methods is provided by Mir (2020).

2.5 Caching and Reprojection

In this section, we give an overview of the most relevant research regarding caching and reprojection techniques. While reprojection techniques are relevant for the methods we chose in Chapter 4, caching is closely related to our contributions presented in Chapter 5.

An early approach to caching illumination information is proposed by Lafortune and Willems (1995). With their 5D data structure being a straightforward extension of an octree to five dimensions, it is easy to implement and allows for producing unbiased results. Instead of directly visualizing the contents of the cache, which would clearly yield a biased result, it is used to guide an adaptive importance sampling pro-

cess. By extending the definition of irradiance to all points and directions in space, Greger, Shirley, et al. (1998) present some explorative work on the possibility of using a volumetric representation of irradiance for approximating illumination effects in situations where standard GI rendering is just not fast enough. The proposed method is suitable for semi-dynamic scenes that exhibit a large number of geometric features. Similar to our own caching technique presented in Chapter 5, queries can be processed in constant time. With the resulting visualization's precision, the technique is mostly expected to be used as a replacement for the ambient term. Generally, it exhibits large similarities to Ward's irradiance caching, but mainly aims for scenes where a large number of objects is static and a smaller number of objects may be dynamic, while the irradiance cache only supports static scenes by itself. Bala, Dorsey, et al. (1999) developed methods for exploiting properties of object space, ray space, image space and temporal coherence to decouple shading and visibility computation in ray tracing and accelerate them independently. They introduce quadrilinear surface interpolants that allow for an error-bounded way of interpolating radiance from the acquired samples, where the error-bound is a user-specified parameter that makes it possible to individually handle the tradeoff between quality and performance. The reported speedups against a basic ray tracer are between 4 and 8.5 for complex scenes with hundreds of thousands of triangles. For accelerating visibility computation, the presented work utilizes a reprojection strategy for accelerating visibility computations, which exhibits a strong relation to our own work on the acceleration of foveated ray tracing. Walter, Drettakis, et al. (1999) introduce the Render Cache, a method based on per-pixel reprojection and interpolation of preceding frames that already ran interactively in a pure software implementation for Whitted-style ray tracing when it was published in 1999. Ward and Simmons (1999) present the holodeck ray cache, an out-of-core approach for caching the illumination data of a scene. Allowing for interactive exploration, it uses progressive, parallel ray tracing and also supports non-diffuse materials. By providing a suitable driver interface, various display systems can be targeted.

Simmons and Séquin (2000) propose a cache-based system for interactively exploring dynamically sampled environments, using a mesh-based reconstruction the authors call *Tapestry*. This is especially noteworthy because the mesh-based approach is a fundamental difference to approaches like the Render Cache, which works exclusively on individual pixels. Their mesh-based display and cache representation allow for continuous refinement and modification, while the reconstruction step also serves as a sampling guide for the renderer. With two processors, the system achieved interactive frame rates at a resolution of 3000×1000 .

An important extension to the original Irradiance Cache by Ward, Rubinstein, et al. (1988) is introduced by Krivanek, Gautron, et al. (2005). In addition to purely diffuse materials, their system also supports slightly glossy materials, which are

reconstructed using sparse sampling and interpolation. Spherical or hemispherical bases are used to represent the incident radiance at a point by a coefficient vector. The authors provide results that show superior quality when compared to plain path tracing, although the kind of artifacts that appear in the images are visually different.

While irradiance caching and radiance caching deliver visually convincing results, the implementation of the original algorithms on graphics hardware is rather inefficient because of the constant switching between data insertions and queries. Gautron, Krivánek, et al. (2005) propose an alternative implementation that avoids such issues and is tailored specifically for GPU implementations. Consequently, the implementation of their approach yields a significant speedup of more than one order of magnitude over CPU-based rendering. Krivánek, Bouatouch, et al. (2006) improve upon their original radiance caching method by introducing a novel technique for adaptively guiding the spatial density of cache samples according to the frequency of indirect illumination, reducing visible interpolation artifacts significantly. In addition, they propose a new approach to reduce light leaks, called *neighbor clamping*.

In situations of continuous movement, temporal coherence between subsequent frames is relatively large. Nehab, Sander, et al. (2007) exploit this by caching shaded fragments and reusing them in consecutive frames, exhibiting only a slight quality degradation. In addition, the authors provide guidelines for cache management and the selection of data to be cached. Debattista, Dubla, et al. (2009) introduce *Instant Caching*, a hybrid method based on combining irradiance caching and instant radiosity. By reusing computations of neighboring geometry intersections, they achieve a speedup over previous instant radiosity techniques. Also, they exploit temporal coherence by avoiding updates of coherent intersection points and instead only performing updates where necessary. At the same time, the presented system also supports lighting and material changes. Krivanek and Gautron (2009) provide a collection of all relevant information for implementing irradiance caching in arbitrary software frameworks, bringing together all findings and techniques of recent years.

Papaioannou (2011) describes a method for caching diffuse-only global illumination in sparsely distributed evaluation points spread throughout the scene, using a combined method relying on elements from reflective shadow maps and grid-based radiance caching algorithms. In order to achieve interactive performance with multiple light bounces, a method for stochastic visibility calculations is suggested.

A bidirectional reprojection technique is introduced by Yang, Tse, et al. (2011). With speedups of 3 to 4 compared to unoptimized rendering, their technique meets the expectations of an increased frame rate, while introducing a small amount of lag which the authors analysed in a user study, finding that it is insignificant. Their

approach is implemented in two variants, one for scenes that are bound by fillrate (only reducing shading), and another one that reduces both shading and geometry computations by only relying on image-based buffers.

Georgiev, Křivánek, et al. (2012a) propose a technique for caching *importance* at sparsely distributed locations in space. The most significant information included in this importance is local visibility, which may be a central cause of variance in scenes that are highly occluded. The authors demonstrate that by using their caching approach together with the novel multiple importance sampling framework, variance can be reduced by more than one order of magnitude for identical computation times. Lehtinen, Aila, et al. (2012) introduce a method for reconstructing a scene’s indirect illumination through a sparsely sampled light field. The proposed technique can handle glossy and anisotropic materials and is only responsible for reconstructing indirect illumination, while direct illumination remains untouched. In the reconstruction step, scene geometry is not considered; instead, the authors provide a robust visibility heuristic.

Scherzer, Nguyen, et al. (2012) present an approach for avoiding the storage of a spherical harmonics function per pixel in radiance caching by replacing it with MipMap-based pre-filtered data. According to the authors, this results in a constant lookup time per pixel. By storing statistics about geometry, the quality of each lookup is also improved. Compared to basic radiance caching, pre-convolved radiance caching is about one order of magnitude faster, while it can be combined with other methods like instant radiosity or MC-based ray tracing. By introducing a novel data structure referred to as *Equivalent Area Light Sources*, Omidvar, Ribardière, et al. (2015) present a new approach to Radiance Caching that also allows for the detailed reconstruction of highly glossy surfaces. At the same time the amount of data that has to be stored for reconstruction is reduced when compared to methods based on spherical harmonics.

In his PhD thesis, Roughton (2019) describes methods for the interactive generation of path-traced lightmaps, a purpose that our HashCache system could also be utilized for. Hirvonen, Seppälä, et al. (2019) improved aspects of earlier work like radiance probes and screen space reflections and added ray tracing to the pipeline. By doing this, visual errors are significantly reduced and accurate global reflections can be computed. For rough surfaces, their approach relies on purely cache-based reconstruction without employing a true ray tracing step. By utilizing a novel strategy for the placement of radiance records and adapting specific error metrics, Zhao, Belcour, et al. (2019) developed a new radiance caching approach that is much better at reconstructing interreflections between glossy materials without introducing an additional performance impact.

2.6 Discussion and Conclusion

In this chapter, we have reviewed the available literature in the fields that are relevant for this thesis.

For our work on guided high-quality rendering, it is important to note that the approach we present in Chapter 3 can be combined with many of the presented optimizations for ray-based rendering, as it is a general scheduling approach. Previous rendering systems did not allow for comfortably and dynamically adapting the sampling rates to the user's requirements by interacting with virtual reality (VR) input devices on large display systems, which is the central contribution of our work, leading to significant speedups in relevant scene parts. In this process, we make use of insights from research on the HVS in order to design the region of interest (RoI) employed in the system.

With our work published in 2016, our approach presented in chapter Chapter 4 was among the first to implement a modern gaze-contingent rendering system for HMDs that uses ray tracing for interactive visualization and meets the HMDs refresh rate. Today, multiple approaches for foveated rendering in HMDs are available, such as the ones based on visual-polar space.

The caching approach presented in Chapter 5 relies on insights from other caching-based methods like irradiance caching, radiance caching, radiosity or photon Mapping, but to our knowledge it is the first to use a hash-based data structure in order to quickly reconstruct the cached data. Also, our we improve upon other work by providing the means for accurately reconstructing non-diffuse illumination. With the layered filtering approach proposed in the same chapter, we combine insights from screen space and path space filtering algorithms to provide an actual filtering framework. This does not rely exclusively on primary hitpoint information, as other ordinary screen space filtering methods do, but it considers multiple hitpoints for each path, similar to true path space filtering, but projected to screen space.

Chapter 3

Guided High-Quality Rendering

Part of the work included in this chapter was previously published in (Roth, Weier, et al. 2015), lead-authored by the author of this thesis. The idea is based on the work published in (Weier, Maiero, et al. 2014b) and (Weier, Maiero, et al. 2014a), co-authored by the author of this thesis, who made substantial contributions to the implementation.

The main goal of the work presented in this chapter is the development of a system which allows for guiding the image quality in global illumination (GI) methods by user-specified regions of interest (RoIs). This can be achieved with either a tracked interaction device on an large high-resolution display wall (LHRDW) or a projection-based system, or a mouse-based method in a desktop environment, making it possible to create a visualization with varying convergence rates throughout a single image towards a sufficiently accurate GI solution. To achieve this, we introduce a scheduling approach based on sparse matrix compression (SMC) for efficient generation and distribution of rendering tasks on the GPU that allows for altering the sampling density over the image plane while avoiding low GPU utilization. Moreover, we implemented an approach for filtering the samples computed so far to a final image that exhibits the correct overall brightness despite the sparse sampling process. The presented methods are benchmarked and discussed, drawing conclusions regarding possible implications for future research in the field.

3.1 Introduction

Modern, large display setups like 4K projection systems or even LHRDWs, like our own system HORNET (Applied Sciences 2014) allow for highly detailed visualizations, while modern PCs enable the implementation of (embarassingly) parallel algorithms like path tracing directly on the GPU. While memory requirements can be handled quite well by current hardware when computing global illumination even in a 4K setup or above, real-time Monte Carlo (MC)-based GI at such high resolutions

remains challenging because of the high computational complexity.

In this chapter, we present an approach for coping with the situation at hand, where designers would like to have a quick, high-quality visualization of products or a scene preview at high resolutions, while modern GPUs are not yet fast enough to compute high-resolution, physically-based GI in an acceptable time frame. Our basic assumption is that in design review, architectural visualization and similar fields, it is often possible to divide an image into areas of varying significance for the user. This implies that the focus, i.e., the user's gaze, will mostly be around regions with higher significance, making the convergence of areas outside these main regions less important as they are not perceived in a detailed way by the human visual system (HVS). Nonetheless, it is still important to provide a visual context through the areas outside the mainly focused regions.

While the GI rendering method we employ throughout this chapter is path tracing, the presented methods are applicable to all rendering methods dealing with similar issues. As our work focuses mainly on large 4k projection systems, which could not be provided in our lab, we decided to use our own LHRDW HORNET, made from 7×5 displays with a 1080p resolution each, to mimic such a system. A schematic overview of HORNET is shown in Figure 3.5.

3.2 System Description

The basic idea of our work is to develop a framework that enables adaptive high-quality rendering that is adapted to the user's current requirements. To achieve this, a number of building blocks are introduced that take care of the various tasks that have to be carried out during the rendering process. These building blocks are described in Section 3.2.1. We also provide exemplary implementations of each building block and plug them together into a complete system in Sections 3.2.2 to 3.2.7.

3.2.1 Building Blocks

Figure 3.1 shows an overview of the proposed building blocks. The interaction block is responsible for getting information on how to construct the RoI and also for computing it, which is done in a per-pixel manner in our exemplary implementation. It is necessary to provide a method for determining the actual importance of a pixel in order to compute such an RoI. A user-defined distance measure is used that takes into account the image resolution, the focused pixel coordinates and a set of user-defined parameters like the user's position relative to the display system. Based

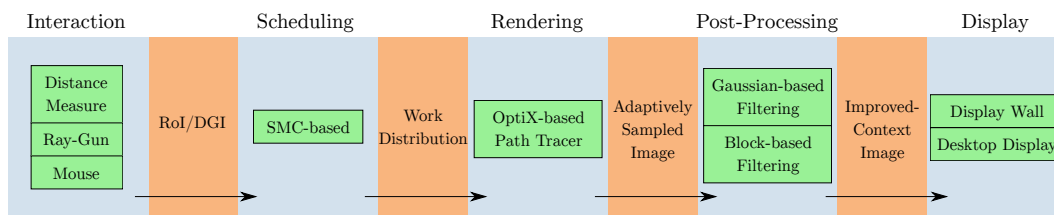


Figure 3.1: Building blocks of the proposed framework, exemplary implementations and data flow. Blue boxes represent the building blocks, containing exemplary implementations in green. The orange regions illustrate the data passed between blocks.

on this data, the distance to the RoI is then computed for each individual pixel. Section 3.2.2 provides an example for such a distance measure.

In the exemplary implementation, the RoI is represented as a per-pixel lookup table (LUT), referred to as the distance-guide image (DGI). While it would certainly be possible to perform the actual computation of the DGI values on-the-fly, we chose the LUT-based approach because the RoI is only modified on-demand in our current implementation. Additionally, our proposed scheduling approach makes use of the DGI stored in memory. The DGI is assumed to have the same resolution as the rendered image and stores values in $[0, 1]$.

This distance information is used by the scheduling block to apply a suitable scheduling algorithm, depending on the representation of distance values as well as the targeted processor architecture employed for rendering.

The generated task distribution is then used to efficiently render an image accounting for the user-defined RoI in the rendering block with a suitable rendering method. As the result is a potentially sparsely sampled image, there may be regions with high variance due to low sample numbers, including completely unsampled pixels outside the RoI.

Because these areas tend to be perceptually disturbing, we propose to include a post-processing block that increases the image quality in order to provide a better visual context to the user. Image parameters such as brightness can be locally adapted to yield a more homogenous appearance and filtering techniques like denoising methods may be applied.

The resulting image is then forwarded to the display block, which is an abstract representation of the actual display system. The following subsections provide further information on how the individual components can be implemented by giving a few examples.

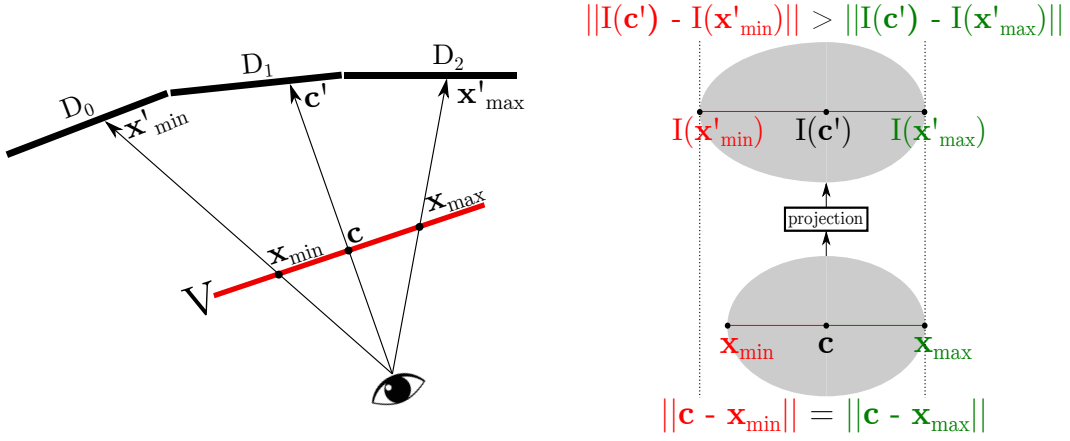
3.2.2 Distance Measure

As the distance measure should be viable for both usage scenarios, the exemplary measure we developed is based on the field of view (FoV)’s elliptical shape in binocular vision. Hereinafter, RoI refers to a concrete region of interest with a location and extents, while RoI refers purely to the extents of such an RoI, but not to a specific location in screen space. The steps to compute the distance measure are described below in detail. To explain the basic idea, we provide an overview of the main steps here:

- (1) Specify an elliptical shape on a plane perpendicular to the gaze vector of the utilized interaction device, with the center being positioned at unit distance to the device. This means we start with an *ellipse*, defined by its *center*, *vertex* (defining the horizontal radius) and *co-vertex* (defining the vertical radius).
- (2) Generate five rays originating at the interaction device, running through the ellipse’s according center and four boundary vertices.
- (3) Intersect these rays with the display system. At this point, the projection will likely transform the ellipse to an oval, either due to the shape of the display system or the interaction device’s orientation.
- (4) For all points in the image, find the closest point on that oval. The method described below shows that because the closest point on the oval is always in the quadrant in which the query point is located, the necessary computations are identical to an ellipse. This means that we have come from an originally elliptical shape to a projected oval and then back to the ellipse, which is easier to handle because of its symmetry.

To determine the projection on the display system, the view center \mathbf{c} and four boundary points are used, defining the horizontal and vertical extents of the RoI ($x_{\min} \in \mathbb{R}_0^-, x_{\max} \in \mathbb{R}_0^+$ for the horizontal boundaries, $y_{\min} \in \mathbb{R}_0^-, y_{\max} \in \mathbb{R}_0^+$ for the vertical boundaries). These boundary points are referred to as *vertices* for the horizontal boundaries and *co-vertices* for the vertical boundaries, derived from the according terms used in the definition of an elliptical shape. For our distance measure, we require four boundary points instead of just working with the ellipse’s vertex and co-vertex, as an elliptical shape becomes an oval when the projection onto a surface is not strictly perpendicular and (different) relative display orientations have to be considered.

When using our application on the LHRDW, we determine the pixel coordinates of these points by intersecting rays with a virtual model of the display system, making no assumptions about its shape, but using an exact geometric representation.



- (a) Top view of the projection process: The view plane V is located at unit distance from the interaction device and oriented orthogonally to $\mathbf{c}' - \mathbf{c}$. Projecting the ellipse defined on V onto the display system, represented by three displays D_0, D_1, D_2 with a common image space, yields intersection points $\mathbf{x}'_{\min}, \mathbf{c}', \mathbf{x}'_{\max}$. Using the according coordinates in image space leads to an oval instead of an elliptical shape.
- (b) Shape transformation through the projection process: The elliptical shape defined on V (bottom) is transformed into an oval shape on the display system (top). The intersection points are converted from the global coordinate system to pixel coordinates by $I: \mathbb{R}^3 \rightarrow \mathbb{R}^2$

Figure 3.2: Projection process of the ellipse defined on V onto the display system.

For our LHRDW HORNET, the display wall is curved with a 10° angle between each display column. Figure 3.3 shows an example of the resulting DGI when aiming the ray-gun at our LHRDW.

The asymmetric oval shape o results from the different distances between the user's position and the intersection points on the display wall, leading to a change of shape after the projection, as illustrated in Figure 3.2

The probability $\text{prob}(\mathbf{p})$ for sampling each individual pixel $\mathbf{p} \in \mathbb{R}^2$ is determined by computing the minimum distance $d_{\min}(\mathbf{p}, o)$ to any point inside the projected oval shape o , normalized to $[0, 1]$. The normalization factor only depends on the size of the display system and does not vary between frames. Thus, normalization is achieved by dividing the actual distance value by the diagonal of the full view space. As the horizontal and vertical extent of the view space are also normalized to $[0, 1]$ in the utilized intersector that determines the hitpoints on the display system, this divisor is $\sqrt{2}$. Therefore,

$$\text{prob}(\mathbf{p}) = 1 - \frac{d_{\min}(\mathbf{p}, o)}{\sqrt{2}} \quad (3.1)$$

is the sampling probability for \mathbf{p} , where d_{\min} is transformed to $[0, 1]$ based on normalized device coordinates. For pixels inside the projected shape, this results in a distance value of 0 and thus a sampling probability of 1.

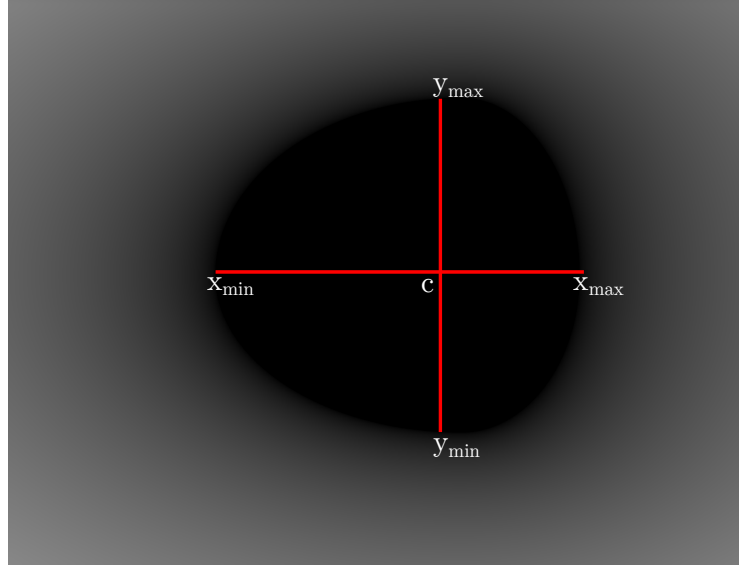


Figure 3.3: Exemplary shape of the resulting DGI when aiming the ray-gun at the display wall. It is clearly visible how the non-orthogonal perspective leads to an oval shape of the DGI.

To determine if a point $\mathbf{p} = (x, y)$ is within o , it is projected to a unit circle according to the center and boundary points o is based upon. First, the quadrant $Q(o, \mathbf{p})$ of o is determined in which \mathbf{p} is located:

$$Q(o, \mathbf{p}) = \begin{cases} \text{I,} & \mathbf{p}_x > \mathbf{c}_x \wedge \mathbf{p}_y > \mathbf{c}_y \\ \text{II,} & \mathbf{p}_x \leq \mathbf{c}_x \wedge \mathbf{p}_y > \mathbf{c}_y \\ \text{III,} & \mathbf{p}_x \leq \mathbf{c}_x \wedge \mathbf{p}_y \leq \mathbf{c}_y \\ \text{IV,} & \mathbf{p}_x > \mathbf{c}_x \wedge \mathbf{p}_y \leq \mathbf{c}_y \end{cases} \quad (3.2)$$

Based on the quadrant, the closest boundary points are chosen from $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ and $\mathbf{s} \in \mathbb{R}^2$ is computed as:

$$\mathbf{s}_x = \begin{cases} x_{\min}, & Q(o, \mathbf{p}) \in \{\text{II, III}\} \\ x_{\max}, & \text{otherwise} \end{cases} \quad (3.3)$$

$$\mathbf{s}_y = \begin{cases} y_{\min}, & Q(o, \mathbf{p}) \in \{\text{III, IV}\} \\ y_{\max}, & \text{otherwise} \end{cases}$$

By computing $H(\mathbf{p})$, we can now determine if \mathbf{p} is within o :

$$H(\mathbf{p}) = \begin{cases} 1, & \|\text{abs}(\mathbf{p} - \mathbf{c}) \cdot \mathbf{s}^{\circ-1}\| \leq 1 \\ 0, & \text{otherwise} \end{cases}, \quad (3.4)$$

where $\mathbf{A}^{\circ-1}$ is the *Hadamard inverse* with $(\mathbf{A}^{\circ-1})_{ij} = (\mathbf{A}_{ij})^{-1}$, i.e., the component-wise inverse of \mathbf{A} .¹

If $H(\mathbf{p}) = 1$, d_{\min} is returned as zero, while otherwise we compute $d_{\min}(\mathbf{p}, o)$ as the minimum distance of \mathbf{p} to o . As we only consider the specific quadrant of o that \mathbf{p} is located in, we can simplify o by assuming horizontal and vertical symmetry, thus resulting in an axis-aligned ellipse e with \mathbf{s}_x as the vertex and \mathbf{s}_y as the co-vertex. Assuming such an ellipse is centered at the coordinate system's origin, each point $P_e(\phi)$ on e can be described as

$$P_e(\phi) = \begin{pmatrix} \mathbf{s}_x \cos \phi \\ \mathbf{s}_y \sin \phi \end{pmatrix}, 0 \leq \phi < 2\pi, \quad (3.5)$$

so that the distance from any point \mathbf{p} to a point $P_e(\phi)$ on the ellipse can be computed by

$$d(\mathbf{p}, e, \phi) = \sqrt{(\mathbf{p}_x - P_e(\phi)_x)^2 + (\mathbf{p}_y - P_e(\phi)_y)^2}. \quad (3.6)$$

To find the closest point \mathbf{q} on e , $d(\mathbf{p}, e, \phi)$ needs to be minimized, i.e., we need to compute ϕ_{\min} so that $d(\mathbf{p}, e, \phi_{\min}) < d(\mathbf{p}, e, \phi)$ for any ϕ in $Q(o, \mathbf{p})$. Eberly (2019) presents an appropriate method which we describe shortly below. For specific details on the mathematical approach and the implementation, please refer to the original work. Eberly makes two key observations that lead to the final algorithm:

- With the query point \mathbf{p} being defined in e 's local coordinate system, it is generally possible to move the query point to another quadrant by changing the signs accordingly. After determining the closest point, the sign changes simply need to be undone to find \mathbf{q} .
- The vector that is normal to e at \mathbf{q} points toward \mathbf{p} .

The implicit form of an ellipse is given by

$$G(\mathbf{q}) = \left(\frac{q_x}{s_x}\right)^2 + \left(\frac{q_y}{s_y}\right)^2 - 1 = 0.$$

The normal vector of a point on the ellipse can be computed as $\vec{\mathbf{n}}(\mathbf{q}) = \frac{1}{2}\nabla G(\mathbf{q})$.

¹The notation of the *Hadamard inverse* is derived from the *Hadamard product*, where $\mathbf{C} = \mathbf{A} \circ \mathbf{B}$ is the component-wise multiplication with $\mathbf{C}_{ij} = \mathbf{A}_{ij}\mathbf{B}_{ij}$.

Since we know that $\mathbf{p} - \mathbf{q}$ has to be normal to the ellipse, there has to be some scalar t for which holds:

$$t \frac{\nabla G(\mathbf{q})}{2} = t \left(\frac{q_x}{s_x^2}, \frac{q_y}{s_y^2} \right) = \mathbf{p} - \mathbf{q}.$$

Based on this representation and the above observations, \mathbf{p} can be written as

$$\begin{aligned} \mathbf{p}_x &= \mathbf{q}_x \left(1 + \frac{t}{s_x^2} \right), \\ \mathbf{p}_y &= \mathbf{q}_y \left(1 + \frac{t}{s_y^2} \right), \end{aligned} \quad (3.7)$$

where $t > 0$ holds for all points outside e . Because \mathbf{p} is always in the first quadrant due to the conversion above, Equation (3.7) has the following solution for some scalar t :

$$\begin{aligned} \mathbf{q}_x &= \frac{s_x^2 \mathbf{p}_x}{t + s_x^2}, \\ \mathbf{q}_y &= \frac{s_y^2 \mathbf{p}_y}{t + s_y^2}. \end{aligned} \quad (3.8)$$

Based on the relation between an ellipse and a unit circle, $(x/s_x)^2 + (y/s_y)^2 = 1$ for all points (x, y) on e . Together with Equation (3.8), we obtain

$$F(t) = \left(\frac{s_x \mathbf{p}_x}{t + s_x^2} \right)^2 + \left(\frac{s_y \mathbf{p}_y}{t + s_y^2} \right)^2 - 1, t \geq -s_x^2 \quad (3.9)$$

and the candidates for \mathbf{p}_{\min} are generated by the roots t to $F(t) = 0$. Eberly suggests to use bisection as the method for finding the unique root of $F(t)$, which is also the approach we chose for our implementation. After the minimum distance $d_{\min}(\mathbf{p}, o)$ has been computed, it can be put into Equation (3.1) to compute the sampling probability for \mathbf{p} .

3.2.3 Interaction

For our system, we chose to implement two interaction methods for selecting an RoI, each suited for its own use case. Besides the methods presented here, experiments have also been conducted with head tracking, as it seemed to be the most natural approach to select a RoI. However, forcing the user to stare at the same region until the desired image quality is achieved is too much of a drawback, as the time necessary to achieve a satisfactory convergence may be anywhere between mere seconds and, in complex cases, hours. Thus, we decided that head tracking was not a practically feasible solution for the task at hand. Clearly, the same reasoning applies for the use



Figure 3.4: ART’s interaction device *FlyStick*. Image courtesy of (Tracking 2020)

of eye tracking. In the following two paragraphs, we introduce the “Ray-Gun”-based as well as the mouse-based interaction.

Ray-Gun This method is based on ART’s interaction device called the *FlyStick*, which is used for pointing at the display system and “shooting” a ray through the center of the desired RoI on demand. Additionally, the horizontal and vertical extents of the RoI may be adjusted via the FlyStick’s analog stick. Exemplary pictures of the FlyStick are shown in Figure 3.4.

To determine the RoI’s projection on the display system, a ray is traced from the FlyStick to the virtual model of the display system, resulting in the RoI’s center. Accordingly, four additional rays are generated and intersected with the display system to determine the projection of the RoI-defining ellipse e , which will generally result in an oval shape (see Section 3.2.2). The orientation of these rays is determined by placing a plane one unit from the FlyStick’s position, perpendicular to the optical axis. The intersection of the optical axis with this plane is the center of e , referred to as \mathbf{c} . The boundary vertices can now be computed as follows:

$$\mathbf{x}_{\min} = \mathbf{c} - \begin{pmatrix} \mathbf{s}_x \\ 0 \end{pmatrix}, \mathbf{x}_{\max} = \mathbf{c} + \begin{pmatrix} \mathbf{s}_x \\ 0 \end{pmatrix}, \quad (3.10)$$

$$\mathbf{y}_{\min} = \mathbf{c} - \begin{pmatrix} 0 \\ \mathbf{s}_y \end{pmatrix}, \mathbf{y}_{\max} = \mathbf{c} + \begin{pmatrix} 0 \\ \mathbf{s}_y \end{pmatrix}. \quad (3.11)$$

The boundary rays are now oriented from the FlyStick’s origin towards the aforementioned points, so that their intersection with the display system yields the desired RoI.

Mouse An RoI can also be defined by using a mouse, simply defining its center with a click at the according location on screen. In a projection system, this is more of a fallback method when no tracking is available. However, it can prove useful in a high-resolution desktop environment. Due to the user’s lower distance to the display

system and the higher pixel density, the aforementioned RoI generation method is also employed in this case. Although, as we assume that there is no tracking system in a desktop environment, the direction for projecting the FoV is perpendicular to the display surface, which is considered to be plane. This projection is performed in an orthogonal way, as no distance between the user and the display system is known. Instead, the size of the RoI can be defined in a graphical user-interface when using the mouse.

3.2.4 Scheduling

With the sampling probability for a pixel \mathbf{p} given in Equation (3.1), a binary sampling decision has to be made for each pixel individually. Obviously, it would be possible to make this decision in the actual kernel execution on the GPU and simply let the according thread return if the decision is negative. However, for sparse data like in our case, this may lead to a large amount of thread divergence, with idle threads leading to a performance hit due to unused resources and low GPU utilization. Because of this, we choose a different approach by providing an actual scheduling method that avoids such idle threads. The sampling decision is thus made in a separate CUDA kernel and stored in a binary image with the same resolution as the final rendered image, where a pixel value of 0 represents a negative sampling decision, and 1 a positive one. Because of the probabilistic approach, the actual number of samples computed in each iteration varies. Our approach to scheduling only the relevant pixels for rendering is supported by CUDA's *cuSparse* library, which supports sparse matrix compression (SMC). By interpreting the sampling decision image as a matrix, an SMC algorithm may be applied which effectively yields the pixel coordinates of non-zero values, i.e., the pixels that actually have to be sampled. Additionally, this provides the total number of non-zero values, which enables us to launch the rendering process with the exact number of threads that are required. At the same time, the pixel area belonging to each thread is directly determined by looking up the coordinates in the coordinate arrays.

$$s(\mathbf{p}) = \begin{cases} 1, & \text{if } \xi \leq \text{prob}(\mathbf{p}), \xi \sim U(0, 1) \\ 0, & \text{otherwise} \end{cases} \quad (3.12)$$

$$\text{Sampling decision} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{\text{SMC}} \begin{matrix} \text{Columns} = (2 \ 1 \ 3 \ 0 \ 1 \ 3 \ 3) \\ \text{Rows} = (0 \ 1 \ 1 \ 2 \ 2 \ 2 \ 3) \end{matrix} \quad (3.13)$$

While the concept of using SMC for this purpose is straightforward, we still need to determine the actual benefits of this approach. The according measurements and benchmarks are presented in Section 3.3.

3.2.5 Rendering

The rendering component is implemented using our path tracer *Spark*, based on NVIDIA’s OptiX framework, which means that the rendering process is completely done on the GPU. For a description of features of *Spark*, see the basic extensions to path tracing described in Section 2.1.6.

3.2.6 Filtering

In addition to the noise caused by the MC process, Figure 3.6a shows that the varying noise and sampling density outside the RoI can be visually disturbing. We propose two optional filters to overcome this issue and provide an improved visual context to the user:

- A Gaussian-based filter with varying kernel size
- A block-based filter that sets all pixels in a square area to their mean value

The implementation of these filters is still prototypical and would require additional optimizations for continuous use in an interactive environment. Nonetheless, they serve to give a general idea of how to improve the visual context in a scenario with sparse sampling.

As explained above, a probabilistic sampling decision is made for each pixel of the image, leading to pixels that have not been sampled at all, especially early in the rendering process. The presented filtering methods only account for pixels that have already been sampled in order to achieve an overall brightness similar to the final image right at the beginning.

The Gaussian filter works with the following parameters:

- k_{\max} : Maximum kernel radius, user-defineable
- $d_{\min}(\mathbf{p}, o)$: Distance value from the DGI
- $w_{\mathbf{p}} = \max\{0, 1 - n_{\mathbf{p}}/c\}$, where $n_{\mathbf{p}}$ is the current number of samples already computed for the specific pixel and c is a blending constant
- $k_{\mathbf{p}} = w_{\mathbf{p}} \cdot k_{\max} \cdot d_{\min}(\mathbf{p}, o)$: Kernel radius for pixel \mathbf{p} , with $\sigma = k_{\mathbf{p}}/3$

By using the blending constant, it is possible to provide a visual context to the user that estimates the ground truth’s overall brightness early in the rendering process, while the image still converges to the ground truth with $n \rightarrow \infty$. The proposed blending method could be modified so that it relies on pixel variance instead of a blending constant to account for the actual noise present in the image. With k_p being the kernel size for the Gaussian filter implementation, the same parameter determines the block size in block-based filtering, while limiting the block sizes to powers of two for improved alignment of the blocks. The blending method is identical in both methods. Exemplary results of the filtering methods are shown in Section 3.3.

3.2.7 Display

As mentioned above, we employ our LHRDW HORNET, consisting of 7×5 1080p displays to mimic a curved 4k projection system. Figure 3.5 shows a schematic illustration of the relevant parts of HORNET. Rendering is performed on the Rendering Node.

3.3 Results

3.3.1 Visual Quality

For the proposed system, our main efforts regarding the optimization of image quality were aimed towards improving the visual context by approximating the true image brightness in sparsely sampled image regions, which can be judged visually. The only numeric quality metric we used here is strictly related to the system’s increased performance, consequently reducing noise, which is explained in more detail in Section 3.3.2. Figure 3.6 shows an image rendered with our system at 128 samples per pixel (SPP) in an unfiltered state, filtered with the Gauss-based filter and filtered with the block-based filter. While the brightness variation in the unfiltered image may be visually disturbing, the visual context provided in the filtered images is clearly improved.

Figure 3.7 shows photographs of the rendering system on our LHRDW. The effect of the proposed filtering method is obvious at all sampling rates.

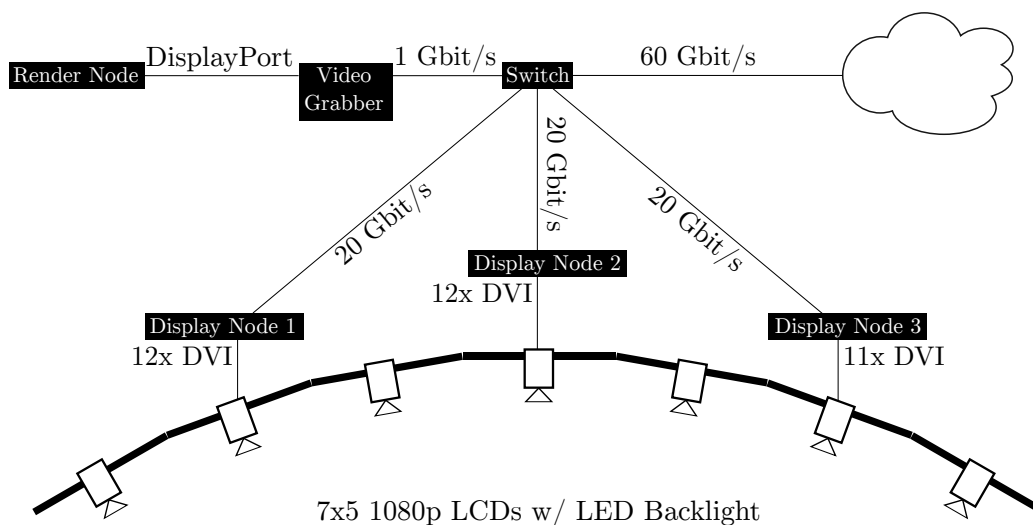


Figure 3.5: Schematic illustration of the relevant part of our large high-resolution display wall HORNET. The system consists of 7×5 displays, each with a diagonal of 46" and a 1080p resolution. Each of the Display Nodes is equipped with three NVIDIA GTX 780 graphics cards with 4 DVI outputs per GPU (with the exception of one GPU in Display Node 3, as we only have 35 displays). The Display Nodes are connected with a 20 Gbit/s optical fibre link. The rendering system generates the image data at a 3840×2160 resolution on the Rendering Node (top left), which is connected to the Grabber Node via DisplayPort. The Grabber Node then provides the data to the display Nodes via SAGE (University of Illinois at Chicago's Electronic Visualization Laboratory and University of Hawai'i at Manoa's Laboratory for Advanced Visualization and Applications 2020) as a video stream. The cloud on the top right represents the additional rendering cluster, which is not used in case of this work. HORNET is also equipped with a tracking system with one camera for each of the display columns.

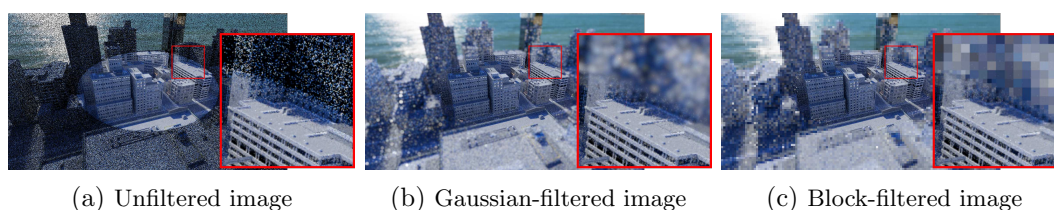


Figure 3.6: Comparison between unfiltered, Gaussian-filtered and block-filtered image, all rendered at 128 samples per pixel in the RoI. It is clearly visible how the overall great brightness variation between the RoI and the peripheral area in the unfiltered image may disturb the viewer. The Gaussian-filtered and block-filtered images overcome this by only accounting for pixels that have already been sampled.

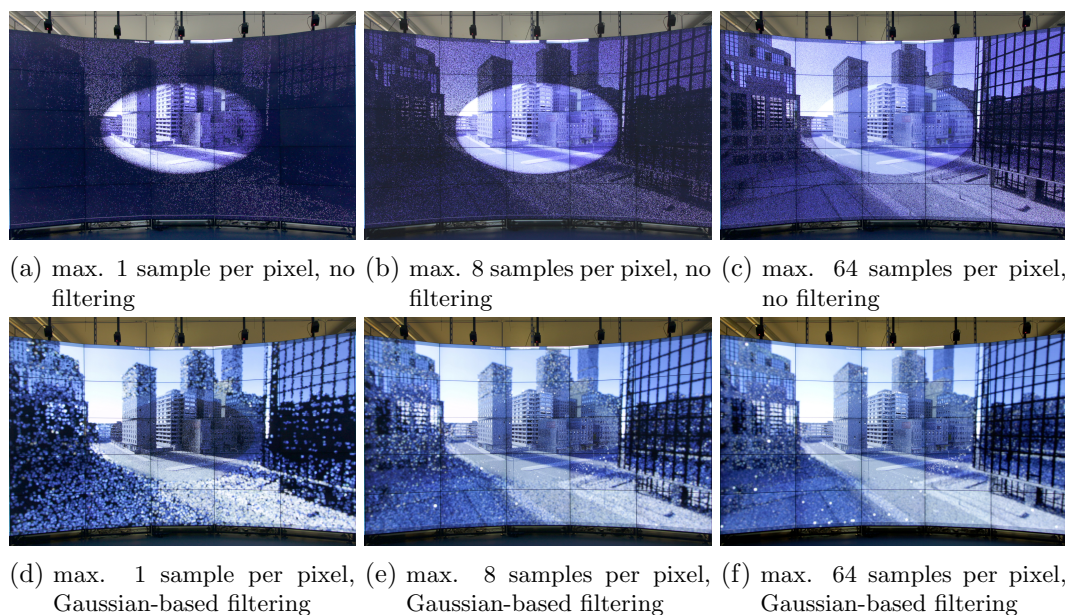


Figure 3.7: Comparison between unfiltered and filtered images displayed on our large high-resolution display wall. Filtering leads to a good approximation of the ground truth’s overall image brightness early in the rendering process.

3.3.2 Benchmarks

In this section, we present the results of various measurements for the presented system components. First, we give an overview of the hardware setup and rendering settings used in our benchmarks. Subsequently, benchmarks for the DGI generation, measurements of the actual sample density resulting from our approach, rendering efficiency, scheduling performance and error rate differences between the RoI and outer regions are presented. Reference renderings for the scenes used for the rendering benchmarks are shown in Figure 3.8.

Setup

As described above, our implementation targets high-resolution projection systems, mimicked by our LHRDW. The Render Node in the utilized setup is equipped with an Intel Xeon E5-2637, 16GiB of RAM and a GeForce GTX 980 graphics card running Linux. We use two test scenes: The first one is a simple Cornell Box with specular cuboids, the second one is the more complex city model *Urban Sprawl 2 by Stonemason* with around 350k triangles. Both scenes are exclusively lit by a high dynamic range (HDR) environment map. These models were chosen to evaluate our method for simple as well as more complex geometry, since real use cases may include both.

All benchmarks that we performed for rendering on HORNET have been configured

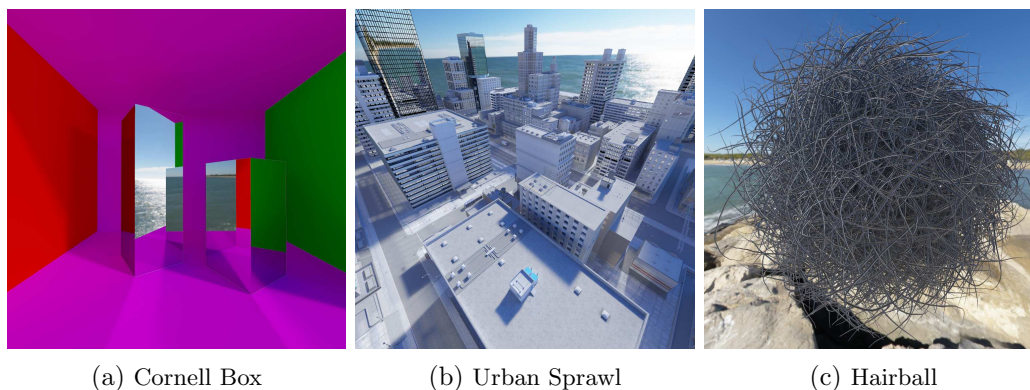


Figure 3.8: Reference renderings of the scenes used for benchmarking

with the viewpoint set to an exemplary position 1m in front of the display wall, looking towards its center. The rendering resolution for these benchmarks is 3840×2160 pixels.

We also carried out additional benchmarks to analyse the performance gained by the SMC scheduling approach. These were carried out locally on a machine equipped with an Intel Core i7-6700k, 64GiB of RAM and a GeForce Titan Xp graphics card at 1024×1024 resolution and include the additional Hairball scene, which serves to represent scenes with very dense geometry.

DGI Generation

The amount of time required to generate the DGI for a 4k resolution was between 2 and 3 milliseconds on the GeForce GTX 980, regardless of the chosen FoV.

Sparsity

The number of pixels rendered with the chosen settings for varying FoVs is illustrated by the red bars in Figure 3.9a and Figure 3.9b. The percentage of pixels that are updated per iteration is shown in the following table:

Field of View	10°	20°	30°	40°	50°	60°
Percentage of Pixels Updated per Iteration	1.2	2.9	5.7	10.5	19.0	36.3

The chosen FoVs should be seen as examples, as the concrete choice of parameters depends largely on the concrete use case. If larger portions of the image are relevant, larger fields of view will be required, while the evaluation of local details will lead to a

parameterization with a smaller FoV. Our measurements provide insights regarding the quadratic influence of the FoV on rendering times.

The data shows that the image generated in each iteration is quite sparse for all FoVs that we experimented with. Because of this high amount of sparsity, the effect of an optimized scheduling method becomes even more interesting. We compare the relation of sparsity and rendering performance for both SMC-based scheduling and unoptimized scheduling below.

Rendering Performance

The decreased ray coherence resulting from the lower sampling density was expected to have a negative impact on rendering efficiency, i.e., the time required per sample. Thus, we measured the rendering time per one million samples for various FoVs, with larger FoVs corresponding to an increased coherence because of the overall higher sample density. This should in turn lead to a lower thread divergence on the GPU.

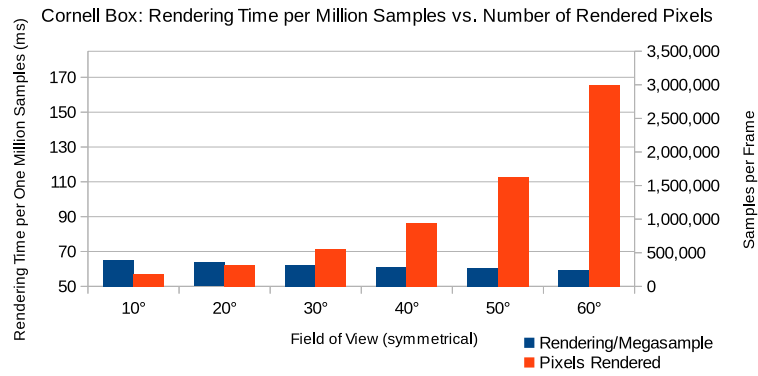
Figure 3.9a and Figure 3.9b show the rendering time per one million samples (referred to as *rendering efficiency*) for the benchmark scenes in context with the amount of rendered samples. Figure 3.9c puts the number of samples and the rendering efficiency into context directly. Rendering efficiency clearly improves with an increasing amount of samples rendered per iteration. We attribute this to the reduced number of idle threads, leading to a higher GPU utilization. The effect of scene complexity is obvious from the diagrams, as the Cornell Box took less than half the amount of rendering time compared to Urban Sprawl.

Scheduling Performance

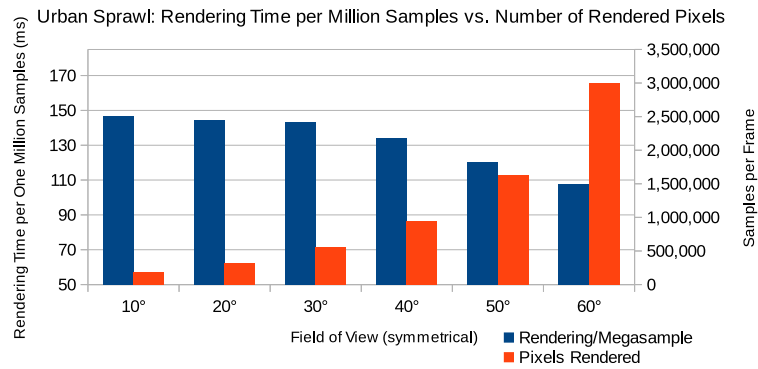
We tested the effectiveness of SMC on rendering performance for three scenes: Cornell Box, Urban Sprawl and Hairball.

Figures 3.10 to 3.12 illustrate renderings of the Cornell Box, Urban Sprawl, and Hairball scenes with the various RoI radii that were used for benchmarking, given as a fraction of the image width. These renderings were done at 1 sample per pixel.

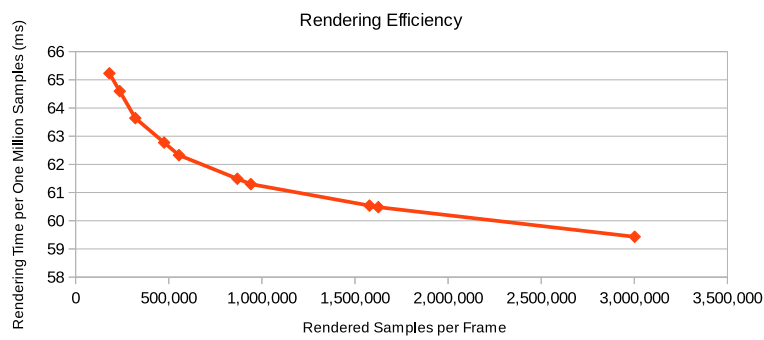
Figures 3.13 to 3.15 show the influence of SMC-based scheduling on rendering times for varying RoI sizes in all three tested scenes. For the Cornell Box, each of the diagrams shows a different maximum recursion depth, ranging from 1 (only primary rays are traced) to 4 bounces. Increasing the RoI radius brings the rendering times of unoptimized scheduling and SMC-based scheduling closer together, with SMC surpassing unoptimized scheduling between radii of 0.35 and 0.4 for a recursion



(a) Number of computed samples vs. rendering time per one million samples for the Cornell Box scene



(b) Number of computed samples vs. rendering time per one million samples for the Urban Sprawl scene



(c) Rendering Efficiency for Cornell Box

Figure 3.9: Computed samples for various FoVs and rendering times per one million samples (megasample).

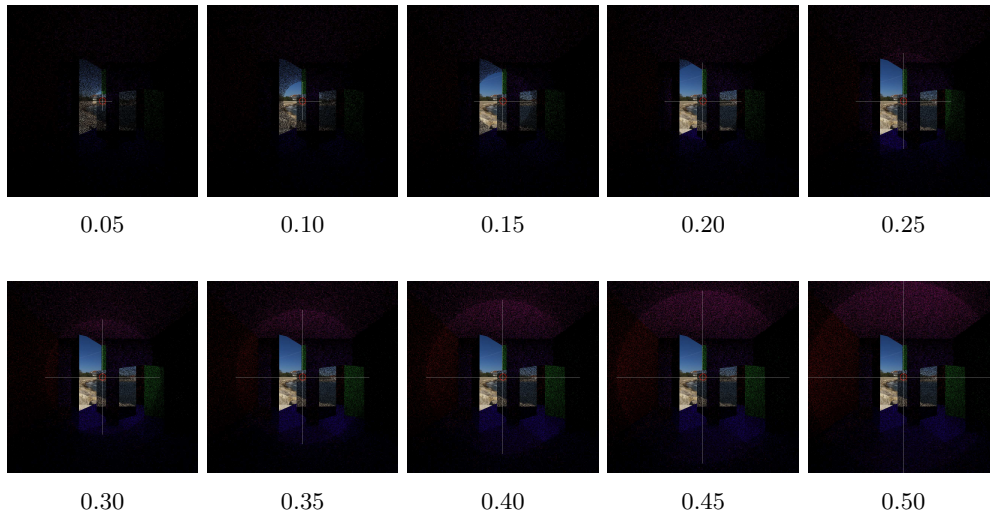


Figure 3.10: Tested RoI sizes for the Cornell Box scene.

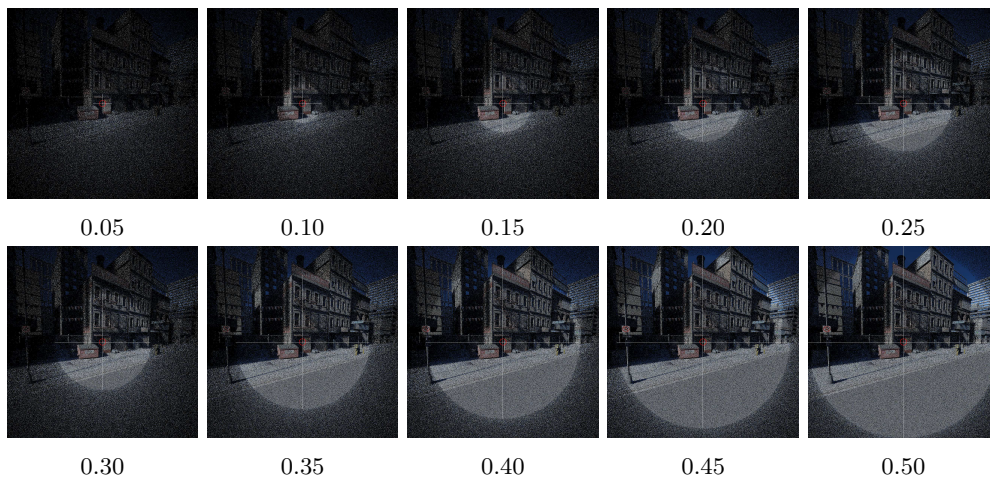


Figure 3.11: Tested RoI sizes for the Urban Sprawl scene

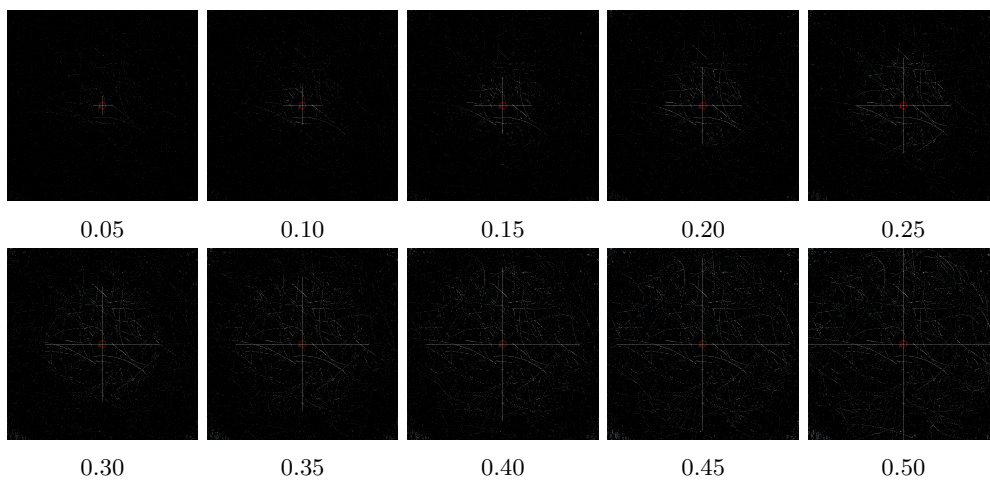


Figure 3.12: Tested RoI sizes for the Hairball scene

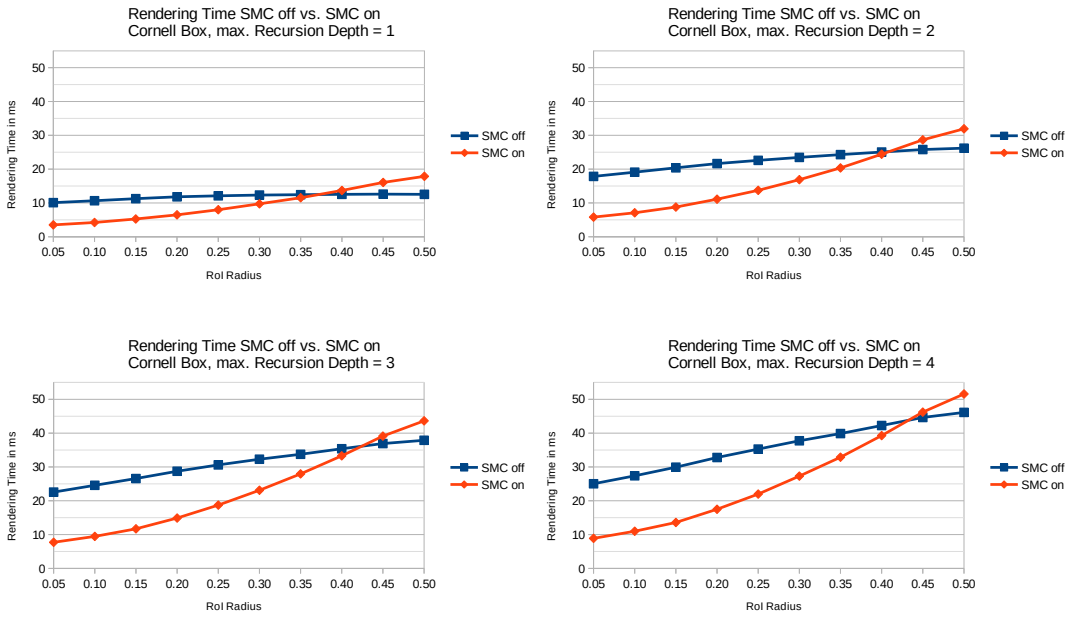


Figure 3.13: Rendering time for the Cornell Box scene with and without SMC-based scheduling and RoI radius between 0.05 and 0.5 times the image resolution (1024×1024). Timings already include the compression step for SMC measurements, which takes between 0.2 and 0.3 milliseconds for all tested configurations.

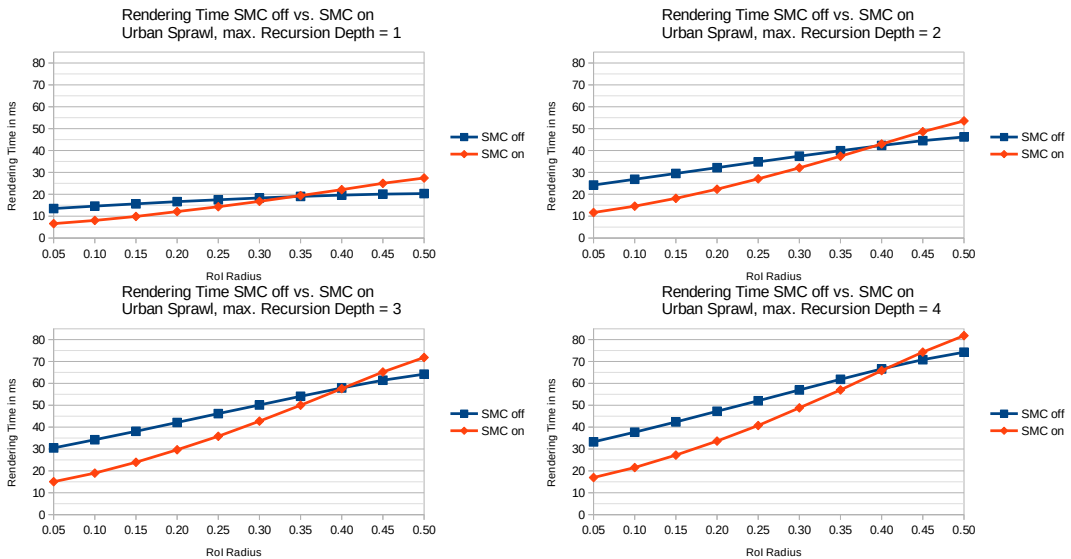


Figure 3.14: Rendering times for the Urban Sprawl scene with and without SMC-based scheduling and RoI radius between 0.05 and 0.5 times the image resolution (1024×1024). Timings already include the compression step for SMC measurements, which takes between 0.2 and 0.3 milliseconds for all tested configurations.

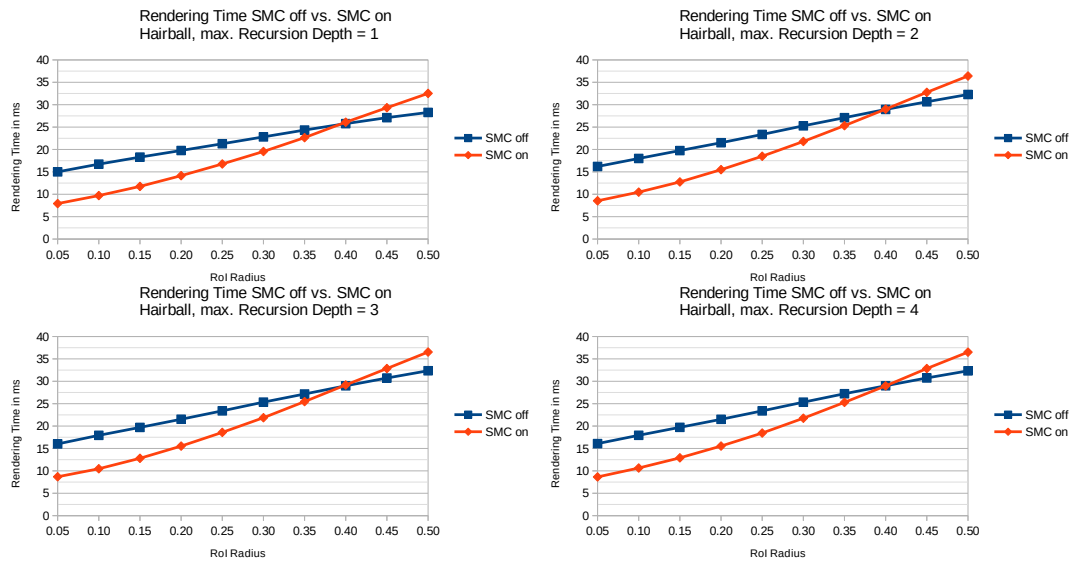


Figure 3.15: Rendering times for the Hairball scene with and without SMC-based scheduling and RoI radius between 0.05 and 0.5 times the image resolution (1024×1024). Timings already include the compression step for SMC measurements, which takes between 0.2 and 0.3 milliseconds for all tested configurations.

depth of 1. However, the advance in rendering time for SMC becomes greater with a higher computational load per pixel. This can be explained by the longer time that each thread on the GPU is running to compute a full path, with threads belonging to unsampled pixels just remaining in an idle state, resulting in a low GPU utilization. For a recursion depth of 4, rendering times with and without SMC are equal between 0.40 and 0.45, which is an increase of 0.1 compared to a recursion depth of 1, or approximately 1.28 times the amount of pixels inside the RoI.

Figure 3.16 gives an idea of the rendering times with SMC-based scheduling relative to rendering times without SMC for varying maximum recursion depths. Benchmarking the scheduling method for varying recursion depths should give us an insight on how a higher computational load per thread influences the rendering performance with and without SMC. Again, it is clearly visible that the advantage of SMC at small RoI radii diminishes with increasing radii. At the same time, this effect is most significant for a maximum recursion depth of 1. However, when analyzing the Hairball scene shown in Figure 3.18, the difference between recursion depths is marginal at most. We attribute this to the far higher amount of geometry compared to the Cornell Box, which is also distributed in a relatively uniform way within the field of view. Also, the camera is placed in front of the hairball, so the primary rays always hit the geometry oriented to the outside. This may lead to a high percentage of rays being terminated early because they hit the background in the second iteration already. Placing the camera inside the hairball leads to the performance behaviour shown in Figure 3.19. On the one hand, the advantage of SMC-based scheduling is slightly higher, but on the other hand, the advantage is consistently higher for the

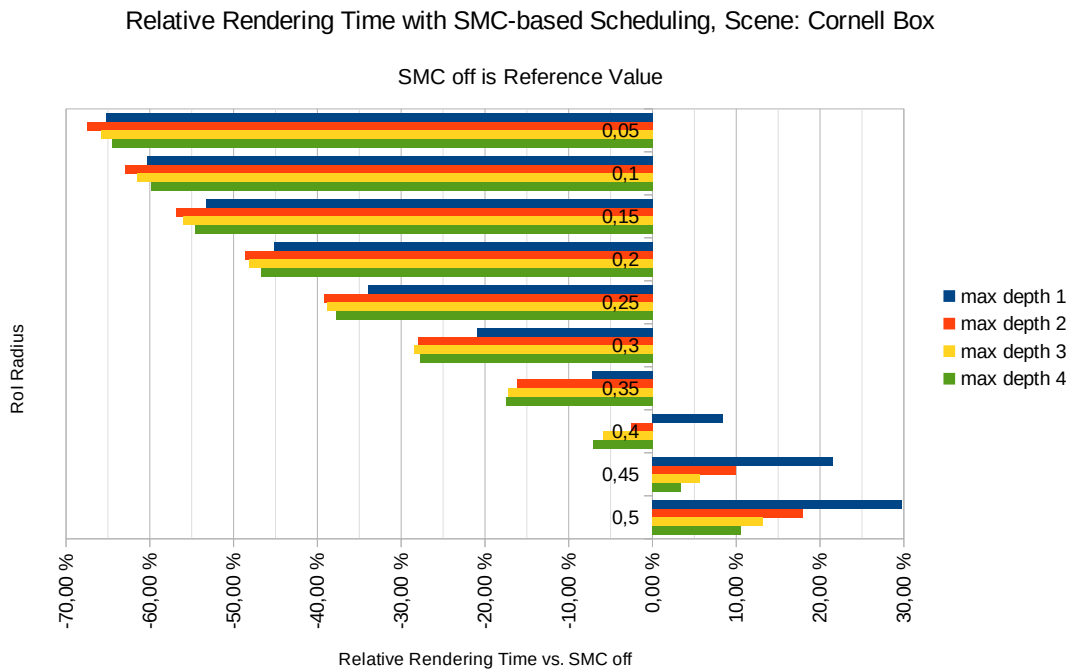


Figure 3.16: Relative rendering times for SMC-based scheduling for the Cornell Box scene, lower is better. Base values are measured with SMC disabled.

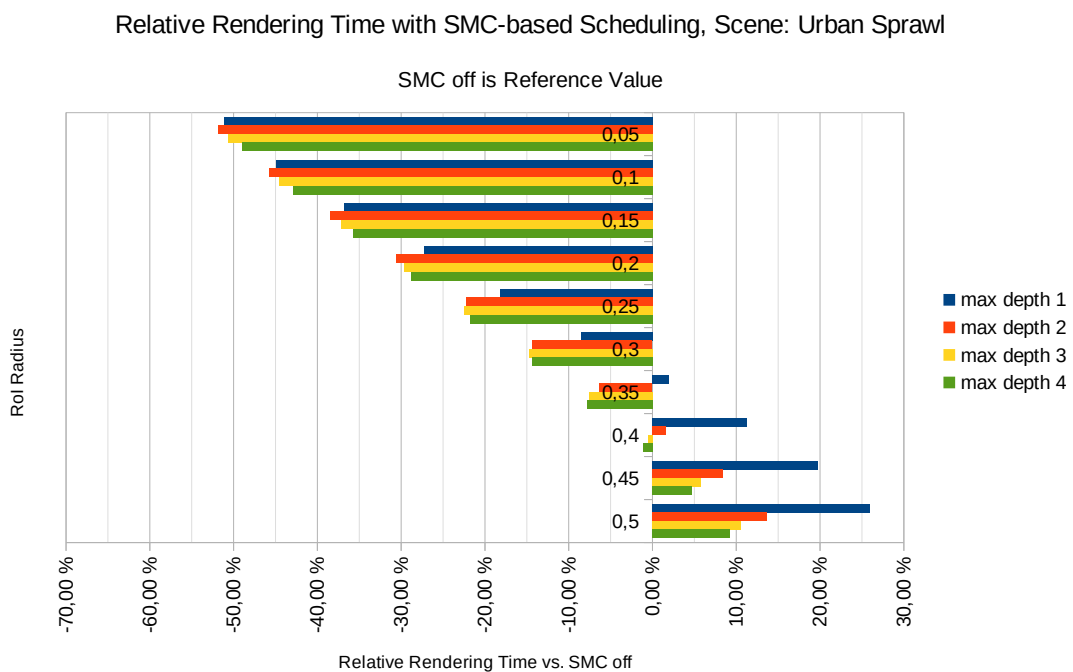


Figure 3.17: Relative rendering times for SMC-based scheduling for the Urban Sprawl scene, lower is better., Base values are measured with SMC disabled.

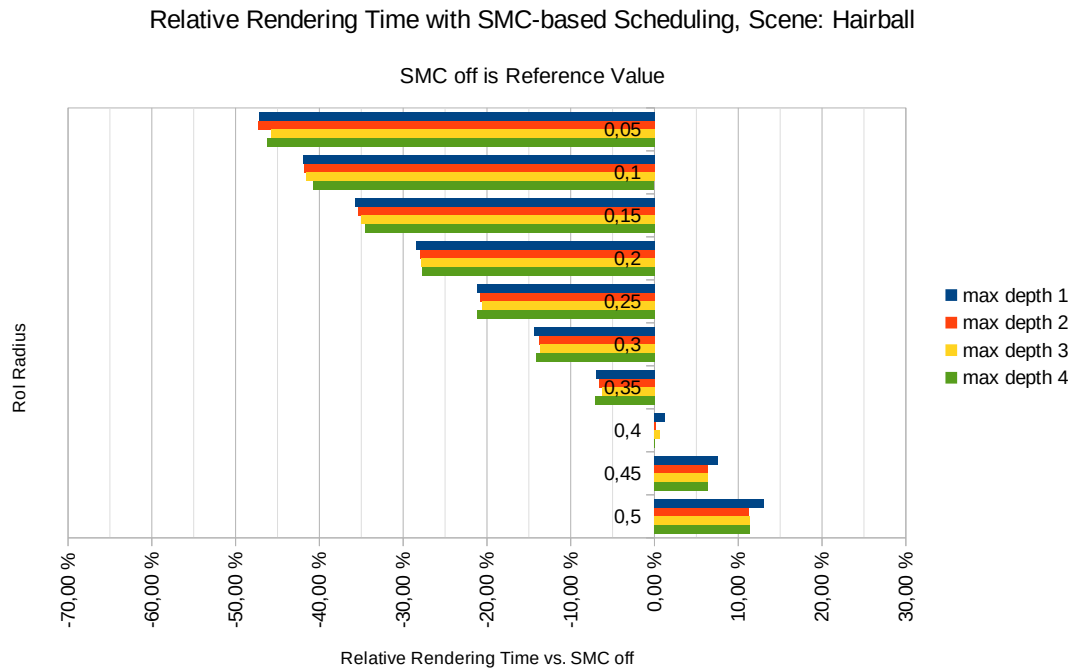


Figure 3.18: Relative rendering times for SMC-based scheduling for the Hairball scene, lower is better. Base values are measured with SMC disabled.

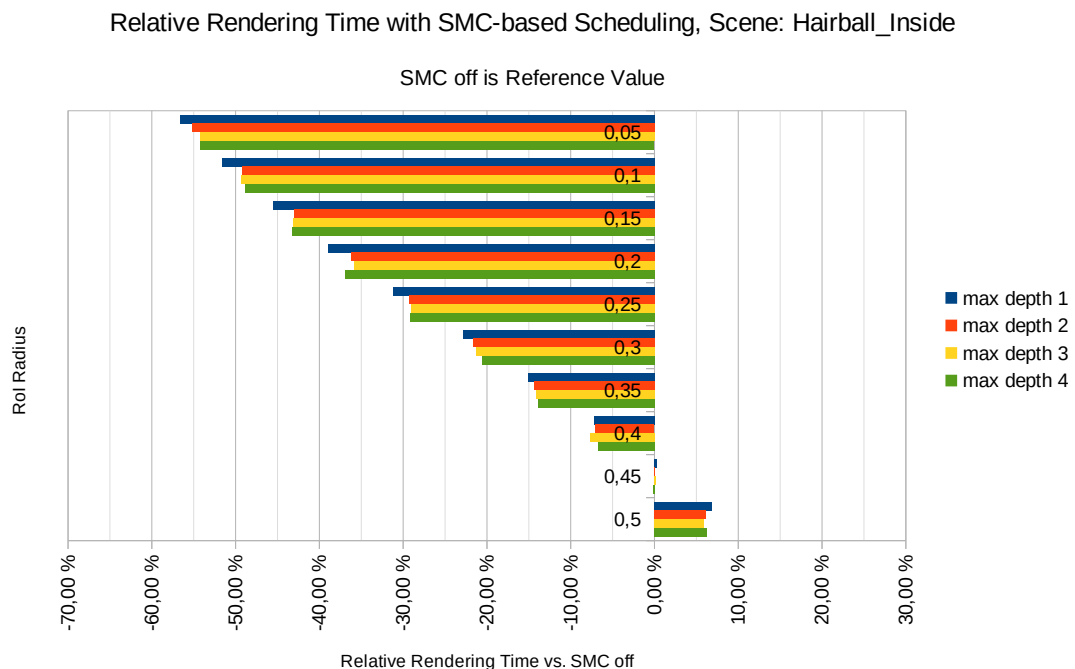


Figure 3.19: Relative rendering times for SMC-based scheduling for the Hairball scene with the camera placed in the middle of the hairball, lower is better. Base values are measured with SMC disabled.

lowest recursion depth. While this seems counterintuitive at first, the differences are pretty small in absolute terms. We attribute this behaviour to an improved ray coherence for primary rays, which degrades with increasing recursion depths. This higher amount of coherence may have a larger influence on rendering times when compared to geometrically simpler scenes like the Cornell Box or even Urban Sprawl, where a large amount of the scene is occupied by relatively simple building models. Figure 3.17 shows the according measurements for the Urban Sprawl scene.

Difference in Convergence Rate

It has to be kept in mind that the rendering time per iteration is a crucial metric, as it enables us to estimate how far an image has converged within the RoI after a certain amount of time. Looking at the numbers for several fields of view, we can observe that for a small RoI the convergence rate in the focused area is very high when compared to uniformly distributed samples. For the Cornell Box scene, with a 4k resolution, a 10° FoV with SMC-based scheduling yields a rendering time of 12.29ms vs. 460ms for full-resolution rendering with uniformly distributed samples. This is a factor of 37.43, corresponding to approximately 84% noise reduction according to MC methods' probabilistic error of $\mathcal{O}(1/\sqrt{n})$. For a 60° FoV, SMC-based scheduling yields a rendering time of 188ms, which still is a factor of 2.44. Note that with the rendering process progressing, the error difference between areas converging at different rates decreases. With a sampling probability of $p_0 = 1$ for pixels inside the RoI and $p_1 \in [0, 1]$ for some pixel outside the RoI, the number of drawn samples after k iterations should be $n_0 = k$ for pixels inside the RoI and $n_1 \approx p_1 k$ for the outside pixel. With the asymptotic error of $\mathcal{O}(1/\sqrt{n})$ the error difference between pixels inside and outside the RoI is in

$$\mathcal{O}\left(\frac{1}{\sqrt{n_1}} - \frac{1}{\sqrt{n_0}}\right) = \mathcal{O}\left(\frac{1}{\sqrt{p_1 n_0}} - \frac{1}{n_0}\right) = \mathcal{O}\left(\frac{1 - \sqrt{p_1}}{\sqrt{p_1 n_0}}\right). \quad (3.14)$$

Figure 3.20 illustrates how these error rates converge with an increasing amount of samples.

3.4 Discussion and Conclusion

In this chapter, we have discussed our approach to guided high-quality rendering, its implementation, and the results of the presented benchmarks and the according implications in the field.

We have shown how using limited RoIs in a path tracing process can increase per-

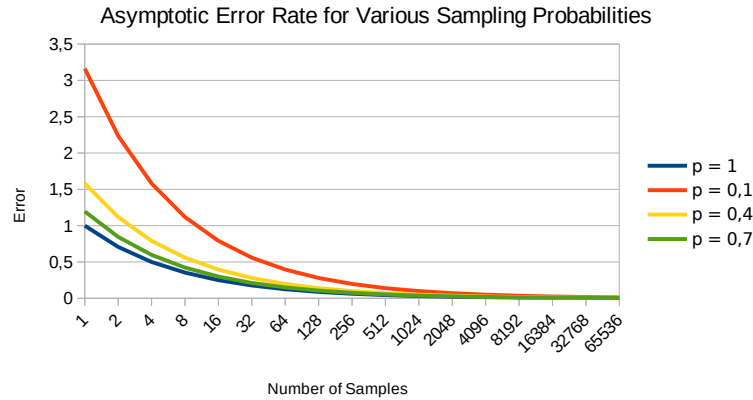


Figure 3.20: Asymptotic error rate for various sampling probabilities

formance and local quality in global illumination rendering with a focus on, but not limited to large 4k projection systems. Moreover, we presented an approach to schedule the rendering tasks guided by an RoI on GPUs, which has proven to lower rendering times compared to standard scheduling in the benchmarks presented above.

Additionally, we have developed filtering methods that help to cope with the sparse sampling in images outside the RoI by improving the visual context early in the rendering process. To interpret the results correctly, it is important to note that our work is based on the assumption that there actually is an important area in the image which is desired to be rendered at a high quality, while the quality of the remaining image is not as important. However, the visual context that is offered by the image area outside the RoI is preserved. We achieved our goal by improving performance for such cases, as shown in the benchmark section, where we also compared the results with standard full-resolution path tracing.

Similar to the adaptation of the sampling process to specific image parts developed in this chapter, an adaptation to the user's perception tailored towards interactive exploration and the use of a head-mounted display (HMD) is presented in the next chapter.

Chapter 4

Foveated Ray Tracing and Eye Tracking Data

Part of the work included in this chapter was previously published in (Weier, Roth, et al. 2016), co-authored by the author of this thesis, who made substantial contributions to the conception, implementation and writing, and was responsible for designing the user study and performing the according result/data analysis. A more detailed description of the recorded tracking data has been published in (Roth, Weier, et al. 2016; Roth, Weier, et al. 2017), lead-authored by the author of this thesis.

When using head-mounted displays with high resolutions for virtual reality applications, high frame rates and low-latency rendering become an absolute necessity, which is challenging for any rendering approach. In addition to its ability of generating realistic images, ray tracing offers a number of distinct advantages, but has been held back mainly by its performance on the available hardware so far, with *NVIDIA RTX* and Microsoft's *DirectX Raytracing* only being available since 2018, which is two years after the publication of our work. In this chapter, we present an approach that significantly improves image generation performance of ray tracing, which is achieved by combining foveated rendering based on eye tracking with repro-



Small foveal region with ($r_0 = 5^\circ$, $r_1 = 10^\circ$, $p_{\min} = 0.01$)

(a) Small foveal region with ($r_0 = 10^\circ$, $r_1 = 20^\circ$, $p_{\min} = 0.05$)

(b) Full Renderer

Figure 4.1: Images generated with our foveated renderer showing the effect of different configurations for the foveal region, including an image that was rendered by ray tracing every pixel.

jection rendering using previous frames in order to drastically reduce the number of required samples per frame. To reproject samples, a coarse geometry is constructed from a G-buffer. Possible errors introduced by this reprojection as well as parts that are critical to perception are scheduled for resampling. Furthermore, a coarse color buffer provides an initial image, which is refined smoothly by additional samples where needed. Evaluations and user tests show that our method achieves real-time frame rates, while visual differences compared to fully rendered images are hardly perceivable. As a result, we can ray trace non-trivial static scenes for the Oculus Rift DK2 HMD at a 1182×1464 resolution per eye within the VSync limits without clearly perceived visual differences.

4.1 Introduction

Presenting a virtual environment to the user in a way that is so compelling and convincing that the sense of presence becomes almost indistinguishable from the real world is a long-envisioned dream in the field of virtual reality (VR). Due to the increased availability of high-quality head-mounted displays (HMDs) with a wide field of view (FoV) in recent years, interest has been drawn towards the field of virtual and reality for researchers and consumers alike. Starting with the *Forté VFX1* at 263×230 pixels per eye, pixel densities have significantly increased over the last two decades: current devices such as the *Pimax 8k* go up to a 4k resolution (3840×2160 pixels) per eye. Figure 4.2 gives an overview of the evolution of HMD resolutions, including a linear trend.

Despite the increasing pixel densities already available, providing the highest possible visual quality to the user would require an HMD to match the full retinal resolution, leading to a resolution of approximately $32,000 \times 24,000 = 768,000,000$ pixels for the human eye's full dynamic field of view according to Hunt (2015). While our own work on this subject is from 2016, rendering at such resolutions interactively is still untractable by current and foreseeable hardware and software solutions. Besides rendering convincing imagery for high pixel densities, high update rates are crucial for limiting potential motion sickness (Hale and Stanney 2014, p. 541), posing a major challenge when bringing image synthesis algorithms to HMDs.

Fortunately, as described in Section 2.3.1, the human visual system (HVS) underlies several limitations which imply that this challenge can be approached with foveated and gaze-contingent rendering methods. By omitting largely imperceptible details in the visual periphery, techniques from this area exploit the limitations of the human eye, as only our central vision affords high visual acuity. Thus, we may degrade rendering quality with increasing eccentricity. In such scenarios, the point a user is currently looking at in image space is referred to as the point of regard (PoR).

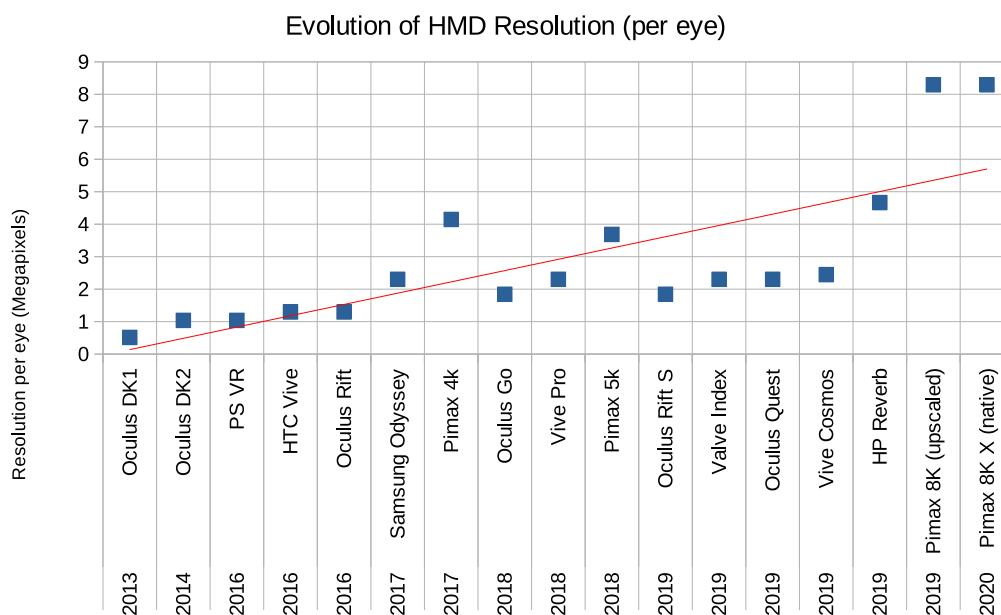


Figure 4.2: The evolution of resolutions in modern HMDs since the release of the Oculus Rift DK1

The parts of the eye responsible for central vision consist of the fovea (approximately 5° in diameter around the central optical axis), the parafovea (approximately 5° to 9°) and the perifovea (approximately 9° to 17°). Peripheral vision refers to the area outside of central vision (Wandell 1995). While the fovea itself affords high visual acuity, this rapidly drops with an increasing distance due to the cone density's hyperbolic falloff. In contrast, the number of rods is decreasing slower, a property that leads to a high sensitivity to spatiotemporal brightness and contrast changes in peripheral vision. This has to be accounted for when developing new techniques, as noted by Legge and Kersten (1987), Geisler and Perry (1998), and Murphy and Duchowski (2007). As Lou, Migotina, et al. (2012) describe, colors, patterns and shapes also have a significant influence on perception, just like motion, as suggested by McKee and Nakayama (1984). Furthermore, visual attention has been shown to affect perception through effects such as visual tunneling, where details outside the area of fixation are largely ignored. This effect has already been described by Miura (1986).

While at the time of our work rendering for HMDs was mainly based on rasterization, it was already clear that ray tracing has several advantages when it comes to stereo rendering, wide FoVs and low-latency rendering (Hunt 2015). Yet, ray tracing has been mainly held back by the required computational effort, which makes it challenging to achieve the same performance as rasterization without specific hardware acceleration. However, in recent years, new techniques became available that provide hardware support for ray tracing, for example through NVIDIA's Turing

and Ampere architectures and the according RTX graphics cards. In this section, we mainly focus on the advantage of ray tracing being able to support the free distribution of samples on the image plane.

We introduce a novel ray-tracing-based foveated rendering system capable of rendering high-quality images at frame rates that are sufficient for modern HMDs running at their native refresh rate. The sample density is reduced by adapting the ray generation to the foveal receptor density. To improve perception in foveated rendering methods, it is necessary to limit the detection of visual artifacts. Using our approach, missing information from the sampling process can either be reconstructed by using a *support image* that is guaranteed to sample the full scene using a lower uniform resolution or by reprojecting the previous frame to improve the quality of the final image.

Note that our method does not handle view-dependent effects like reflections or refractive transmissions explicitly. Rendering such effects using reprojection-based methods remains a challenge; however, it is slightly alleviated by high frame rates and the correspondingly smaller visual differences between frames. While the quality of such effects may suffer due to the reprojection process, the influence on the perceived quality in a system like ours still has to be analysed. Nonetheless, our benchmarks demonstrate the high performance of our implementation when compared to standard ray tracing.

We also conducted a user study using an Oculus Rift DK2 equipped with an eye tracker. This made it possible to substantiate the claimed visual quality provided by our method for static scenes and revealed a significant influence of the foveal region's size, with some observable effects likely caused by visual attention. The latter can greatly support foveated rendering performance, as fewer samples need to be generated in the periphery when users concentrate on a specific part of the scene. Amongst others, the analysis of the recorded eye tracking data revealed an interesting relation between fixation accuracy and quality ratings for different fixation modes that seems counterintuitive at first.

4.2 System Description

In this section we describe the building blocks of our foveated rendering system as well as the setup of the conducted user study. An overview of the entire pipeline is shown in Figure 4.3. The system's core is a fast ray tracer based on NVIDIA CUDA, using an SBVH acceleration structure (Stich, Friedrich, et al. 2009) (Figure 4.3, Block 2). As shown in Section 4.2.1, the generated sampling pattern depends on three parameters describing a foveal region to account for the user's perception

Buffer	Description
Reprojected Image	Reprojected color image for computing the current frame
Reprojected Cache Info	Reprojected weights for accumulating new samples
Final Image	Current frame's color data
Final Cache Info	Current frame's weights
Resampling Info	Used for marking parts that need additional sampling
Support Image	Low-resolution color buffer, used for filling in gaps
Support G-Buffer	Low-resolution, required to generate a coarse mesh for reprojection

Table 4.1: An overview of the buffers used in the reprojection and accumulation process

and gaze. The foveated sampling process results in a sparse image. Thus, increasing eccentricities lead to proportionally larger gaps between sampled pixels. Presenting such a sparsely sampled image to the user would not meet the perceptual requirements, as the gaps would result in the sparse image's brightness being vastly different from the fully sampled image. Also, strong temporal flickering would result from the stochastic nature of the sampling process. This makes it necessary to provide a method for improving image quality outside the foveal region and generating a smooth image from the available samples. Certain requirements have to be met by this method: Performance has to suffice the time constraints necessary to stay within VSync limits (13.3ms at 75Hz), and image quality should not exhibit artifacts that are disturbing to human perception.

To reduce the visibility of artifacts, the ray tracer is coupled to a reprojection scheme (Figure 4.3, Block 1) that provides additional information to improve image quality. This is done by reconstructing a coarse depth mesh from the scene and rendering it from the next frame's perspective to reproject samples, as illustrated in Section 4.2.2. Subsequently, poorly reprojected image parts and regions critical to peripheral vision are detected, as shown in Section 4.2.3. These are marked for resampling in an auxiliary buffer referred to as *resampling info*.

Eventually, information is stored inside a cache and a final image is computed (Figure 4.3, Block 3). Missing samples can either be reconstructed from the reprojected previous frame or from the current frame's low-resolution color and G-buffers (referred to as *support image* and *support G-buffer*), as these are updated in each frame (see Section 4.2.4). These buffers' resolution is a user-defineable fraction of the target resolution required for the HMD. The support image contains a regular low-resolution color image, while the support G-buffer contains the geometric normals and depth values that are later used to reconstruct the coarse geometry for the next frame. An optional post-processing step (Figure 4.3, Block 4) can further improve image quality when stochastic sampling processes are used (see Section 4.2.5). Each of the pipeline's steps will be described in the following sections, while an overview of the buffers used in the process is given in Table 4.1.

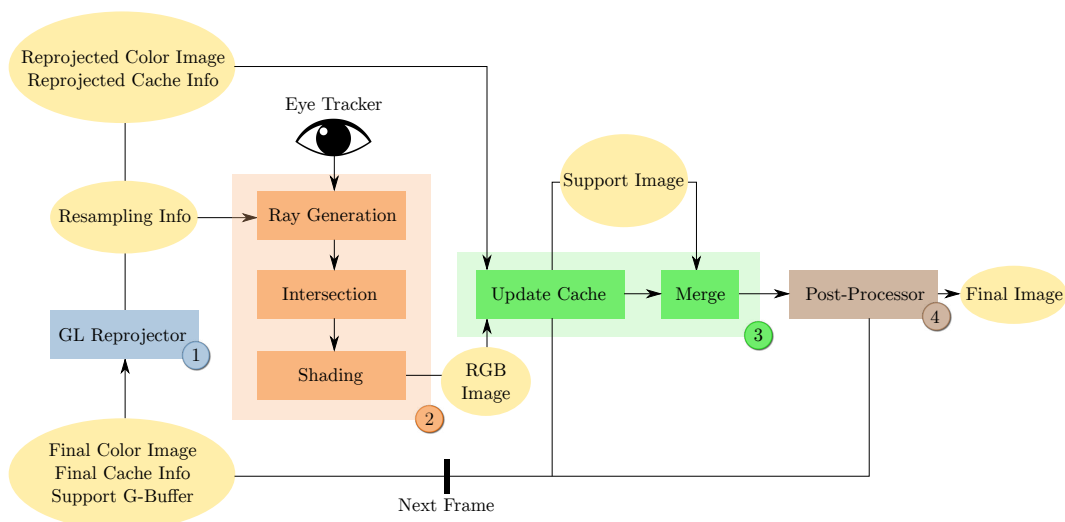


Figure 4.3: Building blocks of our reprojection pipeline. Old *color* and auxiliary *cache info* buffers are reprojected using a *GL Reprojector* (Block 1). New rays are generated based on the user’s gaze and possible errors introduced by the reprojection are marked in a *resampling info* buffer. The ray traced pixel values (Block 2) are blended using a temporal caching and merging scheme (Block 3). An optional *Post-Processor* is used to reduce artifacts arising from stochastic sampling processes like ambient occlusion (Block 4).

4.2.1 Ray Generation and Ray Tracing

When the rendering process is started, a ray generation kernel creates rays by sampling the image plane according to a foveal falloff function. Additional rays are generated for all pixels marked in the resampling info or belonging to the support image. The generated rays are intersected with the scene geometry, resulting in a list of hit points and their respective pixel indices. We employ Aila, Laine, et al. (2012)’s kernel for ray traversal, extended to speed up the evaluation of fully transparent alpha values in the innermost loop.

After computing the intersections, the hit points are shaded in another CUDA kernel. Shading supports Phong lighting with mipmapping, ambient occlusion, point and area light sources. Besides computing the shaded pixel color, the irradiance from the area light sources and an ambient occlusion factor for diffuse surfaces are stored in a separate *light buffer*, allowing for adapting the running estimate to different rates for ambient lighting and point lights on the one hand, and area light sources and ambient occlusion on the other hand. Thus, it becomes possible to reduce the noise introduced by sampling area light sources or ambient occlusion without touching ambient and point light information. Furthermore, the stored data can be spatially reused between neighboring pixels (see Section 4.2.5).

With increasing eccentricities, visual acuity is subject to a hyperbolic falloff (Guenther, Finch, et al. 2012; Strasburger, Rentschler, et al. 2011). However, we chose to use

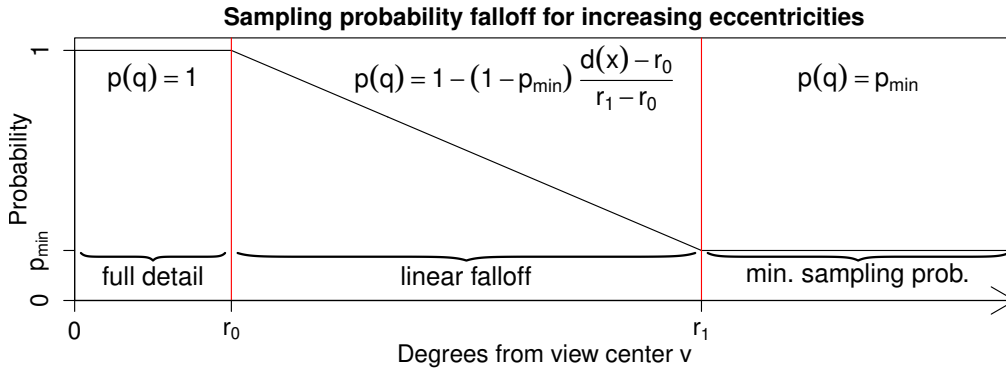


Figure 4.4: Probability $p(q)$ for sampling a specific pixel q based on its eccentricity and user-defined parameters r_0 , r_1 and p_{\min} . Despite the hyperbolic falloff of cones towards outer regions, a linear falloff is employed to improve motion perception in the periphery and to reduce spatial and temporal aliasing artifacts in these areas.

linear falloff model for determining sampling probabilities, as increasing the sampling rate compared to a hyperbolic falloff supports peripheral motion perception. Additionally, a higher sampling rate reduces spatial as well as temporal aliasing artifacts in general, alleviating the challenge posed by the human eye’s high flickering sensitivity at larger eccentricities. Moreover, a linear model matches visual acuity well for small angles (Guenther, Finch, et al. 2012). We refer to this model as the *foveal falloff function*, which is illustrated in Figure 4.4.

To achieve a linear behavior, ray generation is based on two user-defined eccentricity thresholds: An inner threshold r_0 and an outer threshold r_1 , both given in degrees in the visual field. r_0 determines the size of the foveal region, i.e., the area rendered at full detail, while r_1 together with the minimum sampling probability p_{\min} determines the probability falloff beyond r_0 . All three values together are referred to as a foveal region configuration (FRC). Pixels with a larger eccentricity than r_1 are only sampled with a probability of p_{\min} . Generally, pixels are only queued for sampling if $\xi_q \leq p(q)$, with $\xi_q \sim U(0, 1)$. If a pixel is required for the support image, it is always scheduled for sampling. It is also sampled if it has been marked in the resampling info before.

4.2.2 Reprojection

After computing an array of newly shaded samples along with the pixel indices for the current frame, our reconstruction approach relies on information from the previous frame. As computing more samples is expensive, reusing information from previous frames helps to increase performance while it also supports image quality. To create a perspective-correct reprojection of information from the previous frame,

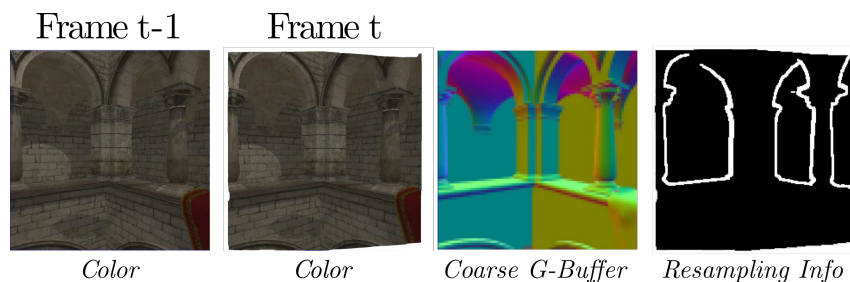


Figure 4.5: Reprojection from frame $t - 1$ to frame t . The perspective change is especially visible at the image borders and also affects discontinuities caused by disocclusions. The coarse G-buffer enables us to derive the resampling info, which serves as a guide for where to place additional samples to resolve visible artifacts.

the scene geometry must be taken into account. Possible approaches include the reprojection methods in (Simmons and Séquin 2000; Tole, Pellacini, et al. 2002), constructing and updating an irregular mesh with potentially as many vertices as pixels at the highest quality level. However, such approaches are costly, while the pure ray tracing step is fast on the utilized hardware, so that expensive methods do not pay off.

Hence, we decided to use a reprojection strategy based on a coarse uniform mesh. Reprojection errors resulting from this geometry are resolved by computing additional samples for these regions instead of constructing a more precise mesh representation. The reprojection process (Figure 4.3, Block 1) transforms the final color image along with the final cache info. The latter is used to keep a state in the buffer that maintains how samples should be combined temporally. As described in Section 4.2.4, this buffer is a `float4` texture that is reprojected along with the color information. For each frame, a uniform mesh is generated from the support G-buffer by creating and displacing a uniform grid of vertices matching the support G-buffer’s resolution.

To reconstruct the scene geometry from the depth information stored in the support G-buffer, the vertices are adjusted to the according image space depth values with a geometry shader. Subsequently, an “unprojecting” step is performed using the previous frame’s model-view-projection matrices (MVP Matrices). This yields the vertex positions in world space, representing the surface of the visible scene geometry from the previous frame. In the next step the vertices are projected to the new frame using the current MVP Matrices. This mesh is rasterized at the full rendering resolution, textured with the last frame’s final color image, eventually yielding the reprojected version of the previous frame. Figure 4.5 shows the reprojection from frame $t - 1$ to frame t .

Due to the change in perspective, each pixel’s footprint may cover a couple of texels of the previous frame’s final color image. Therefore, special care has to be taken

when filtering this texture during rendering. As computing a mipmap hierarchy along with anisotropic filtering for the reprojected texture per frame based on the new footprint is too expensive, another texture filtering method has to be used. We chose to randomly sample the pixel footprint multiple times using a normal distribution inside the fragment shader to compute the final reprojected color.

4.2.3 Handling Reprojection Errors

Because the uniform mesh employed for reprojection is not a perfect representation of the actual scene geometry, perceptible errors may be present, possibly caused by geometry newly entering the view frustum, disocclusions, and undersampling (Mark, McMillan, et al. 1997). If a scene part has not been inside the view frustum in the previous frame and sampling has not been triggered after evaluating the foveal falloff function, missing pixels are reconstructed from the coarse support image (see Section 4.2.4). As shown in Figure 4.1, disocclusions and undersampling can both cause strong visual artifacts to appear in the image. This is caused by incorrectly or incompletely reprojected information. The coarse sampling performed for the support image adds newly computed samples on top, which may also add noise. Therefore, we try to detect and create additional samples for those areas, consequently improving perceived image quality.

First, to detect regions that need further sampling, the scene is rendered using the coarse resolution matching the support image and support G-buffer using the reprojection procedure described in Section 4.2.2. If there is a depth or luminance difference between a pixel and its direct neighborhood in the reprojected image larger than a user-defined threshold ϵ_{depth} or ϵ_{lum} , a pixel is marked for resampling in the resampling info. This process resembles methods like SMAA, presented by Jimenez, Echevarria, et al. (2012).

Depending on the value chosen for ϵ , geometry that does not resemble the scene may be used for reprojection anyway, e.g., in case of relatively flat objects in front of a wall. If such geometry is looked at frontally in frame $t - 1$, moving the camera in frame t can result in undersampling artifacts because the possibly wrongly closed geometry is reconstructed, connecting the object to the wall. These objects might expose depth and luminance distances well below the respective ϵ -thresholds, while the closed geometry resulting from the reprojection process is actually wrong (Mark, McMillan, et al. 1997). Such surfaces occur along the user’s viewing direction, i.e., the angle between the surface normal and the observer is close to 90° . We detect such artifacts with an additional test looking at the surface geometry. From the previous frame’s geometry normal \vec{n} and camera orientation \vec{d} , we compute $e_t = \max\{\vec{n} \cdot -\vec{d}, 0\}$. If $e_t < \epsilon$, the pixel is marked for resampling. Partial derivatives of

texture coordinates would be another measure to detect regions that need additional sampling. They yield information about an observer’s angle towards a potentially undersampled surface. However, we have not found noticeable visual enhancements by using this information, as head movements are limited when wearing an HMD. As illustrated in Figure 4.1, in case of complex geometry, a large part of the image may be covered by possibly undetected and thus undersampled edges. This necessitates a measure for sample quality accounting for a sample’s age, as presented in the next section.

4.2.4 Cache Update and Merging

At this point, the current image consists of the previous frame’s reprojected color image. Newly shaded samples from the ray tracing process have to be combined with this cache image using a temporal blending method. This accumulation process should be designed in a way that reduces the weight of older samples, as simply accumulating samples with equal weights does not make sense for two reasons: First, due to the sparse sampling process, each pixel may have been sampled last at a different point in time. Second, assigning a high weight to old samples leads to visual artifacts like smearing on edges. However, at the same time just using the new sample without considering cached values can lead to disturbing temporal noise, especially because of the human eye’s high peripheral flickering sensitivity. This is caused by correctly reprojected regions having temporally varying color values due to the stochastic sampling process. To overcome this, we chose to apply a smooth temporal blending process with a limit to the samples’ age. This age is used to assign higher weights to samples that should be combined with old samples. While such a process reduces temporal flickering, large-scale contrast for the visual periphery is preserved.

A sample’s color is directly written to the output if it belongs to the central foveal region, is part of the resampling process (marked in the resampling info), or is written to a part of the image that did not contain any reprojected color due to disocclusion or movement. Moreover, the support image and support G-buffer are stored as well, as shown in Figure 4.3. For all other samples, we employ Yang, Nehab, et al. (2009)’s method for amortized supersampling. This is the foundation which current temporal antialiasing methods have been derived from due to its robustness and overall quality. The approach is described shortly below.

Geometry is accounted for by considering the depth difference between $\pi_{t-1}(p)$ (reprojected cached position for the current pixel) and p (the new sample position at time t). If this value is above a threshold ϵ , it is assumed that the reprojection contains an error, as the ray has hit a part of the scene different from the cache.

Therefore, we take the newly generated sample without considering anything from the cache. If the depth difference is below the threshold, $f_{t-1}(\pi_{t-1}(p))$, which is the color information from the previous frame, can be combined with $s_t(p)$. The new pixel value $f_t(p)$ is now computed using a blending value $\alpha_t(p)$:

$$\begin{aligned} f_t(p) &= \alpha_t(p)s_t(p) + (1 - \alpha_t(p))f_{t-1}(\pi_{t-1}(p)) \\ \alpha_t(p) &= \frac{1}{N_{t-1}(p) + 1} \\ N_t(p) &= N_{t-1}(p) + 1, \end{aligned} \quad (4.1)$$

where $\alpha_t(p), N_t(p) = 1$ when p becomes visible for the first time.

To account for the issues caused by scene motion mentioned above, $\alpha_t(p)$ has to be adjusted according to the number of samples accumulated for the current pixel as well as this pixel's most recent update-time. To consider the number of accumulated samples, the update rule for $N_t(p)$ is changed as follows:

$$N_t(p) = \left(\alpha_t(p)^2 + \frac{(1 - \alpha_t(p))}{N_{t-1}(\pi_{t-1}(p))} \right)^{-1} \quad (4.2)$$

To avoid infinite accumulation of samples, we finally compute the weight $\alpha'_t(p)$ as $\alpha'_t(p) = \max\{\alpha_t, k\}$, with k being the minimum possible weight for the new sample.

In contrast to Yang, Nehab, et al. (2009), we do not sample each pixel per frame. Therefore, it is best to adapt k dynamically based on a sample's age. If a pixel has been sampled a couple of frames ago, it has undergone the potentially imprecise reprojection process multiple times, especially since the camera is constantly moving when head tracking is active. If the time span between the previous update and the current time is large, it is better to account for the current sample with a higher weight. Thus, instead of a fixed k , we use the following function:

$$\begin{aligned} k_t(\Delta t) &= \min \left\{ \exp \left(x_0 + \frac{\Delta t - 1}{t_{\max} - 1} (x_1 - x_0) \right), k_{\max} \right\}, \\ x_0 &= \ln k_{\min}, \\ x_1 &= \ln k_{\max}. \end{aligned} \quad (4.3)$$

It can be parameterized based on a fixed interval $[k_{\min}, k_{\max}]$ and the maximum time span t_{\max} we allow for accumulating samples. $\Delta t = t - t_{\text{touched}}$ is the difference of the current frame index and the frame index a value has last been updated in the cache. t_{\max} is the user specified maximum number of frames between two samples. Computing k this way poses a trade-off: t_{\max} should be chosen according to the refresh rate and in a way that resolves possible artifacts as early as possible by giving the new sample a higher weight. At the same time, weighting older samples relatively

high guarantees a smooth temporal transition and reduces flickering. An additional cache information buffer is used to keep track of $\alpha_t(p)$, N_t , and t_{touched} , stored per pixel along with the color image. However, to have these estimates available in the next frame, it is necessary to reproject this buffer to the new perspective the same way as it is done for the color values described in Section 4.2.2. Eventually, another kernel is launched to merge the images with the support image (see Figure 4.3, Block 3). Since both the reprojection process and the foveated sampling can fail for parts that have not been in the view frustum for frame $t - 1$, the support image is used to fill in all missing information.

4.2.5 Post-Processing

When rendering scenes with stochastic sampling processes (e.g., for area lights or ambient occlusion), the resampling process presented above leads to a discrepancy in the state of convergence for reconstructed and resampled image regions, as the latter do not consider any cache information. This may cause a visual difference in these areas, appearing as high-frequency temporal flickering mostly caused by the stochastic processes. An optional post-processing filter (see Figure 4.3, Block 4) can be applied to resampled regions marked in the resampling info to reduce noise. For each pixel q in such a region, the nearest reconstructed (i.e., non-resampled) neighbor along the horizontal and vertical axis on the image plane is searched. The distance to this neighbor is then used to create a search window which is randomly sampled n times. Subsequently, the closest reconstructed pixel r found during the sampling step is selected and the information stored in the light buffer for r is applied to the noisy pixel q .

4.3 Experimental Evaluation: Benchmarks

All benchmarks have been done on a system equipped with a Intel Core i7-3820 CPU, 64GiB of RAM and an NVIDIA GeForce Titan X driving an Oculus Rift DK2. Using the Oculus SDK, we determined the FoV for a single eye and computed the projection matrix. Rendering was done at a resolution of 1182×1464 pixels per eye. Table 4.2 lists the benchmark results of fly-throughs with 1000 frames each. We decided to use the parameters ($r_0 = 10^\circ$, $r_1 = 20^\circ$, $p_{\text{min}} = 0.05$) to configure the foveal region. As shown in Section 4.4, users were mostly unable to detect any visual differences compared to the full renderer for this FRC. For the benchmark process, the foveal region was statically positioned at the image center. A resolution of 256×318 was empirically chosen for the support image and support G-buffer, as it provided a good trade-off between speed and quality for the utilized HMD and scenes. The

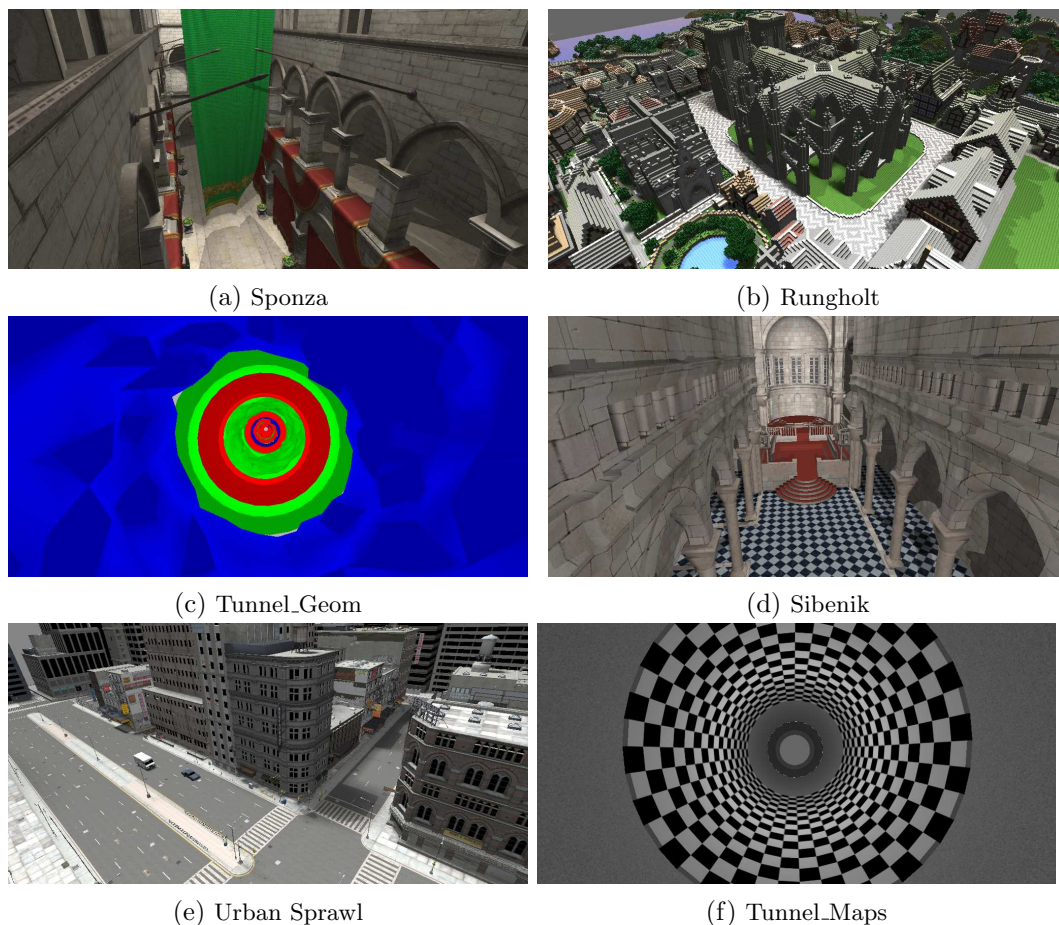


Figure 4.6: Scenes used for benchmarks and user studies of our implementation

four following test scenes were used: *Sibenik*, *Sponza*, *Rungholt*, *Urban Sprawl* (see Figure 4.6). The choice of scenes was mainly driven by perceptual considerations and is elaborated in Section 4.4.1. Each of the scenes was rendered with one point light source, an area light source with 8 samples per pixel (SPP), and ambient occlusion using 16 SPP in order to include different illumination complexities.

Table 4.2 and Figure 4.7 show that the speedup of our foveated ray tracer compared to a full ray tracer scales well with increasing workloads, as it reduces the number of rays. Hence, the smallest speedup of 1.46 is achieved for rendering the scene *Urban Sprawl* with a single point light, while the maximum speedup of 4.18 is achieved for *Sponza* with ambient occlusion. It also shows the time required for reprojection, cache update, merging and the post-processing step, where the latter only has a minuscule impact on performance for all scenarios, The influence of the FRC on the rendering performance measured in frames per second (FPS) for *Sponza* is illustrated in Figure 4.8.

Even though rasterization is inherently different from ray tracing, we provide a few numbers for comparison to state-of-the-art approaches employing this method.

Scene	Type	# of rays (Millions)		Ray Tracing						Total Time		Speed up
		full	ours	Repro-ject	full	ours	Cache Update	Merge	Post	full	ours	
Sibenik 75K Tris	Point Light	3.46	0.81	1.41	10.74	3.85	0.64	0.84	0.00	10.74	6.74	1.59
	Area Light	15.57	3.64	1.51	47.96	14.80	0.67	0.85	0.33	47.96	18.36	2.61
	AO	29.42	6.88	1.45	94.64	27.96	0.64	0.83	0.32	94.64	31.39	3.02
Sponza 154K Tris	Point Light	3.46	0.64	1.41	13.93	4.43	0.58	0.84	0.00	13.93	7.26	1.92
	Area Light	15.54	2.89	1.44	81.61	20.07	0.58	0.83	0.24	81.61	23.17	3.52
	AO	29.36	5.45	1.43	179.01	39.79	0.58	0.83	0.24	179.01	42.87	4.18
Rungholt 6704K Tris	Point Light	2.90	0.58	1.38	9.51	3.54	0.59	0.83	0.00	9.51	6.34	1.50
	Area Light	11.10	2.25	1.38	34.20	9.98	0.59	0.83	0.19	34.20	12.97	2.64
	AO	20.47	4.17	1.39	168.03	51.15	0.59	0.83	0.19	168.03	54.14	3.10
USprawl 773K Tris	Point Light	3.06	0.64	1.37	8.97	3.33	0.60	0.83	0.00	8.97	6.12	1.46
	Area Light	12.34	2.60	1.36	29.67	8.68	0.60	0.83	0.28	29.67	11.75	2.52
	AO	22.95	4.85	1.39	110.11	30.02	0.60	0.83	0.29	110.11	33.13	3.32

Table 4.2: Times in milliseconds for each stage of the pipeline in comparison to a full renderer showing the speedup of our approach. Times and speedups are computed for a medium-sized foveal region with ($r_0 = 10^\circ, r_1 = 20^\circ, p_{\min} = 0.05$) for a single eye with a resolution of 1182×1464 pixels and no oversampling on an NVIDIA GeForce Titan X. For the chosen foveal region, users were mostly unable to detect any visual difference to full rendering in the user study.

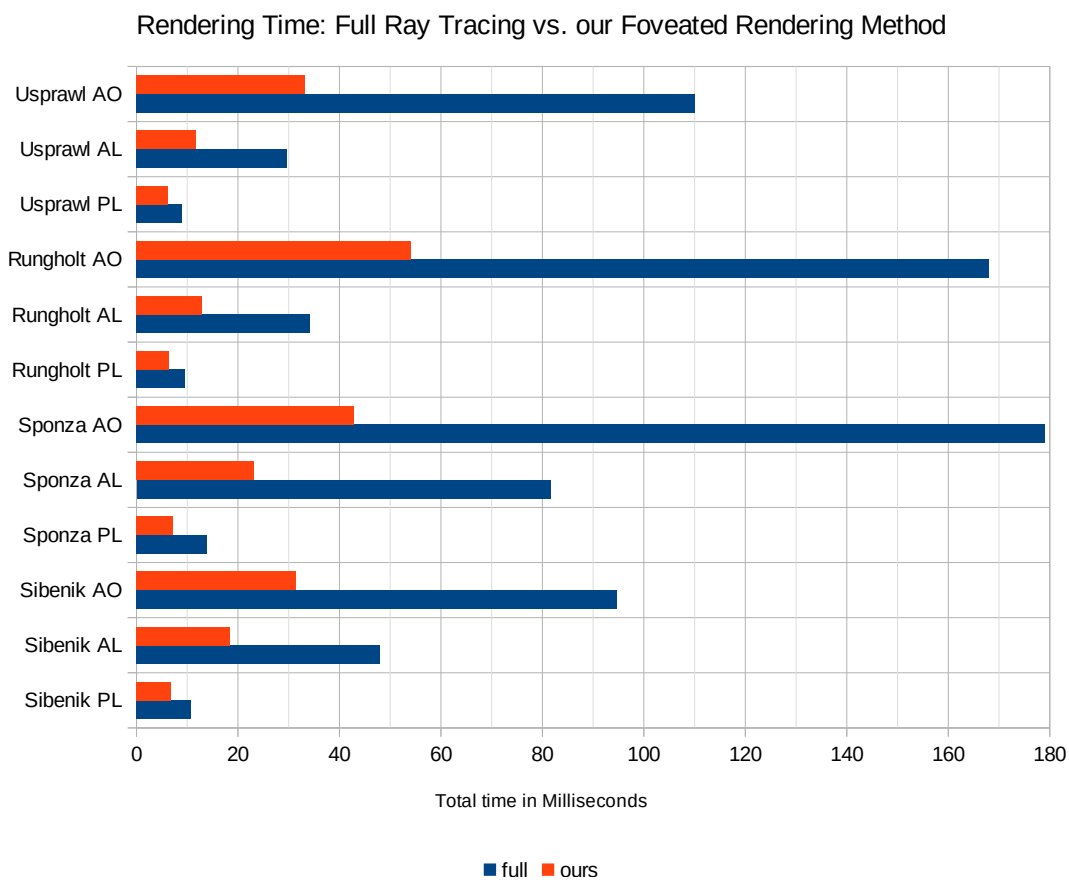


Figure 4.7: Average total rendering times for each scene for our foveated rendering method and full ray tracing.

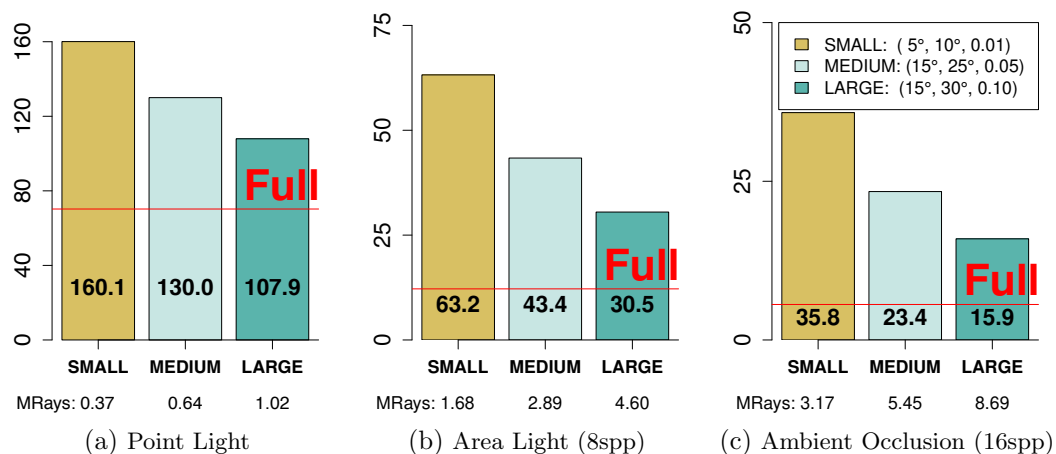


Figure 4.8: Influence of the FRCs on FPS for the scene Sponza and the different lighting conditions (point light, area light, ambient occlusion). MRays denotes the mean number of rays per frame. The scene was rendered at a resolution of 1182×1464 pixels using an NVIDIA GeForce Titan X.

Guenter, Finch, et al. (2012)’s method of rasterizing the image in three layers with different resolutions yields a speedup of 6.2 with only 7% of the pixels being rendered as the images are strongly undersampled. To still achieve an acceptable visual quality, this method needs to rely on specific anti-aliasing methods, limiting its applicability. Moreover, numbers are only reported for a single scene. By using NVIDIA’s Multi-Resolution Shading (Nathan 2015), a speedup of 1.3 to 2 is reported depending on the configuration. Stengel, Grogorick, et al. (2016) report a speedup of 1.34 on average, with the number of shaded pixels being decreased by 65% for a resolution of 1280×1440 pixels and 83% for twice as many pixels. Our method has shown a reduction of sampled pixels by 79% on average for all benchmark scenes, with an average speedup of 2.55. The ray-based approach presented by Fujita and Harada (2014) has shown similar frame rates compared to our approach even though they use different and more GPUs rendering at a lower resolution. The performance of our method could be further improved by generating rays that directly match the image distortion of the HMD, making it possible to cope with even higher resolutions and refresh rates.

4.4 Experimental Evaluation: User Study

Since the proposed system is based on perceptual considerations, instead of relying on a numerical evaluation of image quality, we conducted a user study where participants had to rate the image quality on a Likert scale regarding the possible appearance of noticeable artifacts. It was driven by the following research questions:

- RQ1: Can subjects differentiate between scenes with varying graphical con-

texts, rendered with and without our foveated rendering method?

- RQ2: Do modifications of the foveal region parameters in the ray generation have an effect on the perceived visual quality?
- RQ3: Does the fixation type have an effect on the perceived visual quality?

4.4.1 Experimental Procedure and Design

The experiment was conducted as a within-subject study (Field and Hole 2003), employing a $4 \times 4 \times 3$ full factorial design. Each participant completed 96 trials in randomized order, consisting of a full factorial combination of four scenes {Sponza, Tunnel_Geom, Tunnel_Maps, Rungholt} (see Figure 4.6), four FRCs {small, medium, large, full}, and three fixation types {fixed, moving, free}. A more detailed description of the FRCs and fixation types is given below. All conditions were presented twice. Full ray tracing was included as the FRC *full*, representing our control group. Each trial consisted of an 8-second-flight with one factor combination. In the following paragraphs we describe how we approached the research questions stated above.

RQ1: We varied the test scenes to study the effect of graphical contexts on the noticed visual differences (artifacts). The selection of scenes was driven by perceptual differences of the peripheral visual field as opposed to central vision, including colors, patterns, shapes (Lou, Migotina, et al. 2012), and contrasts in the near peripheral field (Legge and Kersten 1987). *Sponza* represents the most real-world-like scene: While some discontinuities (and thus hard edges) are usually visible, the scene also contains some smoother curves resembling real objects. *Tunnel_Geom* contains a tunnel consisting of noisy, displaced geometry. Depending on the point of view, this scene can contain both hard edges and smooth, continuous surfaces. *Tunnel_Maps* is a tunnel textured with a checkerboard map and a noise texture. *Rungholt* is a scene generated from a Minecraft map with many visible depth discontinuities, which can be challenging for our reprojection and resampling method.

RQ2: The FRCs were given by the following eccentricity thresholds and minimum sampling probabilities: *small* ($r_0 = 5^\circ, r_1 = 10^\circ, p_{\min} = 0.01$), *medium* ($10^\circ, 20^\circ, 0.05$), *large* ($15^\circ, 30^\circ, 0.1$) and *full* ($\infty, \infty, 1$). FRCs were determined by using the angular size of the fovea for r_0 with a steep falloff for the smallest setting and increasing the foveal region and minimum sampling probability while reducing steepness for the other settings. The smallest FRC was expected to yield visible artifacts for most participants, as the foveal region used for rendering matched the fovea. The medium and large FRC extended the foveal region to include the parafovea and perifovea, respectively.

RQ3: While eye tracking determines a user’s PoR in the scene (defining the foveal region), fixation may affect visual attention, potentially leading to visual tunneling effects (Miura 1986). We varied fixation types to trigger different levels of visual attention. The *fixed focus* mode contained a static fixation cross at the image center to be focused for the entire trial. For the *moving target* mode, a set of paths across the image plane was generated. A green sphere linearly moved along these paths, fitted to the according scene depth to support the user’s ability to focus. Paths varied in all trials except repetitions to avoid learning effects. The foveal regions for fixed focus and moving target was centered around the fixation target to avoid the negative influence of the eye tracker’s inaccuracies (relatively low refresh rate, inaccurate tracking towards outer display regions). Trials with free focus fixation enabled the user to look around freely with the foveal region following the user’s gaze. The moving target fixation mode was expected to require more visual attention as the user had to concentrate on following the target. However, it was not clear how this would affect the quality ratings given by the participants.

The setup used for the user study differed from the benchmark configuration. It comprised an Oculus Rift DK2 (SDK 0.8) on a Windows 10 system including an Intel Xeon E5-2609 (2.4GHz), and 64GiB of RAM. The DK2’s native refresh rate of 75Hz was used as the baseline for our user study. As both the foveated rendering and the OpenGL-based reprojection process had to be parallelized in order to achieve this frame rate, it was necessary to deploy two Quadro K6000 cards. These were required because the unavailability of a Linux driver for the utilized eye tracker tied us to Windows, which does not allow for multi-GPU rendering on NVIDIA’s consumer cards. The Oculus was equipped with an SMI binocular eye tracker running at 60Hz (asynchronous). The rendering resolution was equal to the benchmarks with one image being rendered for each eye at the full resolution of the Oculus Rift DK2. With some minor optimizations, we achieved frame rates of at least 75Hz for all scenes, including the final image warping to display them in the Oculus. However, all sequences used for full ray tracing had to be pre-recorded (excluding any optimization like reprojection), loaded at runtime, augmented with the specific trial fixations and displayed at 75Hz.

After signing informed consent and receiving instructions, participants were seated and equipped with the HMD. Prior to the main experiment, six test trials of an alternative flight through Sponza were shown, including the smallest FRC, full rendering and all fixation types. After each main trial the participants rated their agreement to the following statements:

Q1: The shown sequence was free of visual artifacts.

Q2: I was confident giving this answer.

Both had to be rated on a 7-point Likert scale from *strongly disagree* (-3) to *strongly agree* (3).

4.4.2 Results

15 subjects (10 male/5 female, all with academic background) aged between 26 and 51 ($M = 33$, $SD = 7.24$), with normal or corrected-to-normal vision participated in the user study. A multi-factor analysis of variance (ANOVA) (Field and Hole 2003) was performed on the data (1440 trials). We analysed significant interactions and the observed main effects with post-hoc t-tests using Holm's method for p -value adjustment. Confidence values (Q2) were mostly high ($M = 1.62$, $SD = 1.14$), with very small differences. Consequently, we do not consider differences in confidence in our analysis any further.

RQ1: Differentiation between foveated and full rendering. Mostly, users cannot reliably differentiate between full and foveated rendering. This is the case for foveal regions not smaller than roughly 10° , and scenes with little to moderate amounts of high-frequency geometry. Figure 4.9 shows responses for varying FRCs and all scenes. However, statistical data reveals that differentiation depends significantly on all factors: FRC size, fixation mode and the displayed scene. While FRC shows a significant main effect ($F \approx 30.54$, $p \approx 0$), we also found a strong interaction between FRC and SCENE ($F \approx 3.09$, $p < 0.005$). Hence, we performed t-tests, which showed that significant differences between the medium, large, and full FRC were only present in the scene *Tunnel_Maps*. All other scenes only showed significant differences when the small FRC was involved. We mainly attribute this to the regular, high-contrast checkerboard pattern displayed in *Tunnel_Maps*.

RQ2: The effect of foveal region size. As noted in RQ1, if the foveal region is medium-sized or larger, users will hardly notice visual artifacts for most scenes. Figure 4.10 shows the responses for varying FRCs, including the mean values and standard deviations. The small FRC scored significantly lower, while medium and large FRCs were almost identical regarding perceived visual artifacts. Compared to full rendering, the mean rating was slightly lower, with an identical median and a similar standard deviation. As Figure 4.9 illustrates, this can again be mainly attributed to the artifacts visible in *Tunnel_Maps*.

RQ3: The effect of fixation types. Fixation types, associated with different levels of visual attention, had a significant main effect ($F \approx 3.46$, $p \approx 0.03$) on the perceived visual quality. While *free* ($M = 0.43$, $SD = 1.89$) and *fixed* ($M = 0.43$,

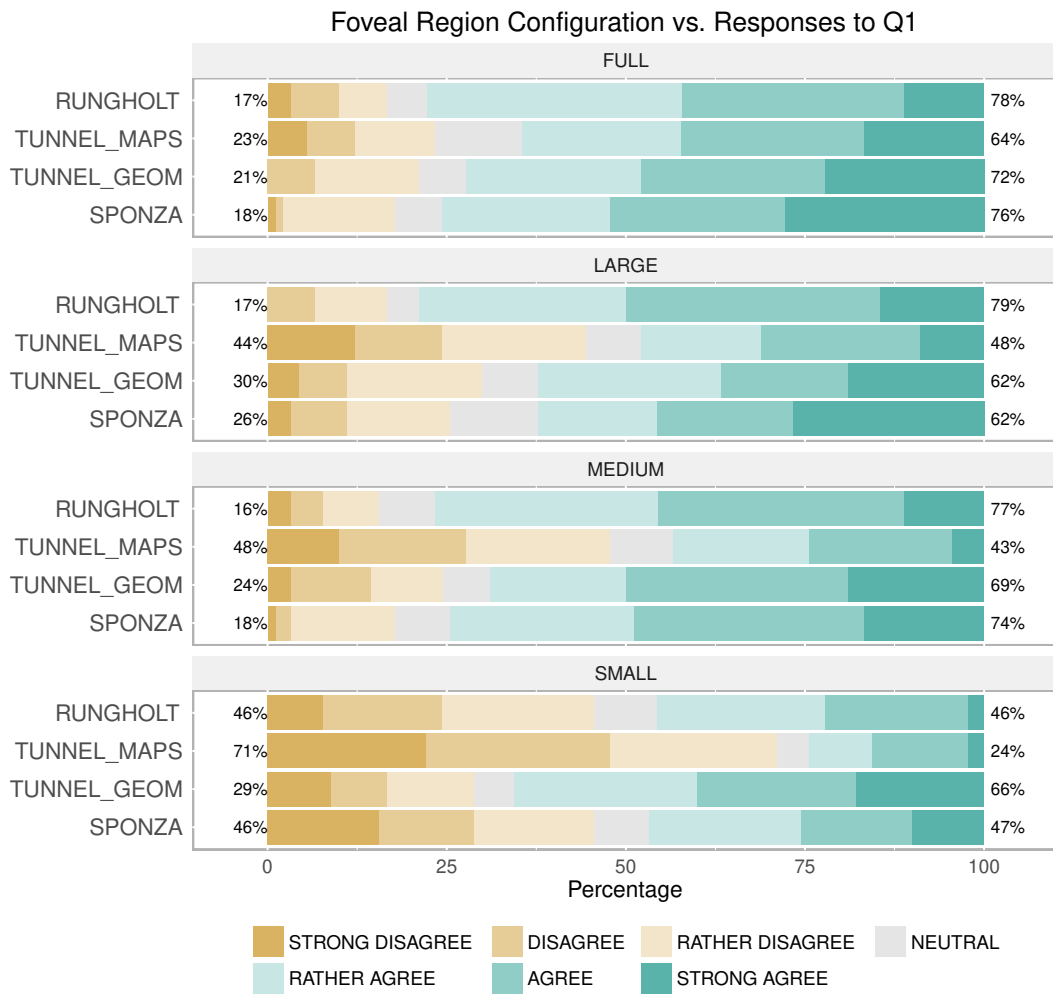


Figure 4.9: Likert-scale ratings for Q1 (*The shown sequence was free of visual artifacts.*) for all scenes grouped by foveal region configurations. The percentages on the left and right represent the fraction of all participants that had a tendency towards *disagree* and *agree*, respectively. It is clearly visible that the smallest FRC revealed a significant amount of artifacts, while the larger foveal regions were close to the full renderer in this regard.

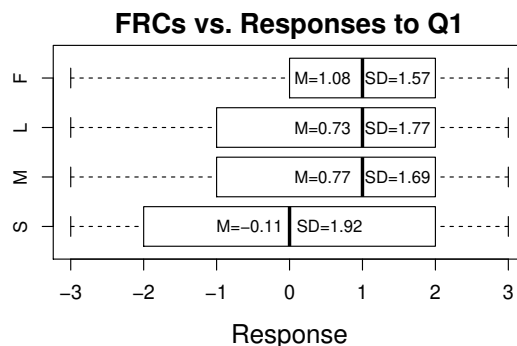


Figure 4.10: Likert-scale ratings of perceived visual artifacts for **S**mall, **M**edium, **L**arge and **F**ull foveal regions. While small caused neutral ratings on average, ratings for medium and large were not significantly different from full rendering.

$SD = 1.81$) modes showed nearly identical responses, the *moving target* was rated significantly better ($M = 0.99$, $SD = 1.63$). Thus, fewer visual artifacts were noticed with presumed higher visual attention caused by the moving target, as users were likely less aware of details outside the focus area (Miura 1986). This is highly interesting as it could lead to the possibility of further reducing the sampling rate outside the foveal region. Furthermore, the foveal region matched the gaze when the target was perfectly followed. We assume visual tunneling did not affect the other fixation modes, which made it easier to spot artifacts in the periphery.

4.5 Experimental Evaluation: Analysis of Eye Tracking Data

In this section, we analyse the recorded tracking data and the corresponding quality ratings for the presence of any noteworthy effects related to visual attention. Such perceptual effects may have an impact on quality ratings that may not be expected from a simple look at the raw data. We would also like to give a deeper insight into tracking quality by providing information on the relation between a user's gaze and the given quality ratings. For distance values, we used the average of the individual measurements for the left and the right eye.

4.5.1 Methods

First of all, we tried to determine the actual tracking precision in order to ensure that the recorded data is valid. As mentioned above, when using a tracking device, degradation of the tracking precision towards outer image areas was quite noticeable. This may also be one reason why the calibration process of SMI's SDK only employs a relatively small area around the image center. We estimate the tracking precision by looking at the deviations of the recorded PoR from the fixation target's current position.

We assume that the fixation accuracy, which describes how well a user can fixate a target, is largely independent of the target's position in the image. This implies that worse fixation towards outer areas results mainly from tracking inaccuracies. To give an estimate of tracking precision, the data is sorted into bins and the mean value for each of these bins is computed. Each bin has a width of $w = 0.1^\circ$, and there is a total of $n = \lceil \max\{F_{p,t}(i)\}/w \rceil$ bins $B_j = (\bar{F}_j, \bar{G}_j)$, $0 \leq j < n$, where

$$F_j = \{F_{p,t} | j \cdot w \leq F_{p,t}(i) < (j + 1) \cdot w\}, \quad (4.4)$$

$$G_j = \{G_{p,t} | F_{p,t}(i) \in F_j\}. \quad (4.5)$$

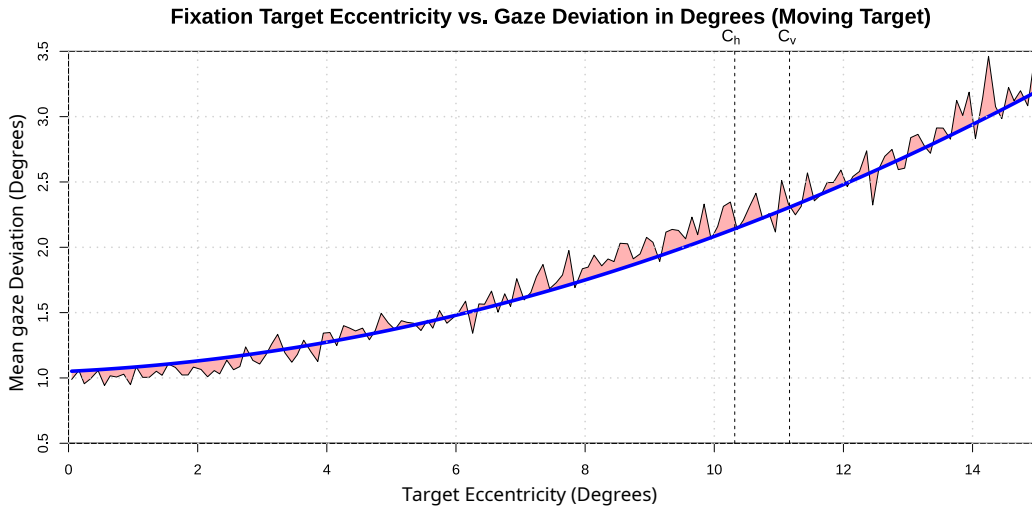


Figure 4.11: Tracking precision vs. fixation target’s distance to the image center. The area used for calibration by the tracking device is also denoted here by C_h (horizontal extent) and C_v (vertical extent). The result of linear regression with a quadratic equation is represented by the blue line. The red area illustrates the residuals.

Here, $F_{p,t}(i)$ is the distance between the fixation target’s current position and the image center, and $G_{p,t}(i)$ represents the distance between the gaze and the fixation target in trial t at frame i for participant p . Thus, \bar{G}_j is the average value of this distance for bin j . It provides an approximate tracking quality measure for the eccentricities contained within the interval $[j \cdot w, (j + 1) \cdot w]$. This data is analysed further by performing a linear regression, the results of which are described in Section 4.5.2. The results of the binning-and-comparing process are shown in Figure 4.11.

Because of the inaccuracies towards outer regions, we filtered the logged data used for analyzing the tracking information to only include the region utilized by SMI’s calibration method. This region extends to maximum eccentricities of approximately $C_h = 10.3^\circ$ horizontally and $C_v = 11.68^\circ$ vertically, resulting in a maximum eccentricity of $\sqrt{C_h^2 + C_v^2} = 15.57^\circ$ diagonally. These numbers were taken from SMI SDK’s 9-point calibration method and converted to angles.

The average fixation accuracy of participants is then compared for all tested scenes. Subsequently, the measured accuracies are compared to quality ratings for each scene individually and we try to give explanations for the apparent effects. In order to support our findings regarding tracking precision, the relation between quality ratings given by the users and the average eccentricities of the PoRs in free focus mode is looked into.

According to Adler, Kaufman, et al. (2011), adults can physically rotate the eye up to 50° horizontally, 42° up, and 48° down around the main optical axis in the

eye's resting position. It has to be noted however that in practice, humans usually do not exhaust these physiological limits. Instead, after exceeding a certain angular deviation, humans will tend to turn their head. According to Defense (1999, p. 17), this angular deviation is referred to as the comfortable viewing angle (CVA) and is approximately 15° . Thus, we chose not to account for fixation target eccentricities larger than the CVA in our tracking precision measurements. Also, the 9-point calibration used by SMI's eye tracking SDK only uses eccentricities that are approximately 0.5° above the CVA.

Head tracking was not implemented in our user study, as it was necessary to present identical visual stimuli to all participants. If we enabled users to freely move their head in the virtual environment, this would not have been possible. However, for fixation target positions further away from the image center than the CVA, users would most likely not just rely on eye movement to fixate a target, but instead incorporate head movement.

4.5.2 Results

In this section we present the results of our analysis of the recorded eye tracking data. Tracking precision, fixation accuracy, as well as the relation between fixation modes and perceived quality are looked into.

Tracking Precision

To analyse the relation between the tracking precision and the PoR's actual eccentricity, which we assume to be identical to the fixation target position at this point, we perform a linear regression with $\hat{G} = \beta_0 + \beta_1 F_j + \beta_2 F_j^2$. This results in a correlation of 0.989 with $\beta = (1.05, 0.024, 0.008)$ and $R^2 = 0.978$ with the constant ($p \approx 0$), linear ($p < 0.01$), and square ($p \approx 0$) terms being statistically significant. The quadratic prediction for gaze deviation is illustrated in Figure 4.11, where the plot of the regression result clearly shows the dependency between eccentricity and tracking precision.

Fixation Accuracy

We analysed angular differences between the fixation point and the tracked gaze. Participants stayed closer to the fixation point for the fixed mode ($M = 0.31^\circ$, $SD = 0.4^\circ$) than for the moving target ($M = 1.9^\circ$, $SD = 1.52^\circ$). Keeping in mind that Tunnel_Maps had the greatest amount of visible artifacts for the participants, it is important to mention that the median angular differences for Sponza, Tun-

nel_Geom and *Rungholt* were between 1.25° and 1.58° , while for Tunnel_Maps, a median difference of 2.24° was found. As this larger distance to the foveal region's center indicates that the gaze was closer to sparsely sampled regions, it is one explanation for the relatively low Likert ratings for this scene. There were no significant differences between angular differences for varying FRCs. The median values over all scenes for the four FRCs were all within $[1.62^\circ, 1.7^\circ]$ for the moving target and $[0.22^\circ, 0.25^\circ]$ for the fixed mode.

Figure 4.12 shows the cumulative distribution functions (CDFs) for the fixed and the moving fixation target for all tested scenes. The horizontal axis represents the angular distance between the user's gaze and the fixation target. A significant difference between the fixation accuracy for the fixed target (95% of the measurements were below 1.1° in the worst case) and the moving target (95% of the measurements were below 4.5° in the worst case) is clearly visible. In Section 4.6, we explain the result that would normally be expected from this difference in accuracy and compare it to the users' actual quality ratings. Figure 4.13 shows heatmaps for the distribution of gaze deviation for fixed and moving targets, respectively.

As the color range indicates, it is also illustrated how often the participants have looked at the respective relative positions to the fixation target. The slight right shift for the gaze deviation can be explained by the utilized fixation paths not being equally distributed regarding the fixation target's movement. Looking into the paths, it became clear that the fixation target moved left more often than right, which is a possible explanation for the PoR's right shift.

We attribute the differing fixation accuracy between the moving target and the static fixation cross to the presence of smooth pursuit eye movements (SPEMs). While the moving object's speed did not exceed a velocity of $100^\circ/s$, where a decrease in accuracy is expected due to physiological constraints, the target's movement was not predictable for the user, which leads to a reduced SPEM precision. Additionally, precision is reduced due to the fact that the background is at the same distance as the pursuit target, which leads to the inability to use other signals to discriminate between target and background (Adler, Kaufman, et al. 2011, p. 229).

Subjective Perceived Quality: Fixed and Moving Target

Figure 4.14 shows that the average quality for the moving target was rated better for all scenes on average. In order to shed some light on the influence of the actual rendering detail, Figures 4.15 to 4.18 illustrate the data for the individual scenes, each with all three fixation modes and all foveal region configurations up to full rendering. The red lines show the means for each of the fixation modes, exhibiting that the aforementioned effect is present in all tested scenes. It also becomes apparent

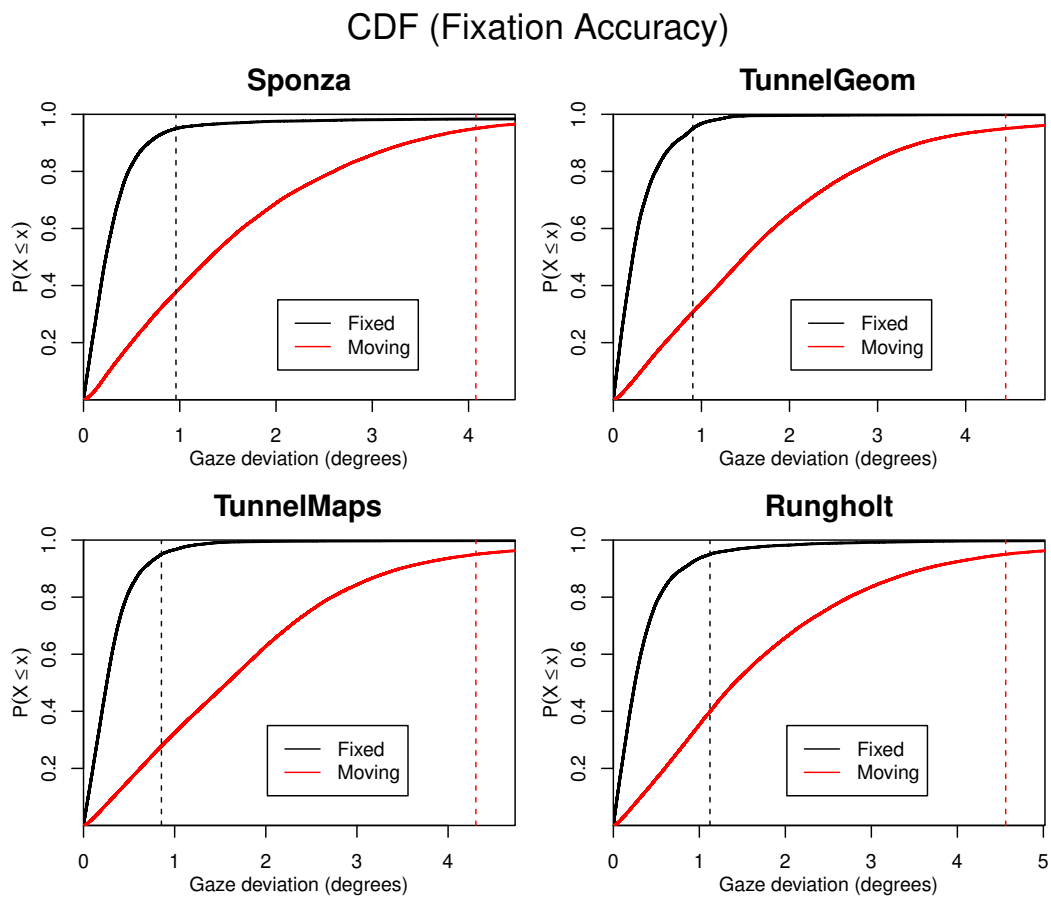


Figure 4.12: CDFs of the measured fixation accuracy for fixed and moving targets. The 95% quantiles of gaze deviations for each scene are illustrated with dotted lines. There are significant differences between the fixation accuracy for the fixed and the moving fixation targets. X is the actual gaze deviation.

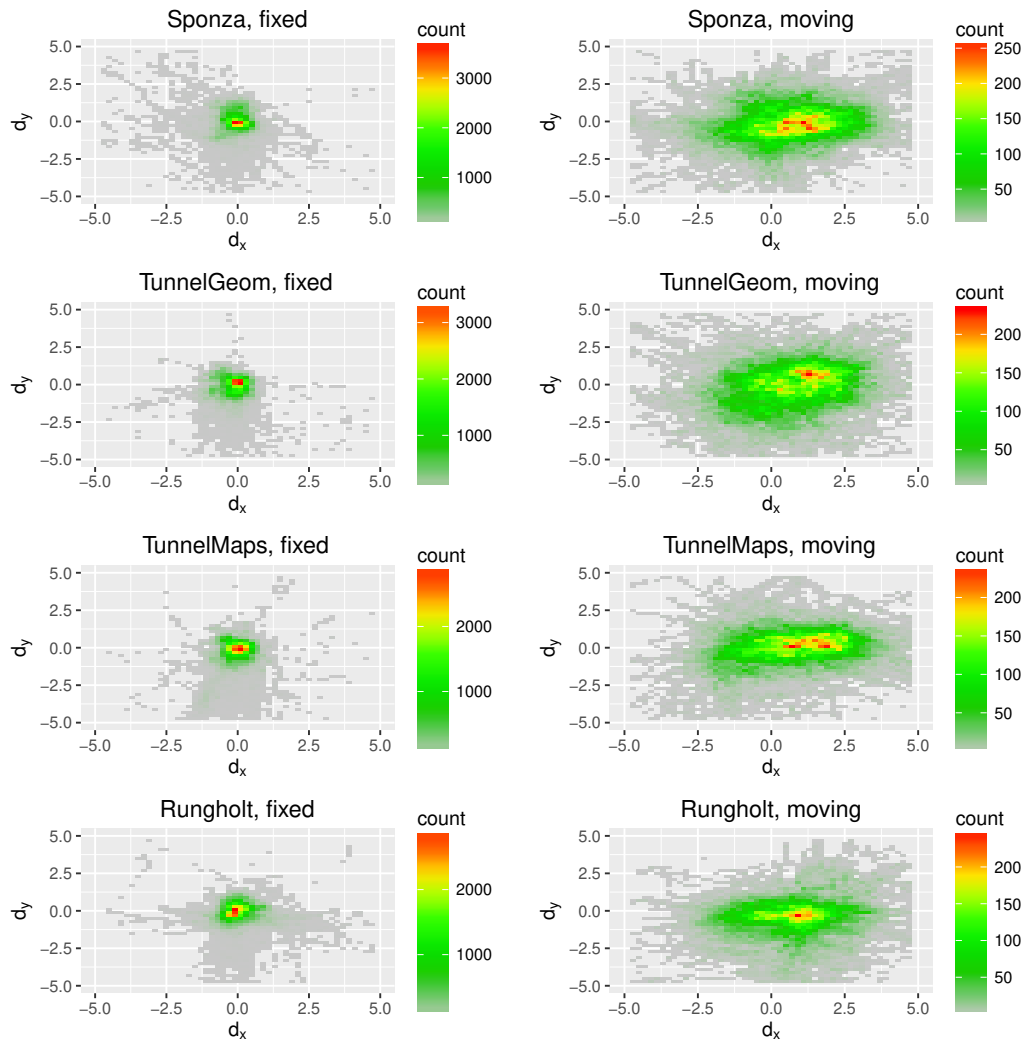


Figure 4.13: Gaze deviation for all individual scenes, fixed and moving targets.

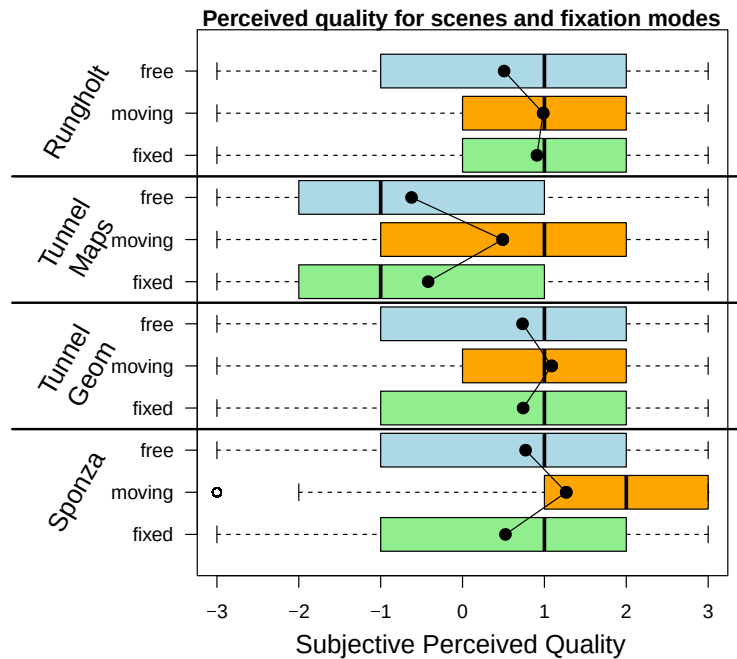


Figure 4.14: Quality for all combinations of scenes and fixation modes. Quality ratings were the highest for all scenes when the moving target fixation mode was selected, although the fixation accuracy was worse for the moving target than for the fixed target. The black dots inside the boxes represent the respective mean quality ratings. They are connected to better illustrate differences.

that the increase in rendering detail between the medium and the large FRC did not result in a consistent improvement of subjective perceived quality. For the moving fixation target, differences from a medium FRC up to full rendering are mostly negligible. Interestingly, in some cases a larger FRC even results in lower subjective perceived quality. We try to explain the given quality ratings in the discussion section, as they contradict intuition at first.

Subjective Perceived Quality: Free Focus

In the free focus mode, users were allowed to move their eyes freely instead of having to follow a prescribed path. As we have shown above, tracking quality seemingly degrades with increasing eccentricities. To prove that this apparent degradation does not only come from fixations, saccades and other disturbances not being filtered from the raw data, we take a look at the eccentricity-dependent quality ratings in the free focus mode. Figure 4.19 shows illustrations of the according data (eccentricity and quality ratings) for all scenes. The left column contains scatter plots for each scene. The horizontal axis represents the eccentricity, while the vertical axis represents the mean quality per bin, which has been computed for bins of size $w = 0.1^\circ$. Each bin contains the mean quality rating of all trials where the according eccentricity

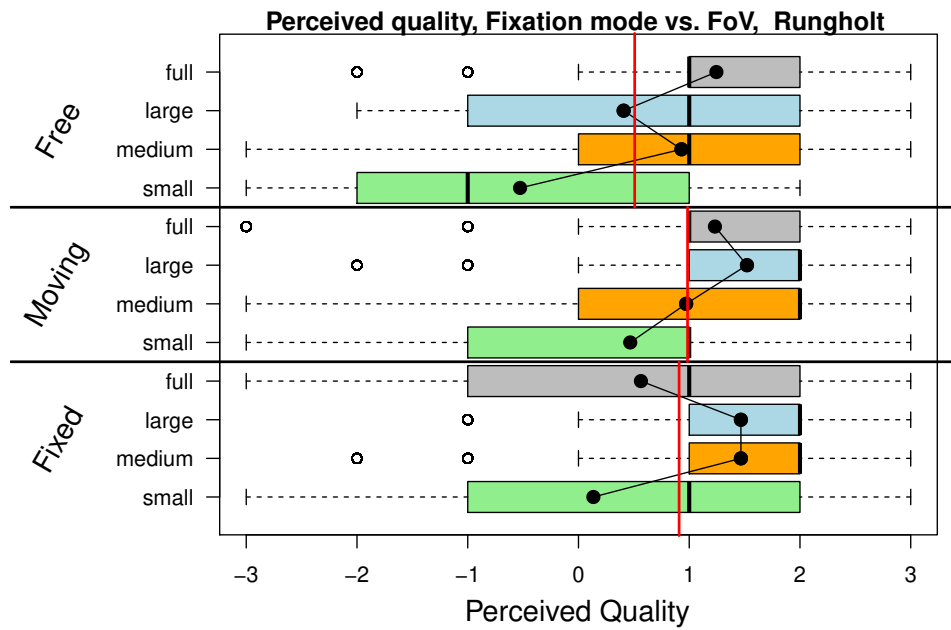


Figure 4.15: Quality ratings for fixation modes and foveal region configurations, scene Rungholt. The red lines represent the mean quality ratings per fixation mode.

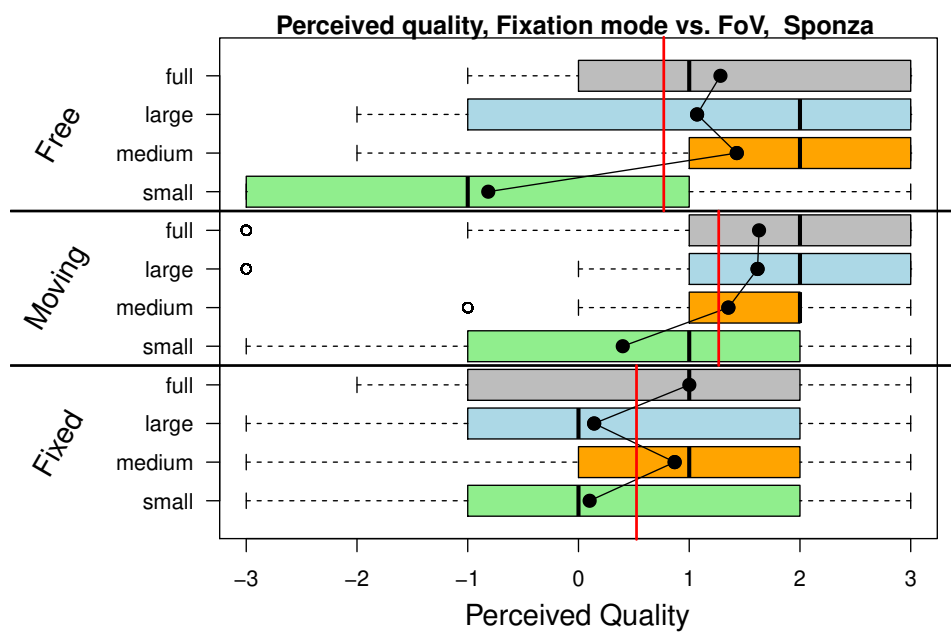


Figure 4.16: Quality ratings for fixation modes and foveal region configurations, scene Sponza. The red lines represent the mean quality ratings per fixation mode.

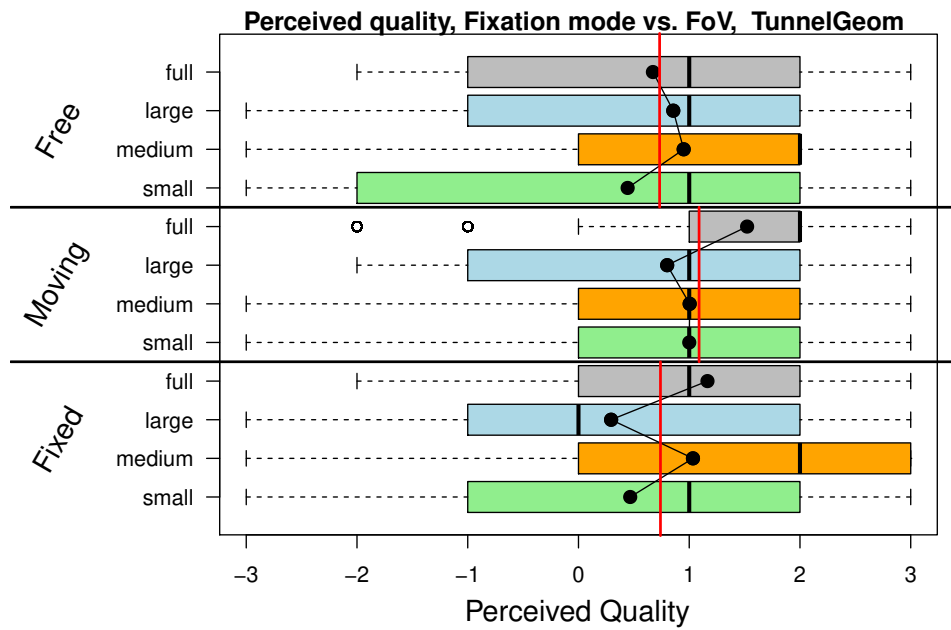


Figure 4.17: Quality ratings for fixation modes and foveal region configurations, scene Tunnel_Geom. The red lines represent the mean quality ratings per fixation mode.

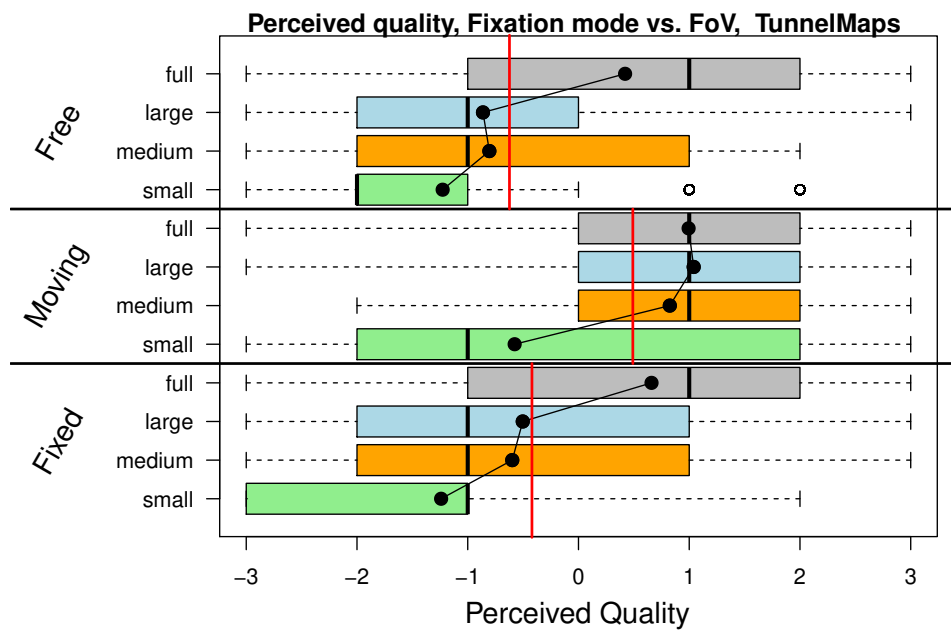


Figure 4.18: Quality ratings for fixation modes and foveal region configurations, scene Tunnel_Maps. The red lines represent the mean quality ratings per fixation mode.

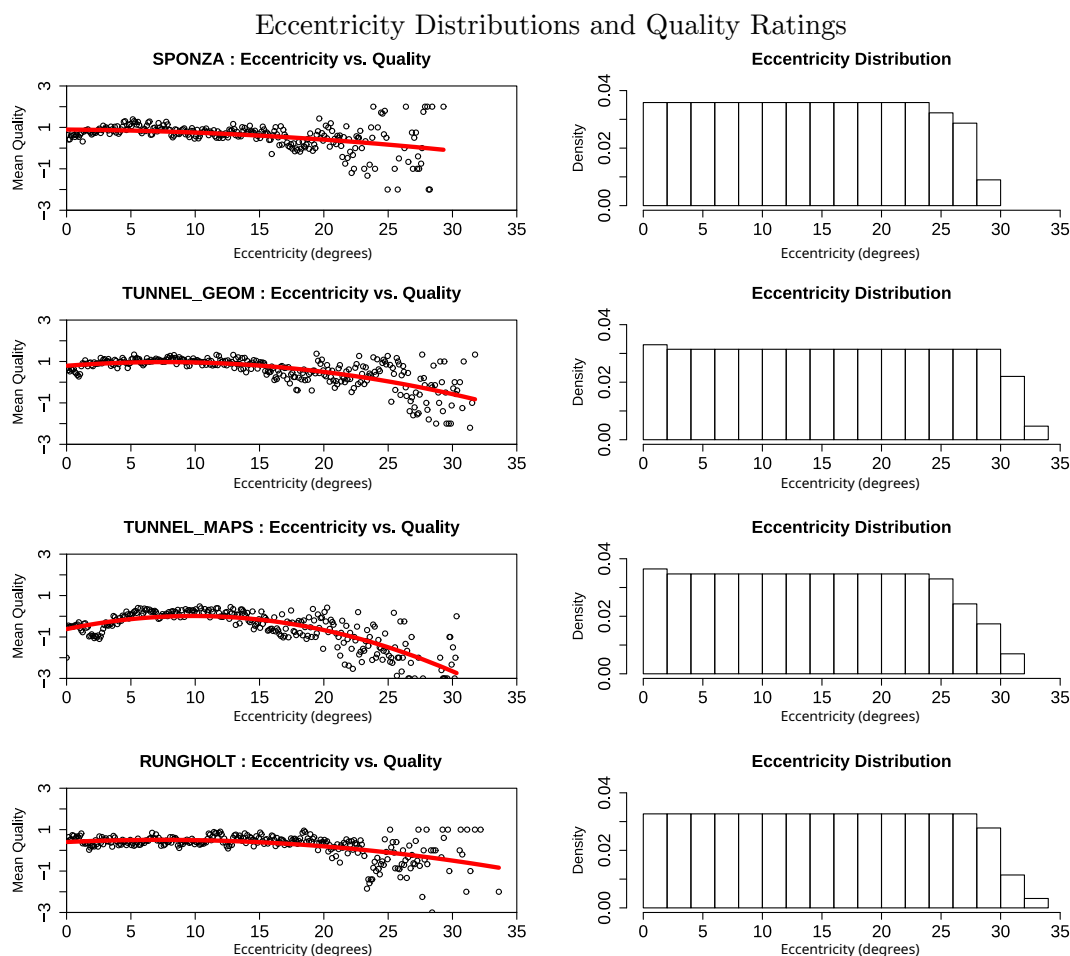


Figure 4.19: Eccentricity-dependent mean quality measurements and eccentricity distributions for all scenes in free focus mode. It is clearly visible that the subjective perceived quality degrades with increasing eccentricities. Eccentricity distributions are similar and almost uniform for all scenes.

had been measured at least once. Another possible approach would be to average eccentricities for all individual trials and bin the data based on that. In addition to the mean quality for each of the bins, we performed linear regressions with quadratic equations, which are included in each of the plots as a red curve. The right column shows eccentricity distributions for each scene, i.e., it gives an idea about how far away from the image center the user inspected the shown scenes. Differences between scenes turn out to be mostly minuscule or at least too small to draw any further conclusions. Further remarks regarding these illustrations are made in the discussion section.

4.6 Discussion and Conclusion

In this chapter, we presented a novel approach for foveated rendering using adaptive ray tracing and reprojection from previous frames. Our method is well-suited for wide-FoV HMDs equipped with an eye tracking device. Sparsely sampled image data is reprojected to new views using a depth mesh generated from a low-resolution G-buffer. The influence of errors arising from reprojection or regions critical for perception is lowered by an update strategy that allows for resampling critical image regions incorporating the samples' age, which is linked to the samples' quality as well.

Our method enables the visualization of static scenes with millions of triangles within the Oculus Rift DK2 at a refresh rate of 75Hz. The benchmarks have shown significant performance gains, while the user study revealed the perceived visual quality for even moderately sized FRCs is almost on-par with full rendering. Anti-aliasing is crucial to rendering for HMDs: Pixels are distributed over a large FoV, making the pixel raster visible. Therefore, jagged edges and undersampling may easily become apparent.

Our strategy is to shoot more rays by jittering the ray positions over the pixel area. To improve the reprojected image's quality and handle errors while accounting for the user's gaze, each pixel that is either part of the foveal region or marked in the resampling info is sampled by shooting a ray. The results are then combined using the running estimate.

We also presented an analysis of the eye tracking data recorded in the conducted experiments. Tracking precision has been analysed regarding its angular dependencies, revealing a clear drop of tracking quality for higher eccentricities. Accordingly, quality ratings for free focus mode also show a clear drop towards larger eccentricities. Such properties of tracking devices have to be accounted for when implementing foveated rendering methods, as the PoR is a crucial measurement in such setups. Having measured these inaccuracies of the tracking device, it becomes clear that applications that rely on methods from this field have to adjust the specific parameterizations for the given circumstances, e.g., by enlarging the area rendered at full detail.

We have analysed the ability of users to fixate static and moving fixation targets. While we found the PoR being scattered over larger areas for the moving target mode, the results seemed to contradict the intuitive assumption that worse fixations should result in worse quality ratings. The mean quality ratings were best for the moving target mode in all scenes, even though the match between the measured PoR and the actually focused PoR was worse than for the static fixation mode. Even

though this may lead to subsampling and reprojection artifacts being exposed to the user, the ratings were still better, which we attribute to the potential presence of visual tunneling effects that are induced by the mental load of the task that has to be carried out, although the task has just been to follow a moving point. Effectively, this reduces the user's FoV.

We have also found that an increase in rendering detail did not always result in improved quality ratings. One possible cause for this is the reprojection method hiding visual artifacts by effectively putting a low-pass filter over them, as even full rendering still is a subsampling of the rendered scene, just with a finer and more regular pixel grid. A more detailed explanation of blur effects that occur when using reprojection methods is given by Yang, Nehab, et al. (2009).

In the next chapter, we discuss our method for accelerating preview rendering of global illumination (GI) solutions with a hash-based caching method. With a fast enough reconstruction, such methods could certainly be combined with a foveated rendering process in order to bring them to an immersive virtual environment displayed on an achmd.

Chapter 5

Hash-based Hierarchical Caching and Layered Filtering

Part of the work included in this chapter was previously published in (Roth, Weier, et al. 2019) and (Roth, Weier, et al. 2020), lead-authored by the author of this thesis.

Modern Monte-Carlo-based rendering systems still suffer from the computational complexity involved in the generation of noise-free images, making it challenging to synthesize interactive previews. We present a framework suited for rendering such previews of static scenes using a caching technique that builds upon a linkless octree. Our approach allows for memory-efficient storage and constant-time lookup to cache diffuse illumination at multiple hitpoints along the traced paths. Non-diffuse surfaces are dealt with in a hybrid way in order to reconstruct view-dependent illumination while maintaining interactive frame rates. By evaluating the visual fidelity against ground truth sequences and by benchmarking, we show that our approach compares well to low-noise path-traced results, but with greatly reduced rendering times. This way, our caching technique provides a useful tool for global illumination previews and multi-view rendering.

5.1 Introduction

We introduce the HashCache, a hierarchical world space caching method for GI rendering of static scenes, based on Choi, Ju, et al. (2009)’s linkless octree. Using a hash-based approach makes it possible to perform the reconstruction of cached illumination in constant time, depending only on the actual screen resolution (assuming that the visible geometry is known). This makes our technique well-suited for the exploration of static scenes. Despite only caching diffuse illumination, our system explicitly supports non-diffuse materials through a hybrid reconstruction scheme. This is an approximate final gathering step similar to Photon Mapping (Jensen 1996; Spencer and Jones 2009), which is performed before the actual reconstruction.

For non-diffuse materials, this step is composed in a hybrid way: Rays are traced up to the first hitpoint that is interpreted as a diffuse material, where the pre-gathered information is then queried from the cache and modulated with the path throughput. This process is described in more detail in Sections 5.2.2 to 5.2.4. Compared to precomputed radiance transfer, our preprocessing time is much shorter, as we only need to determine geometric cell occupations.

In order to reduce quantization artifacts, we employ a spatial jittering method inspired by Binder, Fricke, et al. (2018)’s work on hash-based path space filtering. To increase image quality by reducing noise, we suggest a layered filtering framework, basically projecting Keller, Dahm, et al. (2014)’s original path space filtering method to image space. The practicability of our approach is demonstrated by extending a basic cross-bilateral denoising filter by integrating it into our framework and adjusting it to the kind of noise present in our system, enabling it to filter the image content per light bounce. With this method, we especially aim for improving the visual quality of non-diffuse materials, compared to filtering only at the primary hitpoint without any information about the transport paths. Most image space filtering methods may be integrated into the suggested framework in order to improve their handling of specular or glossy material types.

Eventually, we present image quality comparisons, performance benchmarks, and an analysis of memory requirements, showing the practicability of our approach. While maintaining interactive frame rates, the noise in the image can be reduced significantly. We show that our approach performs comparably to much higher sampling rates in path tracing regarding relative mean-square error (relMSE) and multi-scale structural similarity (MS-SSIM) metrics.

5.2 Method

In this section, we give an overview of the employed cache structure and describe how it can be used to cache the data generated by stochastic rendering methods. Subsequently, we give more details on how samples are generated during the rendering process in order to reuse recursively generated hitpoints. Here, it is also shown how the actual cache updates are performed. Eventually, we provide information about the reconstruction process, including the support of non-diffuse materials, as well as our proposed layered filtering framework.

5.2.1 Cache Structure

Monte Carlo (MC)-based rendering methods provide the means to solve Kajiya (1986)'s Rendering Equation numerically. In our implementation, a straightforward path tracer with next-event estimation and multiple-importance sampling is used for computing the illumination data. The path tracing process generates millions of randomly and sparsely distributed hitpoints located on the scene geometry in each iteration. This means that we are not supporting participating media to be cached. Consequently, a data structure that allows for efficient caching of such data must allow for querying large amounts of randomly distributed keys at a high performance.

The core of our HashCache system is Choi, Ju, et al. (2009)'s concept of a link-less octree, consisting of a number of hash maps implemented with Amenta and Alcantara (2011)'s Cuckoo Hashing. This hashing method allows for a worst-case constant lookup time, making its choice especially suitable for real-time previews. Cuckoo hashing resolves collisions by employing an additional hash function in order to compute two candidate indices in the hash table for one key. When a collision is detected on key insertion, the already-existing entry is replaced by the new entry. Then, the old entry is inserted at its alternative position. Potential collisions are handled the same way iteratively until all entries have been successfully placed. This process may get stuck in an infinite loop if the hash functions are not chosen correctly. However, this issue can be detected and the process is then restarted with automatically chosen alternative hash functions.

Although it would be possible to use a plain grid instead of an octree, we chose to use the hierarchical approach for its inherent level-of-detail support. When rendering a scene from an arbitrary point of view using a non-hierarchical data structure, parts of this data structure will be potentially subsampled, resulting in aliasing artifacts. With a hierarchical data structure like the HashCache, it is possible to choose the hierarchy's level whose resolution most closely resembles the projected pixel size in object space, hiding subsampling artifacts effectively.

While the hash-based octree representation is a compact structure, there still is a trade-off between memory consumption and access time. In order to construct the compact hash map, all cells occupied with geometry have to be marked at the highest resolution available in the octree. This information is determined by testing all grid cells within each triangle's bounding box for an intersection with the triangle, resembling typical grid construction algorithms, such as in the work by P erard-Gayot, Kalojanov, et al. (2017). Because of the large number of grid cells at high resolutions, we choose to represent each cell by a single bit in a field of 32 bit types. Each 32 bit chunk forms a block, which is subdivided spatially at a resolution of $4 \times 4 \times 2 \text{ bits} = 32 \text{ bits}$. The implementation uses CUDA's atomic operations

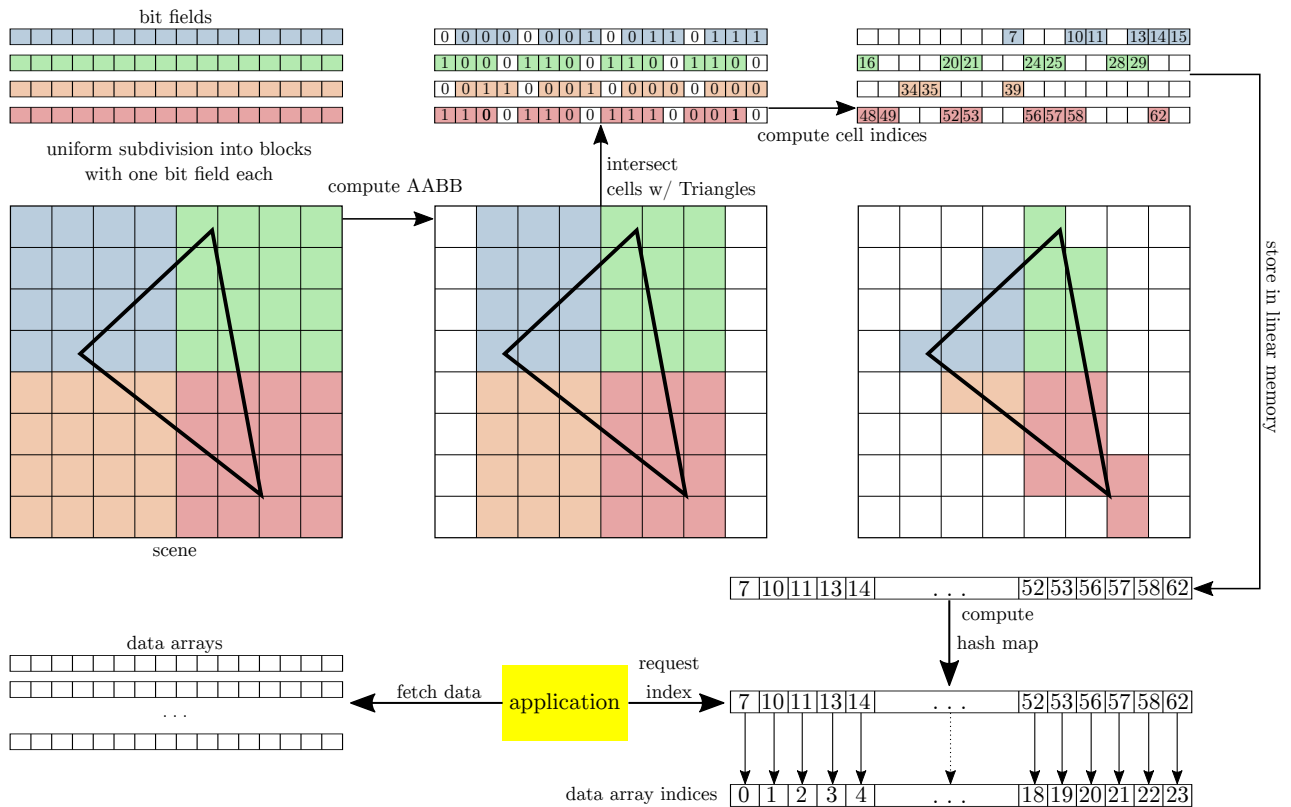


Figure 5.1: Two-dimensional example for the computation of the occupied grid cells and the hash map computation. First, the scene is divided into a uniform grid, where each 4×4 block forms a bit field with 16 elements. In the three-dimensional case, these blocks have a depth of 2 and thus form a bit field with 32 elements each. A rough approximation of the occupied cells is computed by using an axis-aligned bounding box. For each cell of this bounding box, it is then computed whether or not the cell intersects the triangle; if there is an intersection, the according bit is set to 1, if not, it is set to 0. After computing the bit fields, the according linear cell indices are computed and stored in a linear array. This is then passed to the hash map initialization in order to compute the appropriate hash functions that enable access to the data arrays through the linear cell indices without having to store superfluous data.

on the respective chunks, effectively yielding the number of occupied cells. For an illustration of our approach for determining occupied cells, see Figure 5.1.

During the hash map initialization, the number of occupied cells is used in combination with a space-usage factor to limit the actual memory requirements. We choose an initial space-usage factor of $f_0 = 1.1$. If the hash map construction fails, another attempt is made with $f_n = 1.01 \cdot f_{n-1} = 1.1 \cdot 1.01^n$ until construction succeeds. This construction process is performed for each octree level, with cell indices being adapted accordingly. As the utilized hash map implementation is bound to 32 bit keys and an octree's extents are limited to powers of two, the maximum representable resolution is 1024^3 . Higher resolutions are represented by splitting space into multiple hash maps per octree level.

The values stored in the octree's underlying hash maps are actual indices to global data arrays. These arrays occupy exactly the space required to store all of the information computed throughout the process. Note that the presented implementation relies on caching only the outgoing diffuse illumination without any directional information other than the front and back of each cache cell, where the front is determined to be the inverse orientation of the first ray that hits any geometry within a cell. While it would be possible to store information for more directions, this would negatively affect storage requirements and performance. However, storing at least two directions is necessary, since even infinitesimally thin geometric primitives may be illuminated differently from both sides. To store more accurate GI information for these cases, we construct the arrays to contain the following data per cell:

- Diffuse illumination for the front and back of each cell as six half values (96 bits),
- Compressed cell normal (32 bits),
- Currently accumulated number of samples (32 bits),
- Reset information: frame index denoting when the cell has last been wiped (32 bits).

The cell normal is compressed as follows:

$$\text{compress}(\vec{n}) = \vec{n}_c = \begin{pmatrix} \vec{n}_{c,x} \\ \vec{n}_{c,y} \\ \vec{n}_{c,z} \end{pmatrix} \quad (5.1)$$

$$\vec{n}_{c,x} = (2^{15} - 1) \cdot \frac{\vec{n}_x + 1}{2} \quad (5.2)$$

$$\vec{n}_{c,y} = (2^{15} - 1) \cdot \frac{\vec{n}_y + 1}{2} \quad (5.3)$$

$$\vec{n}_{c,z} = \begin{cases} 1, & \vec{n}_z \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.4)$$

This transforms the x and y component to the interval $[0, 2^{15} - 1]$ and the z component to either 0 or 1, depending on the original sign in the uncompressed normal. Rounding the first two components to integers results in a total amount of $15 + 15 + 1$ bits of information, which is stored in a 32 bit integer. Thus, the total amount of memory required for the data of one cell is $(12 + 4 + 4 + 4)$ Bytes = 24 Bytes. The reset information is required to rebuild the cache when illumination changes occur. Note that the diffuse illumination is not attenuated by the diffuse material color (albedo) at this point. Instead, albedo is accounted for after reconstruction, which allows for a higher-quality representation of spatial variation in the

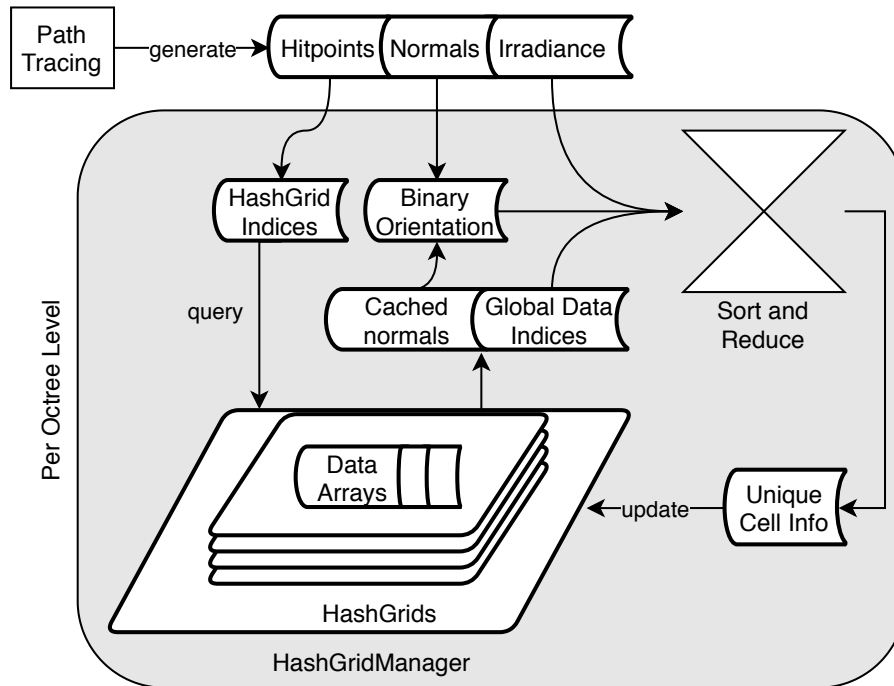


Figure 5.2: An overview of the caching process. Hitpoints, normals, and irradiance samples are generated by the path tracing process. Based on the hitpoint coordinates, the HashGrid indices for each hitpoint are then determined. Using these indices, the HashGridManager is queried for the global indices to the data arrays and the cached normals. A binary orientation for each irradiance sample, the actual irradiance sample, and the global data indices are now utilized to collate the data in a sort-and-reduce operation. Finally, the resulting unique information per grid cell is then merged with the current cache data.

appearance of diffuse surfaces. In order to determine the front normal of each cell, which is required to discern the stored orientations of each cache cell, an atomic compare-and-swap is used to store the current normal in a cache cell if no normal is stored so far. All generated samples can then be assigned to the front or back by comparing their stored normals with the front normal.

There are no specific constraints for the number of triangles per octree cell (or, vice versa, the number of octree cells per triangle), as the required resolution largely depends on the lighting situation and the actual camera settings and position. For quick previews during the modeling process of individual objects, lower resolutions such as 256^3 or 512^3 , may already yield satisfactory results.

5.2.2 Caching

While Figure 5.2 already gives a general overview of the process described in this section, including the general data flow and algorithmic elements, a further description is given below. During the caching process, rays are shot into the scene from the

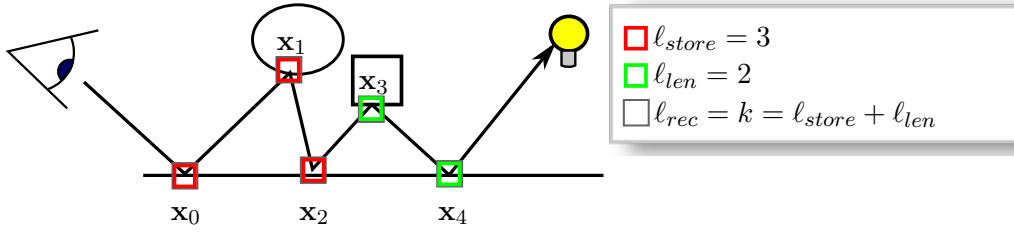


Figure 5.3: Parameters for a single path. The parameter ℓ_{store} determines the maximum depth to which values are stored in the cache. After that depth, the illumination along subpaths is computed up to a maximum length of ℓ_{len} . The length of both determines the maximal recursion depth ℓ_{rec} .

current point of view and traced along randomly generated paths $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$, with \mathbf{x}_i being that path's individual vertices located on scene surfaces, and $k = \ell_{rec}$ being the maximum recursion depth. As we want to cache data not only for the first hitpoint (which would effectively only represent directly visible geometry), we compute illumination along subpaths with a maximum length of ℓ_{len} and store these for the first ℓ_{store} hitpoints. Thus, since all vertices of a path should account for the energy transported along the same number of consecutive vertices in order to provide consistent data, the maximum path length is $\ell_{store} + \ell_{len}$, and the indirect illumination contributed to each vertex \mathbf{x}_i along the path has to be limited to the subpath vertices $\mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+\ell_{len}}, i + \ell_{len} \leq \ell_{rec}$. This is illustrated in Figure 5.3.

As soon as the local illumination and the reflected direction ω_i for the current vertex \mathbf{x}_j have been computed, the energy transported along the current path is updated by computing the throughput $\mathbf{T}_i = f_r(\mathbf{x}_i, \mathbf{x}_j) / \text{prob}(\mathbf{x}_j)(\omega_i \cdot \vec{n})$ according to the locally evaluated bidirectional reflectance distribution function (BRDF). The first vertex along the current path that should still account for energy originating at the current vertex is at index $p = \max\{0, j - \ell_{len}\}$. In order to take into account the accumulated throughput for the current subpath from vertex \mathbf{x}_j back to vertex $\mathbf{x}_{j'}$, each preceding vertex $\mathbf{x}_{j' < j}$ is updated with the reflected local energy \mathbf{L}_j by computing the Hadamard product

$$\mathbf{E}_{j'} = \mathbf{L}_j \circ \frac{\mathbf{T}_{j'}}{\mathbf{C}_{j'}} \circ \prod_{m=j'+1}^j \mathbf{T}_m. \quad (5.5)$$

Here, the diffuse material color (or *albedo*) $\mathbf{C}_{j'}$ is not accounted for in vertex $\mathbf{x}_{j'}$. It is instead taken into account after reconstruction in order to avoid loss of spatial variation in the appearance of diffuse materials. All vertices from each path that belong to a Lambertian material are stored in the respective arrays indexed by the hash map. This includes diffuse illumination values \mathbf{E}_j , compressed normal vectors \vec{n}_j , the linear map index \mathcal{H} (only required if the hash map's resolution exceeds 1024^3), and the linear cell index \mathcal{C} , where \mathcal{H} and \mathcal{C} are necessary to store the data

in our data structure correctly.

Now, the respective cells of the HashCache are updated with the newly computed light transport data. When updating the individual octree levels, the collected data is pre-accumulated before performing an update on the global data arrays in order to avoid synchronization issues. Pre-accumulation is implemented by first sorting the data using a radix sort approach and consecutively performing a reduction on the data with the global data index as the primary key and the binary orientation information (front or back) as the secondary key for both the sorting and reduction. In order to use the orientation information as the secondary key in the reduction, the individual sample's normal vectors have to be replaced with binary front/back information: 1, if the sample lies within the front-facing hemisphere, and -1 otherwise. If storage is not an issue, more directions can be represented, which may also allow for caching slightly glossy materials. Afterwards, the data is coarsened for the preceding octree level and the process is repeated until all levels have been updated. The full octree update is in $\mathcal{O}(n \log n)$, with n being the number of updated cache cells.

5.2.3 Reconstruction

As rendering scenes with Lambertian materials exclusively may cause them to appear visually dull and unrealistic, our system provides the means for handling materials with glossy or specular properties. An overview of the process described in this section is given in Figure 5.4, while a further description is given below.

For the reconstruction step, primary geometry hitpoints are determined for each individual pixel, with the exception of glossy and specular materials, where the specific rays are traced further until they eventually arrive at maximum depth ℓ_{rec} , a diffuse material, or hit the background. For each path, the accumulated throughput is stored for the first ℓ_{p-1} vertices as $\mathbf{T}_{acc} = \prod_{i < \ell_{p-1}} \mathbf{T}_i$ together with the diffuse material color, the local normal, and the appropriate octree level (selected by projecting the pixel area in object space). The reconstruction is executed per-level and accumulated in the image by selecting the correct orientation from each cell and multiplying the retrieved diffuse illumination value with $\mathbf{T}_{acc} \cdot \mathbf{C}_{\ell_{p-1}}$.

In order to reduce the blocky appearance caused by low cache resolutions, we employ a spatial jittering method to compute the actual cell index. This jittering method is based on the hitpoint's local tangent plane:

$$\mathbf{p}' = \mathbf{p} + s_c \cdot s_j \left(\xi_0 \cdot \vec{t} + \xi_1 \cdot \vec{b} \right). \quad (5.6)$$

Here, \mathbf{p} and \mathbf{p}' are the original and the jittered hitpoints, $\xi_i \sim U(-1, 1)$ are uniformly

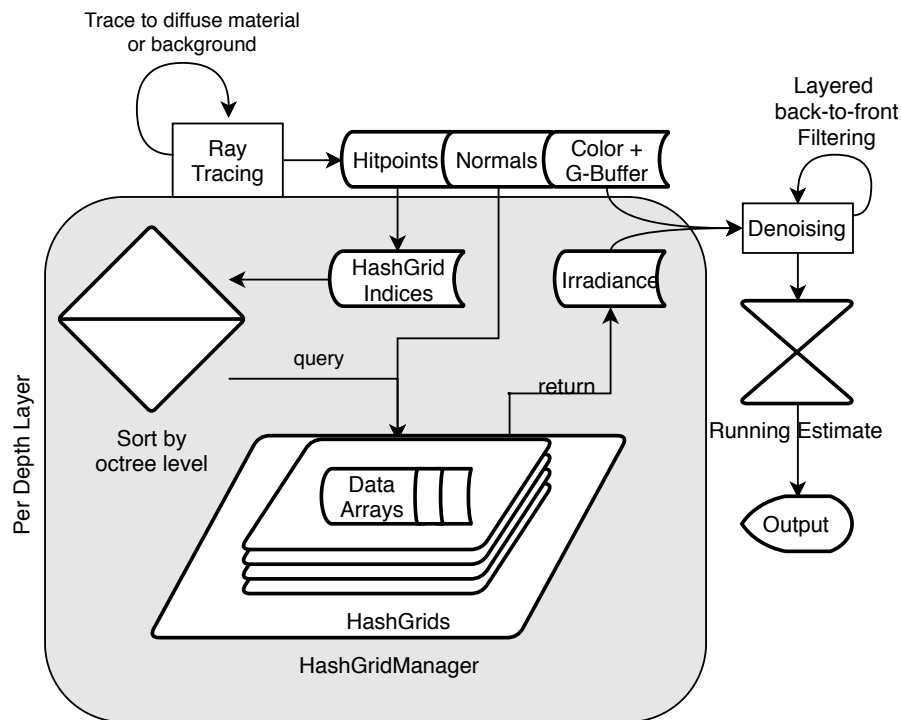


Figure 5.4: An overview of the reconstruction process. A ray tracing step is performed to find the actual hitpoints that have to be reconstructed from the cache. In order to support non-lambertian materials, this step includes tracing of rays until a diffuse surface is found, the background is hit, or a user-definable maximum recursion depth is reached. The HashGrid indices computed from the hitpoints are then sorted by the appropriate octree level, and the cache is queried for the actual data indices. Together with hitpoint-wise texture and geometry information, the acquired irradiance is filtered in a denoising step in a layered back-to-front-way. This means that each recursion level is filtered individually and then combined with the next (closer) level, until the primary hitpoints are reached. The result is then combined in a running estimate to achieve higher image quality when the camera is not moving.

distributed random numbers in $[-1, 1]$, \vec{t} and \vec{b} are the tangent and the binormal, s_c is the actual cell size, and s_j is the user-adjustable scale of the jittering. Finally, a basic edge-aware cross-bilateral denoiser filters remaining noise for each depth layer individually, and also fills gaps with neighboring information where cache information is not available.

Figure 5.5 shows the effect of jittering and denoising in two areas: While the wall in the back shows more high-frequency noise, the statue in the front reveals quantization artifacts due to the great differences between neighboring cache values. However, such artifacts are efficiently removed by the spatial jittering.

Note that spatial jittering may result in slight artifacts when cells are processed which do not have geometry in all neighboring cells that lie on the respective tangent plane, as shown in Figure 5.6.

This is mainly caused by the fact that our data structure does not support enhanced sparsity encoding, but rather relies on constrained access (Lefebvre and Hoppe 2006) to avoid further memory consumption. Two cases may appear:

- (1) The hash key for the neighboring grid cell may belong to another cell that belongs to the scene's geometry. In such a case, visual artifacts may occur.
- (2) The hash key for the neighboring grid cell may yield an empty entry in the hash map. In this case, the irradiance value is set to the average of the pixel's neighbors, i.e., invalid or unsampled pixels resulting from spatial jittering are filled in.

However, during our evaluation, we did not observe any major artifacts resulting from this. Thus, we decided not to include any way of querying a cell for its grid coordinates.

In Section 5.2.4, it is described how we extend our system with a filtering approach that accounts for multiple bounces of glossy and specular reflections and refractions in order to improve visual quality. Note that in our implementation, caching and reconstruction are independent of each other. The caching process can be executed with an arbitrary sampling scheme at freely selectable resolutions, while the reconstruction just retrieves the stored illumination values from the data structure. Thus, the caching process may actually rely on arbitrary distributions of rays throughout the scene, which also enables strategies like randomly or adaptively sampling the scene along camera paths or creating importance-based sampling schemes. As for the sampling along camera paths, this may be a viable approach for accelerating the rendering of fly-throughs with fixed camera paths. Where and how to sample specifically to optimize performance has to be analysed thoroughly, though, as just randomly casting samples along a path will impact performance due to the reduced

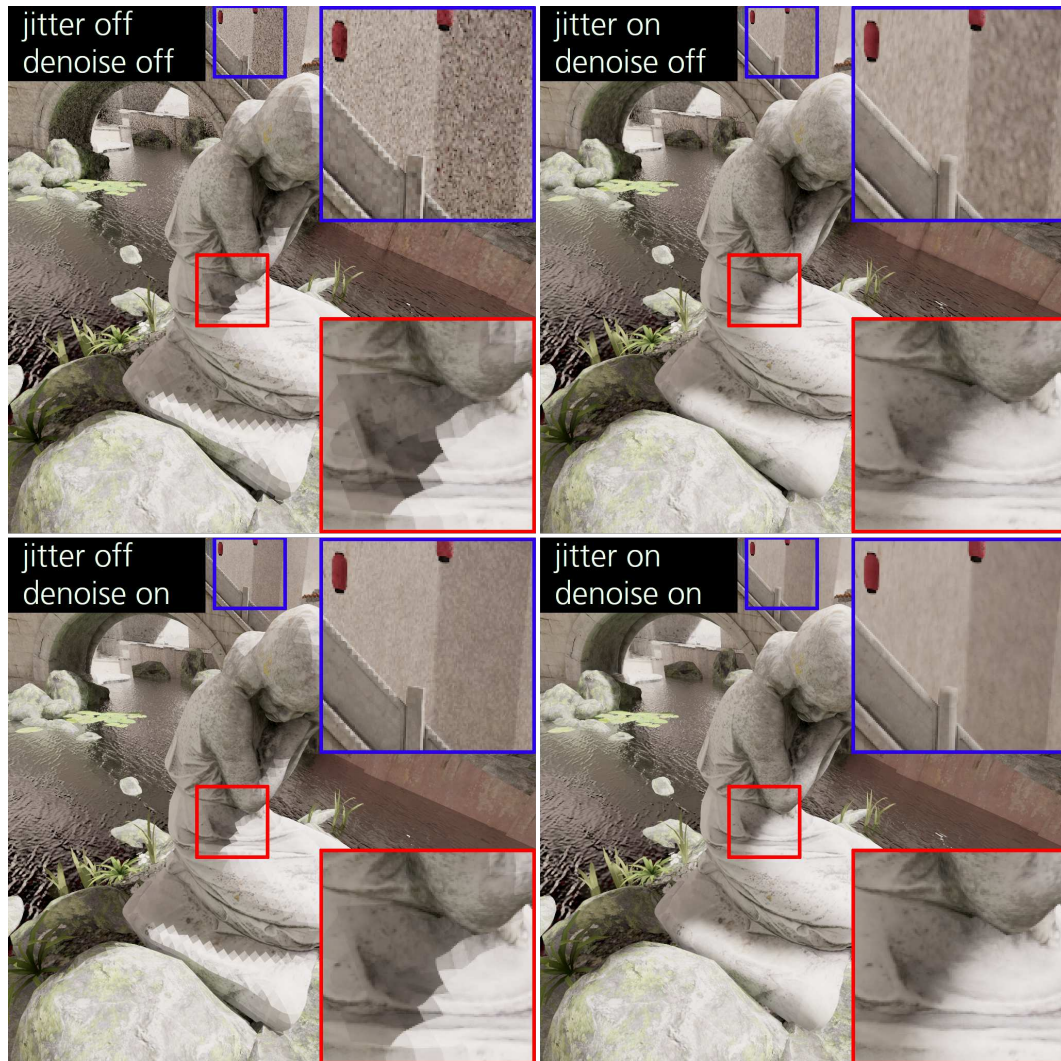


Figure 5.5: **(Top left)** HashCache-based reconstruction without any spatial jittering or denoising applied. **(Top right)** While spatial jittering cannot get rid of the high-frequency noise visible in the upper inset, it works well for hiding quantization artifacts. **(Bottom left)** Denoising alone does work well for fine-grained noise, but cannot remove quantization artifacts very well. **(Bottom right)** Combining jittering and denoising works well for both kinds of artifacts.

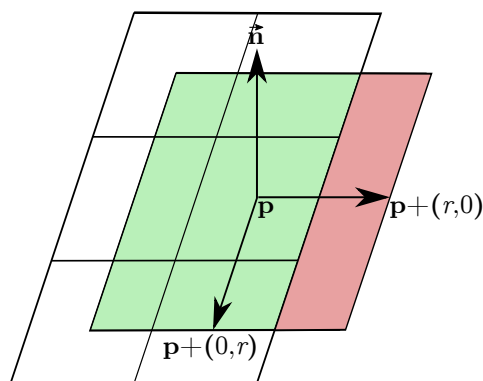


Figure 5.6: Two dimensional example of the possible issue with spatial jittering: For a hit-point \mathbf{p} on an arbitrary scene surface with a jitter radius of r , the sampled cell resulting from $\mathbf{p}' = \mathbf{p} + \vec{j}$, $J = \{\vec{j} | \vec{j} \in [-r, r]^2\}$ may not contain any scene geometry. Consequently, the according cell has no memory explicitly reserved in the hash map. The computed hash key may thus lead to empty or even plainly wrong cells. The filled area represents the set of possible jittered points. The green area is the subset of points that result in valid keys, while the red area depicts the subset of invalid points.

ray coherence. Additionally, the separate caching step can be performed with arbitrary numbers of samples. In our case, we tested the caching performance at various resolutions, as shown in Section 5.3.

5.2.4 Layered Filtering

While the plain octree reconstruction described so far may suffice in some scenarios, it is known from regular path tracing that convergence can be slow in many scenes. This makes it necessary to employ filtering methods in order to achieve noise-free images within an acceptable time frame. The kind of noise remaining in the generated images largely depends on the employed rendering method, while quantization artifacts are effectively reduced by the aforementioned spatial jittering method. However, it is important to note that the existing filtering methods developed for path tracing will not work for the kind of noise our approach exhibits, as noise scales with the distance to a surface because of the limited cache resolution. Our suggested filtering approach aims to increase visual fidelity under such circumstances, while explicitly accounting for glossy, specular, and refractive materials.

The main idea of our method is to split the traditional filtering step that is carried out on the final image into multiple steps by filtering each bounce of light in an individual layer (cf. Figure 5.7). With a non-layered filtering approach, the scene information available to the actual algorithm is limited to the first hitpoint, and multiple bounces between reflective and refractive materials have to be processed

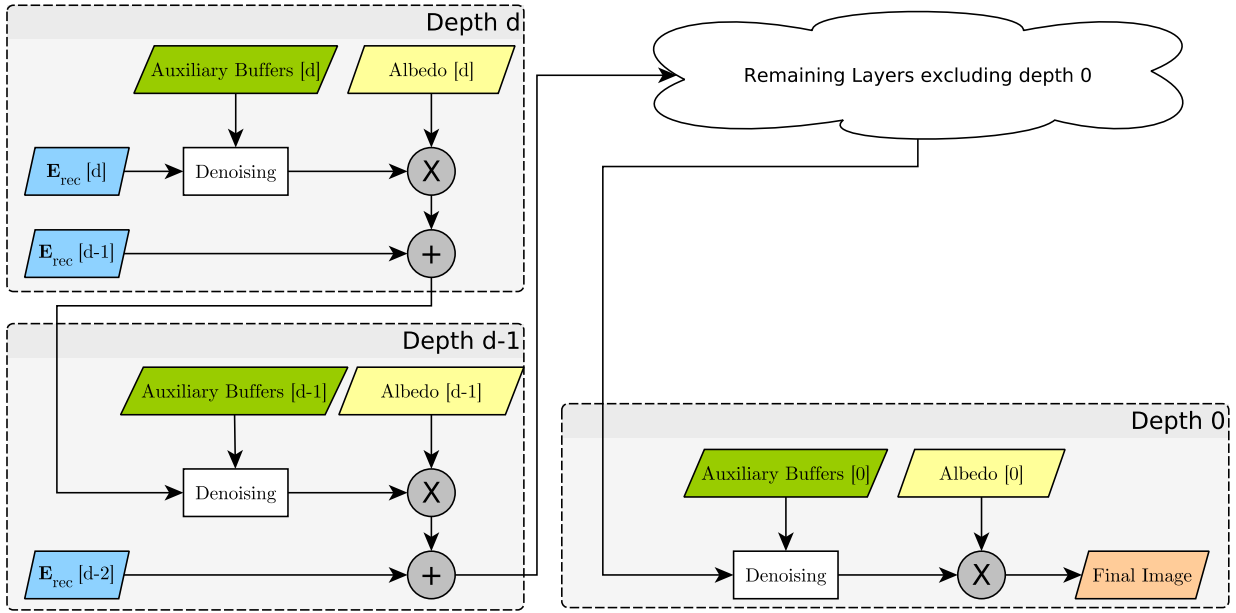


Figure 5.7: Layered filtering process. The reconstructed illumination \mathbf{E}_{rec} for hitpoints that reached cached materials is added to the current result at each level. Together with the auxiliary buffers, such as the local G-buffer, it is then processed by the denoising method. Afterwards, albedo is taken into account, and the data is propagated to the next layer. This process is repeated until the first layer, formed by the primary hitpoints, is reached, and the final image is reconstructed.

with the information at hand. This may result in a loss of detail or the need for more samples in order to achieve satisfactory results.

At the core of our layered filtering approach is an image space denoising filter that needs to be able to deal with noise appearing at varying scales. In this exemplary case, we used a slightly extended cross-bilateral filter with a sparse sampling pattern based on the voxel filtering technique presented by Laine and Karras (2011). In contrast to proposing a concrete filtering method, we present a layered filtering *framework*, which is inherently independent of the filtering method used. This way, more recent filtering approaches could also be integrated and adapted to further improve the results. In order to provide the necessary information to this filter for each bounce of illumination, we expand upon the stored data described in Section 5.2.3. For each vertex $\mathbf{x}_i^{(p)}$ belonging to the path $\bar{\mathbf{x}}^{(p)}$, the following information is stored:

- Path segment length: $D_i = \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$,
- Accumulated path length: $D_i^\Sigma = \sum_{j \leq i} D_j$, serving as an extended depth buffer,
- Geometric normal for the current vertex: \vec{n}_i ,
- Throughput for the next generated ray \mathbf{T}_i ,

- Shininess of the current material α_i ,
- Reconstructed diffuse illumination \mathbf{E}_i ,
- Diffuse material color (albedo): \mathbf{C}_i .

The edge-stopping functions we use for the bilateral filter are defined as follows:

- Normals: $w_n = s_n \max\{0, \vec{n}_p \cdot \vec{n}_q\}$,
- Path length: $w_d = 1 - s_d \Delta D^\Sigma$,
- Path segment length: $w_{d'} = 1 - s_{d'} \Delta D$,
- Albedo: $w_c = 1 - s_c \Delta c / (\Delta c + 1)$,
- Luminance: $w_l = \max\{0, 1 - \log(1 + s_l \Delta l / (\Delta l + 1))\}$,

with

$$\begin{aligned}\Delta D^\Sigma &= |D_p^\Sigma - D_q^\Sigma|, \\ \Delta D &= |D_p - D_q|, \\ \Delta c &= \log(1 + \kappa \|c_p - c_q\|) / \kappa, \\ \Delta l &= \kappa |l_p - l_q|.\end{aligned}$$

As each bounce is filtered individually and the results are propagated towards the preceding bounce, we omit the depth index i from the description of the edge-stopping functions. The pixel indices are denoted p and q . Shininess is required to distinguish diffuse from glossy materials and parameterize the filter adequately. Albedo is required for preserving texture details, while throughput is required to apply the layered approach to non-diffuse materials properly. The parameters $s_{\{n|d|d'|c|l'\}}$ are user-defined scaling factors for normals, accumulated depth, local depth, color differences, and luminance differences, respectively. The color- and luminance-based edge-stopping functions have an additional weighting factor κ . The color difference is computed in L*a*b* color space. Generally, user-definable parameters have been chosen empirically by the best subjective visual impression. The values used for the evaluation are mentioned in Section 5.3. Note that non-lambertian materials have a separate weighting factor for the color difference, which is not explicitly mentioned here.

All aforementioned information is available on a per-pixel, per-layer basis, which means that there is an actual image per light bounce (which we refer to as a layer). While later bounces of individual paths may arrive at different points in the scene, this is partially accounted for by using the accumulated ray depth as a filter guide.

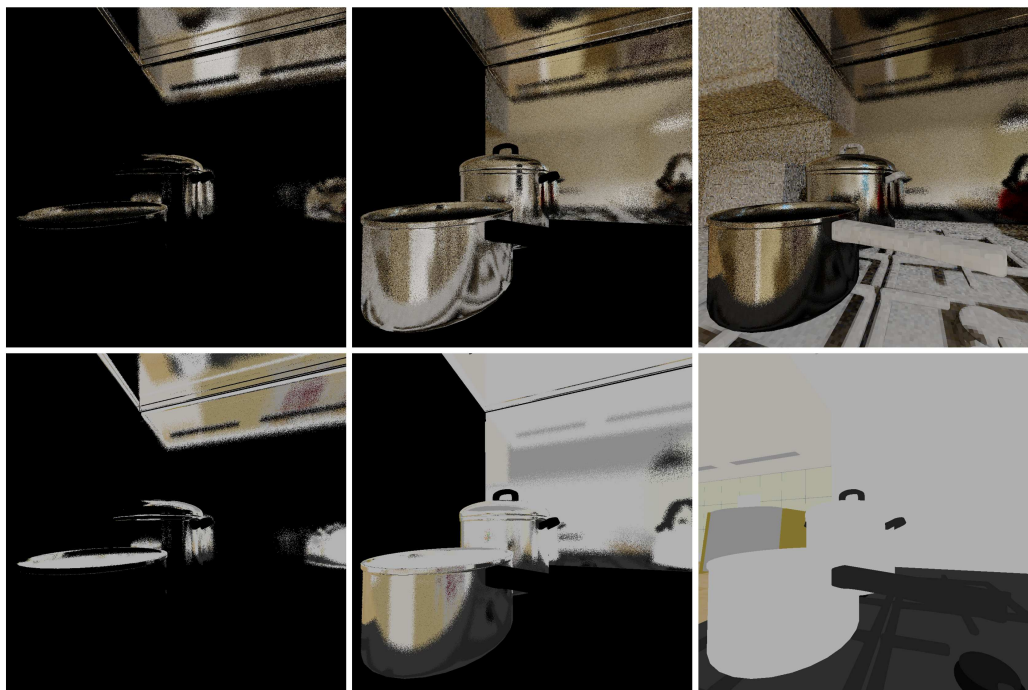


Figure 5.8: Top: Unfiltered propagation of light bounces from the third hitpoint to the primary hitpoint (left to right). Bottom: The diffuse material colors (albedo) of each bounce above, multiplied after each filtering step. Note that non-diffuse materials appear white because they do not change the appearance of cached illumination values, which are strictly diffuse. The grey appearance of these materials resulted from tonemapping.

Although this yielded satisfactory results in our tests, there may be scene arrangements and material properties that cause this to be an issue. In such cases, we suggest using hitpoint world coordinates or HashGrid cell coordinates as possible additional or alternative filter guides.

Consequently, we process this data in a per-layer fashion, starting at the maximum bounce, filtering the result, and propagating it to the previous bounce. Each time the filtered result is propagated from layer i to $i-1$, it is multiplied with \mathbf{T}_{i-1} to account for the actual path throughput. The result is then added to the reconstructed diffuse illumination \mathbf{E}_{i-1} , and the accumulated image is filtered and propagated again until the primary hitpoints are reached. Additionally, after each layer has been filtered, it is multiplied with the local albedo \mathbf{C}_i in order to account for high-frequency content, as it is often contained in diffuse textures and shaders. Figure 5.8 shows the reconstructed diffuse illumination propagated from the third bounce to the secondary and then to the primary bounce.

Our approach to caching, filtering, and accumulating is essentially an approximate final gathering step split into two separate steps: For diffuse materials, the illumination is approximated by integrating the energy arriving at each octree cell in the



Figure 5.9: Interleaved comparison between filtering the first three accumulated bounces with traditional (red insets) and layered filtering (green insets). The image has been reconstructed with one sample per pixel after filling the cache with four samples per pixel just for the illustrated viewpoint. The utilization of local geometry information through layered G-buffers shows clear improvements in image quality. Filter weights for color and luminance differences have been adjusted in comparison to the weights chosen in the evaluation measurements.

caching phase. For all non-lambertian materials, rays are traced until a diffuse material is hit in the reconstruction step. Then, the pre-gathered diffuse illumination is queried from the cache at these points. This hybrid approach is directly supported by our layered filtering method, which makes it possible to filter the illumination gathered in the octree cells separately based on local scene information, even if it is only indirectly visible in an image. Figure 5.9 shows a comparison between traditional first-bounce-only and our own layered filtering approach for the first three bounces.

5.3 Results and Evaluation

In this section, we evaluate the visual quality, performance, and GPU memory requirements of the presented system. All measurements in this section were performed on a Linux system equipped with a GeForce GTX Titan X (Maxwell), a Core i7-7700 CPU, and 32 GiB of main memory. Scaling factors for the layered filtering were statically chosen to be $s_n = 1$, $s_d = 100$, $s_d' = 400$, $s_{c,diffuse} = 1.5$, $s_{c,glossy} = 3$, and $s_l = 3$. Images were rendered at a resolution of 1024×1024 pixels.

5.3.1 Visual Quality

The proposed methods in this chapter lead to an approximate reconstruction of the scene’s true illumination. To judge image quality here, we employ purely computational methods for estimating image quality. At the same time, perceptual considerations are taken into account by using MS-SSIM in addition to the purely pixel-based relMSE. To determine the visual error, we rendered the scenes Country Kitchen (CK) and Streets of Asia (SoA) using a camera fly-through of 500 frames with different configurations of the HashCache and regular path tracing for comparison. CK was chosen because the illumination results mainly from a lamp in the

room, but is also influenced by the environment map illuminating the scene through the window, which poses a rather difficult scenario for a standard path tracer. On the other hand, SoA was chosen because it can be sufficiently illuminated by exclusively relying on an environment map, which is a less difficult scenario when using path tracing. On the one hand, standard path tracing was rendered for 1, 8, 64, 512, 4096 and 16384 samples per pixel (SPP) for both scenes and additionally for 131072 SPP for CK, as lower sample counts still revealed visible noise, especially in shadowed regions. The outputs with the highest sample counts are used as reference images for error computation. On the other hand, results using the HashCache system were generated with 1, 8, 64 and 512 SPP with the presented reconstruction technique. The hash map resolution was chosen to be 4096^3 for CK and 2048^3 for SoA. All rendered images underwent the same tone-mapping process. The results illustrated in Figure 5.10 show a comparison of the visual quality measured as (top) the relMSE and (bottom) the MS-SSIM (Wang, Simoncelli, et al. 2003) for varying sample counts. MS-SSIM mimics the multi-scale processing of the human visual system (HVS) and is an important tool to judge the perceived image quality. The reuse of already computed information in combination with a cross-bilateral filtering method already leads to the HashCache at 64 SPP yielding a visual quality similar to path tracing at 4096 SPP for the scene CK, even coming close to the reference image at 131072 SPP in the middle of the sequence. For SoA, the HashCache does not show an improvement in relMSE between 64 and 512 SPP. At 64 SPP, the relMSE is similar to path tracing at 512 SPP. The scene SoA was rendered using a HashCache with a lower maximal resolution of only 2048^3 . As the spatial extent of the scene is high and the camera gets close to certain objects in the scene during the fly-through, quantization artifacts are likely to occur and cause a larger difference between the reference solution and the cached irradiance.

We expect even better quality at low sample densities when more advanced state-of-the-art filtering methods are integrated into our layered filtering framework. Nonetheless, they have to be adapted to the specific appearance of noise resulting from the HashCache (see Section 5.3).

Applying a denoising method that is developed for standard path tracing is not suitable because of the visual appearance of noise in the HashCache system, mainly because it appears at varying scales that may be larger than just individual pixels. See Figure 5.11 for an example of noise appearance in our system. However, applying an additional denoiser on top of our filter may improve image quality even further by removing the fine-grained noise more reliably, while allowing for more conservative settings in our own filtering process.

Figure 5.12 shows a visual comparison of the quality improvements by increasing the cache resolution.

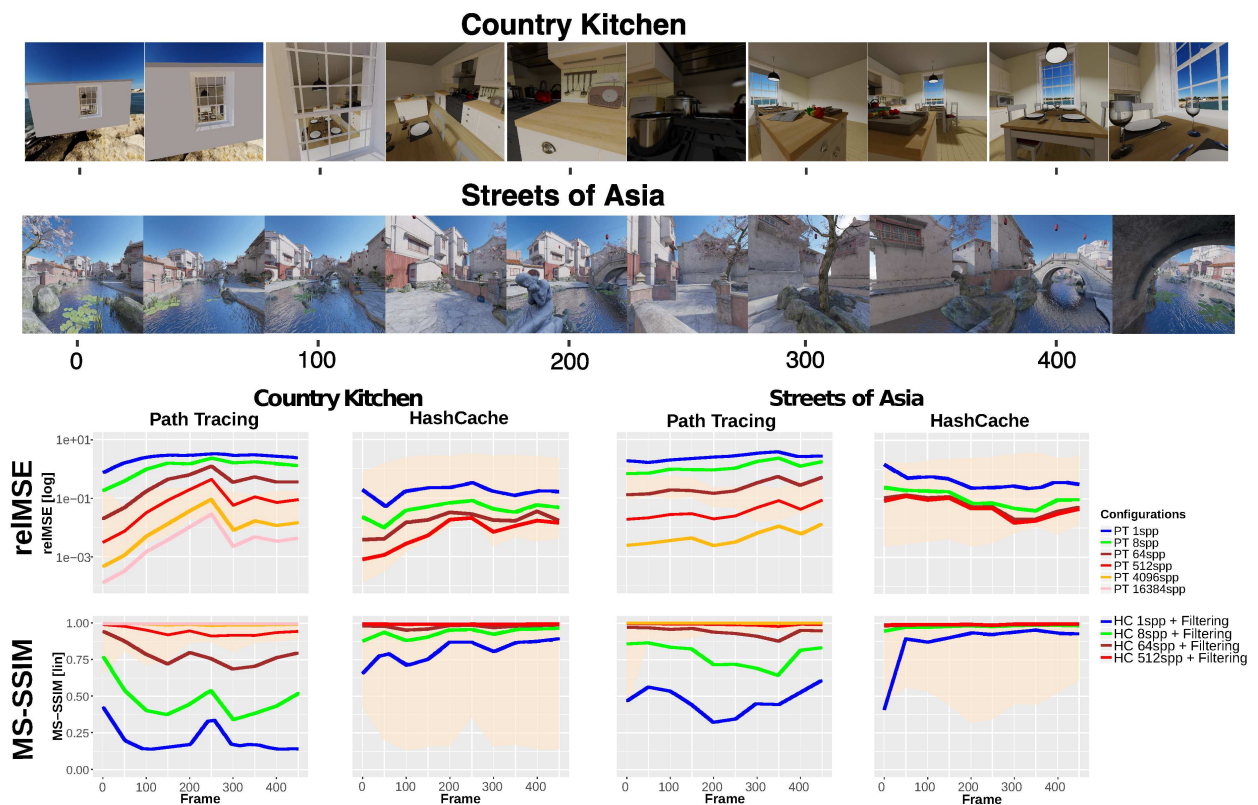


Figure 5.10: relative mean-square error (lower is better) and multi-scale structural similarity (higher is better) for the scene Country Kitchen (cache resolution 4096^3) and Streets of Asia (cache resolution 2048^3) at various sampling densities using path tracing and the HashCache system. The shaded area in the diagrams outlines the respective reIMSE and MS-SSIM ranges for the other rendering method for better comparison.

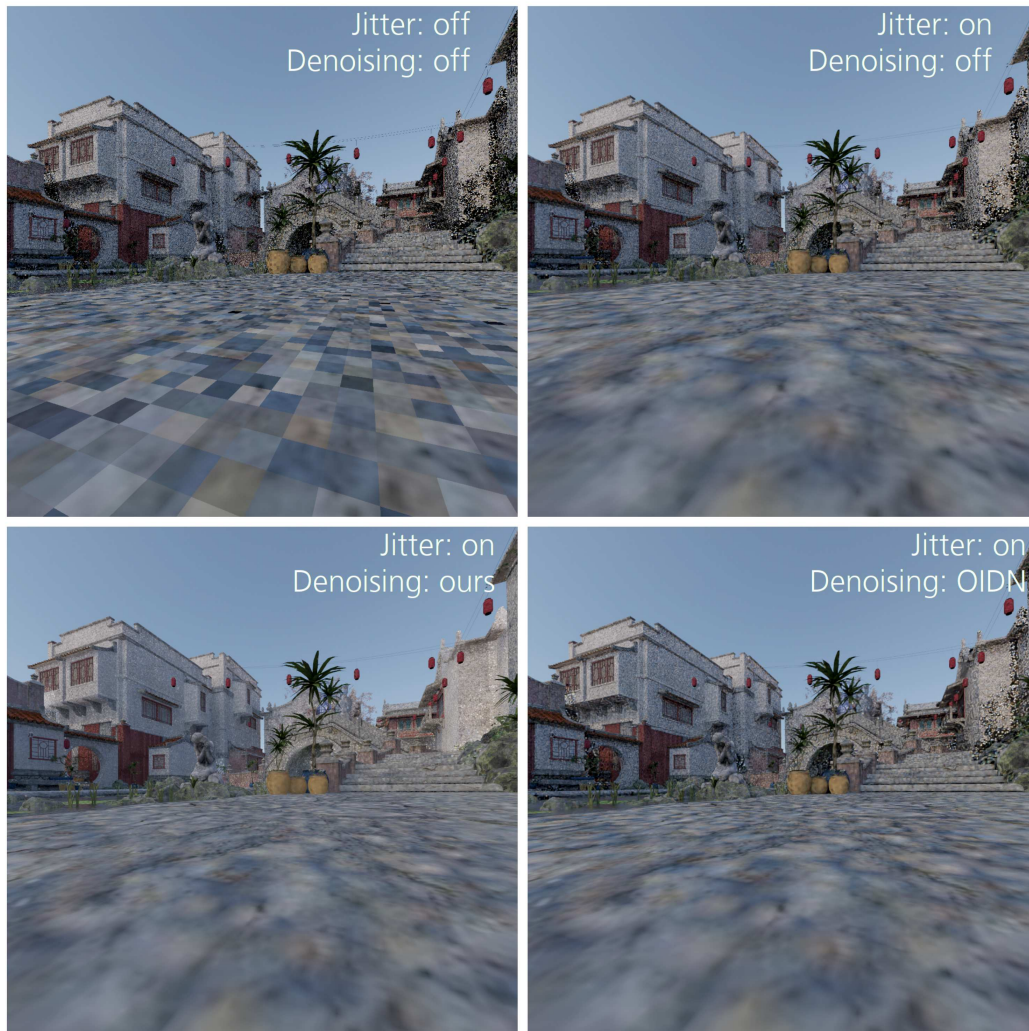


Figure 5.11: The appearance of noise when using the HashCache. Without filtering and jittering, the distance-dependent scale of noise becomes clearly visible. Turning jitter on improves the visual appearance of quantization artifacts. Turning denoising on clearly improves regions with low sample counts, even filling in areas where no samples have been cached at all. The bottom right image shows the result with Intel’s Open Image Denoising (<https://openimagedenoise.github.io/>) instead of our denoiser. It becomes clear that the method cannot cope with the kind of noise exhibited by the HashCache.



Figure 5.12: The effect of adjusting the cache resolution is especially visible at high-resolution details like the transition between shadows and illuminated regions. The rightmost pictures show the reference rendering at 131,072 spp. The rest of the images show cache resolutions from 2048^3 down to 256^3 , decreasing by powers of two (from right to left). It can be seen that the decreased spatial resolution leads to quantization artifacts which are effectively filtered by jittering and denoising. However, fine-grained details would not be resolvable at low resolutions, and a complete loss of spatial details may occur.

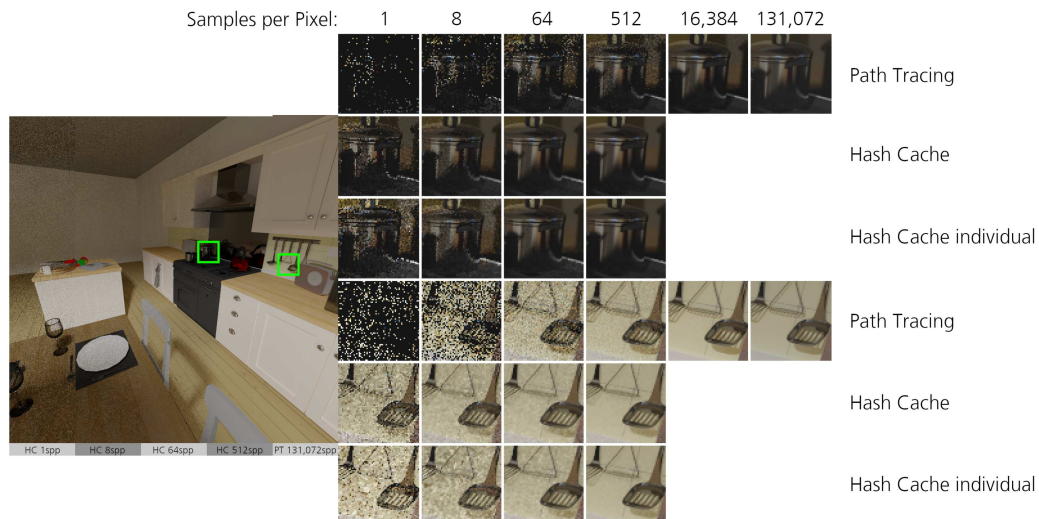


Figure 5.13: Comparison of images generated with pure path tracing and with the support of HashCache. **(Left)** The full image subdivided horizontally into areas rendered with HashCache at 1, 8, 64, and 512 spp, as well as with Path Tracing at 131,072 spp. Insets in green are magnified on the right for the various settings. **(Right)** Zoomed-in rendering for comparison, rendered with Path Tracing, HashCache (with three more frames in the sequence rendered before), and HashCache individual (the camera position has been rendered without filling the cache in the preceding frames). For the upper inset, the HashCache system already yields a quality at 64 spp (512 spp for individual rendering) that is on-par with the path-traced image at 131,072 spp. However, the lower inset shows more artifacts than the reference, even at 512 spp, for the HashCache rendering. The main reason for this is that we tuned the denoising filter to maintain shadows. Different denoising methods used with our layered framework should be able to resolve this well.

Figure 5.13 shows a comparison of images from three rendering modes: Pure path tracing, HashCache rendering an image sequence, and HashCache rendering an individual frame. Glossy reflections appear at a high quality early in the process, and details in such reflections are preserved very well.

Figure 5.14 shows how well our system performs with a reflective surface. Due to the layered filtering approach, a quality similar to the reference rendering is already achieved with a fraction of the samples required for regular path tracing. The effect of layered filtering for glossy and specular (refractive) materials is shown in more detail in Figure 5.15. Overblurring of details in refractions and reflections is avoided by the layered approach. Instead, the diffuse surface hit by the reflected and refracted rays is filtered in its own layer, and it is then propagated along the path.

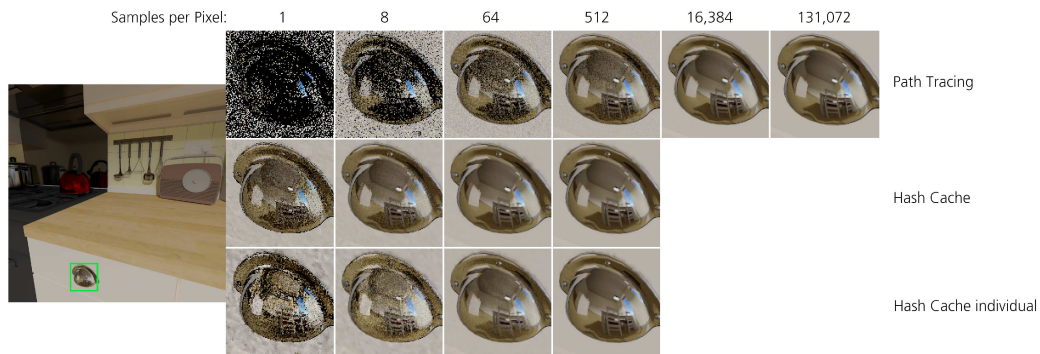


Figure 5.14: Comparison of images generated with pure path tracing and with the support of HashCache. **(Left)** The full image rendered at the reference sample count of 2^{17} . The green inset is magnified on the right for the various rendering settings. **(Right)** Zoomed-in rendering for comparison, rendered with Path Tracing, HashCache (with four more frames in the sequence rendered before as a warm-up phase), and HashCache individual (without filling the cache in the preceding frames). For the upper inset, the HashCache system already yields a quality at 64 spp (512 spp for individual rendering) that is at least visually on-par with the path-traced image at 131,072 spp.



Figure 5.15: The effect of layered filtering for glossy and specular materials. The cache was filled irregularly over 128 frames at 1 spp for a random fly-through. **(Left)** Unfiltered image at 128 reconstruction samples per pixel. **(Center)** Layered filtering at 128 reconstruction samples per pixel. **(Right)** Magnified insets. It is clearly visible how the noisy illumination from the back wall is reflected in the glossy surface on the ground when layered filtering is deactivated. With layered filtering, the noise can be filtered in a separate layer and then be accounted for in the glossy reflection. For the refractive material on the right, a similar effect is visible: Refracted rays pick up the noisy cache values from the back wall when layered filtering is deactivated. With layered filtering turned on, this noise can be effectively filtered before it is propagated to preceding bounces. This allows for removing the noise locally without loss of details. An example for the individual layers being filtered is shown in Figure 5.9.

5.3.2 Performance

The rendering times and the average number of samples cached per frame for two different cache settings, ($\ell_{store} = 3, \ell_{len} = 3$) and ($\ell_{store} = 8, \ell_{len} = 8$), are shown in Figure 5.16. The caching resolution factor (CRF) serves to adjust the number of rays cast in the caching phase. If set to 1, the number of primary rays will be the same as the chosen image resolution in the reconstruction step. While the cache construction times behave quadratically with regard to the CRF (because resolution is multiplied with the CRF both horizontally and vertically), reconstruction times do not depend on this setting.

The average filtering time measured for the bilateral filter is 22 ms, while the cache query took 4.5 ms and the ray tracing phase for determining the visible geometry took around 31 ms. Thus, at a frame rate of around 17 fps, the reconstruction itself is well-suited for interactive previews.

The additional time taken by the caching procedure depends largely on the rendering and caching parameters. Setting low values for ℓ_{store} and ℓ_{len} causes the ray-tracing step to work not only with shorter, but also more coherent paths, which is beneficial for its performance. However, despite the top configuration delivering the fastest caching times, the number of updated cache cells on the right also shows that cache convergence should be expected to be relatively slow.

Only storing the first vertex for each path also causes issues with glossy, specular, or transparent materials where the positions of subsequent vertices have not yet been seen directly by the user. Imagine a white wall reflected inside a mirror. It will only show a correctly rendered reflection if it has already been viewed before directly. However, this issue can be easily avoided by choosing a higher setting for ℓ_{store} , which is desirable anyway in order to cache illumination for scene parts that are not directly visible.

The relatively slow cache update times for high CRF settings and high recursion and storage settings visible in Figure 5.16 are not a real issue for the interactive exploration; in order to keep the process interactive, we modified our implementation to adjust the CRF to maintain a certain framerate while the camera is moving and only perform caching at the full resolution in the absence of movement.

As shown in Figure 5.17, the actual reconstruction time is low enough to allow for fully interactive camera movements when the CRF is lowered. While the initial construction of the cache may take a couple of seconds depending on the set resolution, it is faster than Radiance Caching approaches (Krivanek, Gautron, et al. 2005; Omidvar, Ribardi re, et al. 2015). Yet, this means that the HashCache is too slow to support fully dynamic scenes. It may be possible to omit the necessity of an

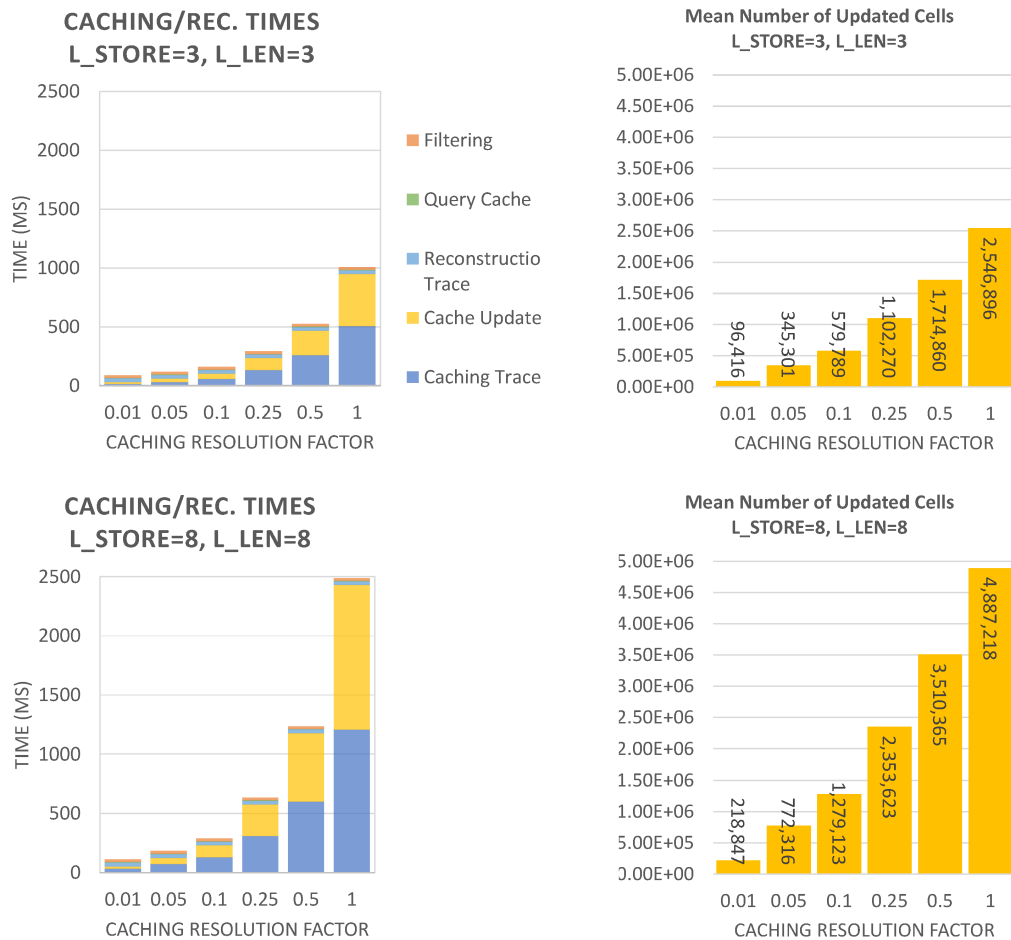


Figure 5.16: Statistics for the scene Country Kitchen with different cache settings at a resolution of 2048^3 . **(Left)** Total rendering times, split into different steps. It becomes clear that the tracing and cache update steps take by far most of the time, while the cache query can almost be disregarded entirely. **(Right)** Average number of cells updated per frame of the 500 frame sequence. These numbers include the update of all octree levels. Note that the resolution factor can be adjusted dynamically to the current situation, e.g., when the user is moving.

explicit hash map construction step by using non-perfect hashing schemes in future research.

5.3.3 Memory Requirements

The amount of required GPU memory for both CK and SoA is shown in Table 5.1. While data density decreases by a factor of roughly 0.5 from level i to $i + 1$, memory requirements still increase by a factor of roughly 4 to 5. In total, the 4096^3 representation of CK required an amount of 2.17 GiB for the data arrays and 406.92 MiB for the hash maps, while SoA required 691.37 MiB for the data arrays and 126.75 MiB for the hash maps at a total resolution of 2048^3 . One possible approach to handling the memory requirements resulting from higher resolutions is the integration of an out-of-core component into our system, dynamically loading currently required data from host memory into GPU memory.

5.3.4 Comparison to State-of-the-Art

Methods such as the work by Schied, Kaplanyan, et al. (Schied, Kaplanyan, et al. 2017; Schied, Peters, et al. 2018) have lower filtering run times (e.g., 4–5 ms on Titan X vs. 22 ms for the unoptimized HashCache filter) and produce a high visual quality, but cannot simply be applied to HashCache renderings because of the different appearance of noise. In addition, these techniques rely on strong temporal coherence between subsequent views. In contrast, the HashCache allows for integration of GI data from arbitrary spatial locations. Thus, we are certain that all of these techniques can benefit from the knowledge from world space caches. With the HashCache’s hybrid reconstruction method, specular materials can be supported with ease (see Figure 5.14). Notwithstanding, querying the world space cache is slower than a cache in image space (4.5 ms for HashCache vs. ~ 0.5 –1 ms for Schied, Peters, et al. (2018)). Yet, in contrast to techniques such as NVIDIA’s machine learning solution presented by Chaitanya, Kaplanyan, et al. (2017) that needs specific training data or might produce inconsistent results, the HashCache can be filled at run time. Admittedly, the caching itself is a costly operation (see Figure 5.17), but the CRF can be freely adapted. The Figure also shows how the actual camera settings may influence caching and rendering times. For CK, rendering at the beginning takes only a little time because the camera is still outside the room, which means that a lot of rays actually hit the background. As the camera gets closer to the room, recursion depth increases because the rays are reflected between surfaces a lot more often. For the reconstruction, there is no vast difference caused by such a scenario. The only increase in recursion depth is caused by non-diffuse materials.

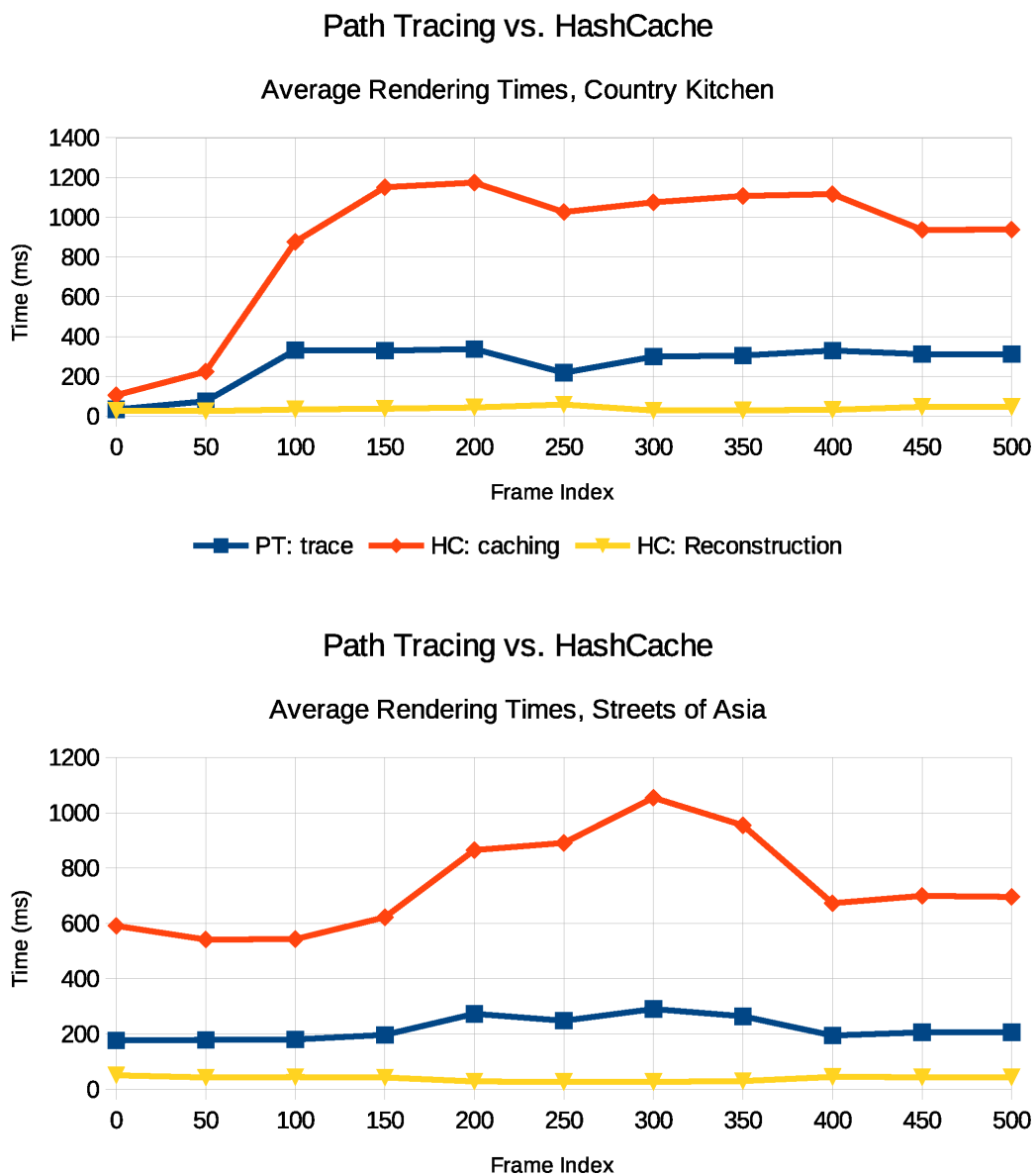


Figure 5.17: Path-tracing and HashCache rendering times for Country Kitchen and Streets of Asia fly-throughs. Rendering time is given for 1 spp. *PT: Trace* is the time for pure path tracing at a maximum recursion depth of 8, *HC: Caching* is the caching part of HashCache computations which includes path tracing and cache updates, and *HC: Reconstruction* is the reconstruction part of the HashCache computations, which includes ray tracing, fetching the respective data from the cache, and reconstructing an image from the computed data (including filtering). While caching times are significantly higher than pure path-tracing times, data is reused between frames so that the actual caching resolution factor (CRF) can be reduced either permanently or deactivated during user input, allowing for smooth interaction.

Country Kitchen				
Level	Res	Density	Data Mem.	Hash Mem.
0	1	1	24 B	4.4 B
1	2	0.5	96 B	17.6 B
2	4	0.375	576 B	105.6 B
3	8	0.193	2.32 kiB	435.6 B
4	16	0.124	11.93 kiB	2.19 kiB
5	32	0.070	54.07 kiB	9.91 kiB
6	64	0.038	234.47 kiB	42.99 kiB
7	128	0.023	1.11 MiB	209.23 kiB
8	256	0.012	4.66 MiB	874.83 kiB
9	512	0.006	19.43 MiB	3.56 MiB
10	1024	0.003	81.60 MiB	14.96 MiB
11	2048	0.002	362.09 MiB	66.38 MiB
12	4096	0.001	1.71 GiB	320.90 MiB
Sum			2.17 GiB	406.92 MiB

Streets of Asia				
Level	Res	Density	Data Mem.	Hash Mem.
0	1	1	24 B	4.4 B
1	2	0.5	96 B	17.6 B
2	4	0.266	408 B	74.8 B
3	8	0.197	2.37 kiB	444.4 B
4	16	0.135	12.91 kiB	2.37 kiB
5	32	0.074	57.14 kiB	10.48 kiB
6	64	0.043	265.92 kiB	48.75 kiB
7	128	0.025	1.21 MiB	226.35 kiB
8	256	0.014	5.50 MiB	1.01 MiB
9	512	0.008	25.68 MiB	4.71 MiB
10	1024	0.004	114.77 MiB	21.04 MiB
11	2048	0.003	543.88 MiB	99.70 MiB
Sum			691.37 MiB	126.75 MiB

Table 5.1: Data densities and memory requirements. The data density is the quotient of occupied cells and the actual number of cells for a full grid of resolution n^3 . Data memory is the amount of memory required to store the full data arrays. Hash Memory is the amount of memory reserved for the hash map representation for the respective level.

Once caches are filled to a certain extent, it is possible to limit the CRF largely and thus significantly reduce the cache update times. Moreover, the run time and caching behaviour can be adapted dynamically to the user’s requirements to stay within certain frame rate limits.

5.4 Discussion and Conclusion

We presented a method for caching diffuse global illumination in a hash-based data structure and reconstructing diffuse and non-diffuse illumination from the cached data. Performance-wise, the reconstruction process only depends on determining the correct hitpoints for each pixel, which is a ray tracing process. For diffuse materials, data is directly reconstructed from the HashCache, while non-diffuse materials allow

for tracing the rays further through the scene and performing reconstruction at the first diffuse hitpoints. This allows for supporting arbitrary non-diffuse materials as well.

To reduce memory requirements, the employed hashmaps rely on 32bit keys, which effectively limits the size of an octree based on such hash maps. For octree resolutions above 1024^3 , we split the scene into multiple octrees that are individually managed by the HashGridManager. As even higher octree resolutions still exhibit quantization artifacts (cell borders) when visualized directly, we employed a spatial jittering strategy that effectively approximates trilinear filtering temporally, but avoids having to store additional structural information that would increase memory requirements. Visual artifacts that may result from this jittering approach did not cause any disturbances in our tests.

In addition to the caching and reconstruction mechanism working directly on the cache, we developed a framework for filtering the reconstructed illumination in a layered way, based on the idea of path space filtering. With a limited recursion depth of n , we store n buffers with the original image resolution, each holding only an individual bounce/vertex of each pixel's path. The idea is to filter each bounce individually, while our framework does not enforce a specific filtering algorithm. Instead, the cross-bilateral filter and the auxiliary buffers it relies on are merely an exemplary implementation and it would be certainly interesting how other recent denoising approaches could be adapted to the layered framework and the specific noise appearance in our system.

Benchmarks showed the reconstruction from the HashCache to be clearly interactive, while the caching process may be computationally demanding at times. Therefore, we suggested to perform the caching process with an adaptive resolution depending on the required time and desired frame rate. Visual quality has been compared with a purely numeric method (relMSE) as well as a perceptual method (MS-SSIM); comparing it to reference renderings showed that our approach performs equal to one to two orders of magnitude more samples from a basic path tracer.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we have presented multiple approaches for improving the quality and performance of ray-based rendering methods in several different environments. After giving an introductory overview of the most important research in the fields of ray-based, perception-based and gaze-contingent rendering, denoising and caching methods, we described the methods and findings of our approach to *guided high-quality rendering*.

In the context of designing this user-centered method for steering the Monte Carlo (MC)-based rendering process in global illumination (GI) rendering on large display systems, we have developed a model that adapts the rendered detail based on a user-selectable region of interest (RoI) using a tracked input device. With this device, the RoI can be placed and scaled freely by aiming at the display system. This region is then rendered with a high sampling rate, while the periphery is scheduled with much lower sampling rates. To avoid extreme brightness variations in the image due to varying sampling rates, we presented exemplary implementations for appropriate filtering methods that approximate the ground truth brightness of the image early in the rendering process over the whole image plane.

In order to accelerate the rendering process for sparsely placed samples, we introduced a novel scheduling scheme based on sparse matrix compression, that led to significant speedups for small to medium-sized RoIs. With a small RoI corresponding to a field of view (FoV) of 10° , only 1.2% of the total pixels of the image had to be rendered, while a FoV of 60° required 36.3% of the pixels. We have shown that our approach accelerates the convergence rate inside the RoI significantly when compared to full rendering. The measured speedups ranged from 37.43 for a 10° FoV to 2.44 for a 60° FoV. This gives the user, who might be an artist modeling a scene and previewing it on the display system, the means to analyse specific parts of the scene much quicker than with full rendering.

Subsequently, we shifted our focus from large display systems to head-mounted displays (HMDs) in our work on foveated ray tracing for HMDs. We described the building blocks of our reprojection-based foveated rendering pipeline that relies on adapting the sampling rate using a piecewise linear falloff function and reusing image information from preceding frames as well as a low-resolution support image to fill in gaps from the sparse sampling process. In addition, we described the temporal accumulation and potential post-processing methods to improve image quality. Experimental results have shown significant speedups between 1.46 and 4.18 for the tested scenes. Users did not experience any significant difference between foveated rendering and full rendering for the chosen parameters, as shown in the user study.

Based on the latter, we took a closer look at the recorded eye tracking data to analyse the tracking precision and connections between fixation accuracy and quality ratings. We found tracking accuracy to be around 1° in the central region, while at larger eccentricities the accuracy was reduced (to about 3.5° at an eccentricity of 15°). With regard to the quality ratings, we found potential evidence of visual tunneling effects, as despite the reduced fixation accuracy, quality ratings were highest for the moving fixation target.

In the subsequent chapter, contrary to the preceding two chapters, we did not focus on a specific display system, but rather on a specific acceleration technique for GI rendering. Building upon the concept of linkless octrees Choi, Ju, et al. (2009), we developed a hash-based world space caching strategy for global illumination data acting as an interactive process. Using a hybrid reconstruction approach, the presented technique also allows for the visualization of glossy and specular materials. We showed how spatial jittering improves the quantization artifacts that were otherwise present in the direct visualization of the cached data.

Combined with an exemplary filtering technique implemented in our novel layered filtering framework, we demonstrated the viability of our approach for interactive scene exploration. By using multi-scale structural similarity (MS-SSIM) to compare image sequences rendered with various numbers of samples per pixel for both basic path tracing and the HashCache system, we showed that it provides an effective way to maintain the visual quality at a much faster rendering rate. The results showed an improvement by one to two orders of magnitude with regard to sample reduction for a similar image quality when compared to basic path tracing.

6.2 Contributions

In Chapter 2, we found that despite the large body of research in the field of ray-based rendering, bringing such methods to interactive systems is still an ongoing process. Looking at the long history of ray-based rendering, which has been around for more than 50 years, it becomes clear that there has been a lack of hardware support for a large portion of that time. Freely programmable graphics devices have only appeared in 2007 in conjunction with NVIDIA's CUDA, while specific ray tracing support in hardware has been available since 2018.

Therefore, our aim was to improve the applicability of ray-based methods in several rendering contexts: for preview rendering of GI of a static viewpoint on large display systems, for real-time rendering (without GI) in virtual reality (VR) environments with an HMD, and for preview rendering of full GI with the possibility of interactive scene exploration.

Knowing that the research in the field of ray-based rendering still develops rapidly, all our methods are mostly independent of the underlying rendering method that is employed. This means that the specifics of the sampling process for acquiring relevant paths as described in Chapters 3 and 5 remain untouched, so that the GI renderer itself could be replaced with ease. Similarly, our foveation and reprojection method described in Chapter 4 could be used with an arbitrary rendering system that allows for fully adaptive sampling of the image plane, or even full GI rendering in the future.

Based on appropriate methods identified in the work described in Chapter 2, we achieved the aims stated in the introduction.

First of all, we have developed a method for selectively focusing on specific image regions on a large display system using an appropriate interaction device. This enables the user to select parts of an image that are important in a process like design review or an iterative process that alternates between modeling and preview rendering.

Regarding interactive ray tracing on HMDs, we have developed a foveation method that utilizes the eye tracker in the HMD to adapt the sampling rate to the user's gaze. At the same time, the acquired speedup is sufficient for the whole system to provide the necessary rendering times for staying within the HMD's VSync limit.

Interactive scene exploration with GI rendering is made possible with our caching and reconstruction method, that also allows for using non-diffuse material descriptions, which was explicitly stated as a main goal for this system. At the same time, our novel layered filtering framework allows for further improving the appearance of

such materials.

A more detailed description of the individual contributions we made in each chapter is given below.

6.2.1 Guided High-Quality Rendering

We developed a general framework for adaptively rendering an image on a large display system in a user-centered context. This includes two interaction metaphors for placing and scaling the relevant image regions, the model for density adaptation itself, and a novel scheduling method based on sparse matrix compression (SMC), that turned out to yield a significant speedup in our system. This way, an image with varying sample density can be efficiently rendered on the GPU. In addition, we propose a basic filtering approach that improves the user’s visual context by reducing brightness variations resulting from sparse sampling of the image plane.

6.2.2 Foveated Ray Tracing in Head-Mounted Displays

With our foveated rendering system, we provide a high-performance, adaptive sampling approach for ray tracing driven by eye tracking and limitations of human perception. By using a coarse approximation of the scene geometry, the utilized reprojecting and merging process aids the reconstruction of the final image from sparse samples at a high performance. We conducted a user study showing that our method has only minimal impact on the perceived quality when considering foveal region limits. Also, the outcome of the study reveals a great potential of deploying visual attention to further optimize rendering techniques. An analysis of the recorded eye tracking data made it possible to give an estimate of the utilized eye tracker’s tracking precision, supported by the evaluation of eccentricity-based quality ratings. Furthermore, the achieved fixation accuracy was inspected and a connection could be established between the subjective perceived visual quality and the specifically measured fixation accuracy, providing possible evidence of the presence of visual tunneling effects and the magnitude of their influence on the user’s perception.

6.2.3 Hash-based Hierarchical Caching and Layered Filtering

We developed a hash-based approach for caching and reconstructing the global illumination of a scene, tailored specifically towards fast reconstruction and the implementation on GPUs. In addition to diffuse materials, our system also supports glossy and specular materials with a hybrid reconstruction method similar to final

gathering. We proposed a layered filtering technique, derived from the idea of path space filtering: multiple layers of illumination information are held per pixel. These are denoised separately, subsequently propagating them along the recorded paths' image representation. The main effect of this filtering approach is an improved quality of non-diffuse materials, as overblurring is avoided by providing additional details to the filtering method.

6.3 Future Research and Impact of Technological Developments

On the one hand, future research may include further development of some individual methods we presented. On the other hand, some of our techniques may also be combined with each other. We believe that several of the central aspects of the research presented in this thesis are useful contributions to the graphics community, which is supported by the publications made during the process.

6.3.1 Guided High-Quality Rendering

Possible future research regarding guided high-quality Rendering includes scheduling methods for increased ray coherence, which may result in further performance benefits. Sparsely sampled as well as focused areas could be processed with modern denoising methods discussed in Chapter 2. It has to be analysed whether denoising should be performed at the full image resolution or at area-specific fractions of the original resolution. Also, when performing reconstruction and filtering steps of this kind, performance and perceptual aspects have to be taken into account. Even though the basics of gaze-contingent rendering are well-understood, a user study should be performed on how the noise introduced by the stochastic nature of path tracing affects the perceived quality of rendered images using RoIs based on the user's gaze. Depending on their performance, reconstruction methods could also be implemented to be executed once every few seconds or iteratively for static viewpoints.

Generally, a next step to our work would be rendering at the full resolution of a large high-resolution display wall (LHRDW). As rendering at the according resolutions for such a display system is a large computational effort, rendering clusters with dozens of GPUs come into play. Hybrid scheduling approaches for adaptive resolutions are one of the major challenges when implementing such a distributed rendering system.

Technological developments that happened since this work has been published certainly have an influence on the results. The availability of hardware support simply

implies a higher performance of our system, but the benefit of our technique remains untouched. However, more recent denoising methods that already yield visually pleasant results at very low sampling rates may raise the question of our approach's utility nowadays. While image quality provided by such methods is great, our approach delivers unbiased results within the image regions defined as important. At the same time, applying a denoising filter in this region would introduce bias, so whether or not our approach can be seen as useful in the light of recent technological developments depends largely on the requirements regarding the generated visualization. It is also noteworthy that denoising methods that work at very low sampling rates often rely on temporal accumulation, while we mainly aimed for rendering a static image, not moving around interactively. In such a case, temporal accumulation comes down to the usual running estimate employed in path tracing.

6.3.2 Foveated Ray Tracing in Head-mounted Displays

Looking at both Chapters 3 and 4, including the SMC-based scheduling approach from guided high-quality Rendering may accelerate the foveated rendering system significantly. Depending on the resulting speedup, additional effects could be included and higher resolutions would become viable, reducing potential aliasing artifacts. However, it still has to be analysed how strong the effect of reprojection on the rendering of non-diffuse materials might be; adaptive reconstruction methods could be necessary in order to avoid visual artifacts, but at the same time, ray tracing gives us the advantage of being able to (re-)sample individual pixels efficiently. This could be used to include view-dependent effects in the resampling process, but it should be considered that this could lead to a fully sampled image, which would degrade performance too far to provide the required refresh rates. Still, improved frame rates would also increase the coherence between subsequent frames, which in turn also supports the quality delivered by the reprojection process and reduces the required amount of resampling.

Adaptive vertical synchronization techniques, nowadays ubiquitous in desktop displays, would greatly support immersion and the avoidance of motion sickness in HMDs. Generally, the foveal falloff as well as the minimum sampling probability could be altered dynamically, making our system adaptive to the complexity of the visual content.

Frameless rendering is an additional potential field for future research. It may be a viable approach to run the rendering and reprojection components of our foveated rendering system in separate threads, so they would generate and merge new samples and resolve reprojection errors in an asynchronous way.

Our findings regarding the correspondence of eye tracking data and quality ratings

in foveated rendering lead us to the assumption that there are circumstances which make it possible to reduce visual quality without the user noticing. This is the case in games, where events can be triggered that produce a change in the visuals, or task-driven environments, where task or navigation complexity may lead to high mental workloads. Moreover, certain events may allow for deriving a hint which part of the scene attracts attention. Thus, visual quality could be reduced even further without the user noticing. However, attentional models and gaze predictions are far from accurate (Weier, Stengel, et al. 2017). Nonetheless, more recently, the flicker observer effect and the higher temporal resolution for peripheral vision has successfully been used to direct the user’s gaze by Waldin, Waldner, et al. (2017).

Future research may also include the analysis of how optimal foveal region configurations (FRCs) can be determined and how the point of regard (PoR) can be optimally placed even with imprecise tracking. Also, it may be possible to exploit visual tunneling effects directly to improve performance or, alternatively, visual quality for central vision. In addition, it may be worth looking into the comparative behaviour of tracking devices with different update rates for analyses such as the one we have presented here.

At the time our system has been developed, explicit hardware support for ray tracing was unavailable, which is why we had to leverage two NVIDIA Quadro GPUs to meet the actual frame rate requirements of 75Hz. For the HMD used in our evaluation, with current graphics hardware, a single GPU will certainly suffice to meet these requirements, but while GPUs have developed quite a lot, so have HMDs. With high-end models providing a 4k resolution per eye and a refresh rate of 90Hz or even higher, ray-based rendering at sufficient frame rates is still challenging. Also, we have reasoned in the according chapter that the resolution of HMDs is required to increase a lot until it comes even close to what the human visual system (HVS) can actually perceive. This implies that our approach – and foveated rendering in general – is still an important tool today, and it is even likely to become more important in the future, since very high resolution displays may also suffer from the limited bandwidth of the display connection. Combining foveated rendering with hardware support by display controllers could in turn lead to the possibility of adaptively refreshing individual pixels of a display, consequently reducing the required bandwidth of such a connection.

6.3.3 Hash-based Hierarchical Caching and Layered Filtering

Utilizing the fast reconstruction step of our HashCache, improvements of guided rendering as well as foveated rendering are thinkable. However, in order to make the approach suitable for the high frame rates required for foveated rendering, the

caching and reconstruction processes have to be separated further in order to avoid the performance impact that occurs when alternating between the caching and reconstruction processes on a single GPU. Instead, employing a secondary GPU or even a centralized client-server architecture would allow for filling the cache parallel to the reconstruction process.

Another possibility of utilizing such a parallelized architecture would be to employ it for stereo rendering or multiple viewports in general, as in a multi-user system. While a central system would be responsible for updating and maintaining the cache, clients would only request cache entries from the server and perform the remaining reconstruction themselves. The resulting network traffic and latencies would have to be thoroughly analysed in order to judge the viability of such an approach.

To improve the general convergence rate of the presented methods that are based on GI rendering, other, more sophisticated rendering techniques could be employed, such as bidirectional path tracing (BDPT) (Lafortune 1993; Veach and Guibas 1995a) or Metropolis light transport (MLT) (Veach and Guibas 1997). Extending the HashCache with an additional component that supports adaptive importance sampling based on the currently available illumination information would be an enhancement to significantly boost the system’s performance. However, special attention needs to be directed to the adjustment of the individual samples’ contributions, as modified probability distributions would also have a “temporal” aspect: taking the same sample at two points in the progressive process would yield different contributions if not done correctly due to varying probability densities. With modern filtering methods that produce visually pleasant results even at 1 sample per pixel, an extension of our foveated ray tracing system towards global illumination could also be a viable effort.

Integrating more elaborate filtering techniques into our layered filtering framework and adapting them to the kind of noise present in the HashCache would be interesting especially for the reconstruction of glossy materials. We are certain that the HashCache has great potential, especially in conjunction with methods that rely upon temporal integration in image space (Schied, Kaplanyan, et al. 2017; Schied, Peters, et al. 2018), or other (machine-learning-based) techniques (Chaitanya, Kaplanyan, et al. 2017). However, the latter may require new training sets due to the more stable output produced by the HashCache and the noise appearing at different scales.

An adaptive resolution of the HashCache or an additional caustics cache may improve the possible support of caustics in the system. Performance improvements may be achieved by looking into different addressing schemes for the octree cells, following a more localized scheme like Morton order.

One option would be to provide a dynamic local caching mechanism that works at higher resolutions and adjusts to a scene's light distribution. Additionally, instead of storing each level of the octree in a separate hash map, we could store all levels in one hash map with occupation information. This approach avoids separate hash map queries per octree level, making it possible to compute the correct data array indices from the occupation information. However, memory requirements would increase.

Besides the general hardware support for ray-based rendering, there are two main developments we see as influential for a method like ours. First, the amount of available GPU memory is very important for a method that stores a large amount of data describing a scene's appearance in world space and improves with higher resolutions. Second, the bandwidth of mass storage is increasing with M.2 SSDs being widely available. Already with PCIe 3.0, such devices may achieve a theoretical maximum transfer rate of almost 4GiB/s, which is already doubled with PCIe 4.0. This opens new possibilities regarding the implementation of an out-of-core approach that actually uses more than only GPU memory for caching illumination, and instead also leverages system memory and mass storage, which would make it possible to cache illumination at a very high resolution.

Bibliography

- Adler, F. H., P. L. Kaufman, et al. (2011). *Adler's Physiology of the Eye*. 11th ed. Elsevier Health Sciences.
- Aila, Timo, Samuli Laine, et al. (2012). *Understanding the Efficiency of Ray Traversal on GPUs - Kepler and Fermi Addendum*. Technical Report NVR-2012-02. HPG2012 poster. NVIDIA.
- Amenta, Nina and Dan A. Alcantara (2011). *Efficient hash tables on the gpu*. Library Catalog: www.semanticscholar.org. URL: [%5Curl%7B/paper/Efficient-hash-tables-on-the-gpu-Amenta-Alcantara/fb56eada079ad85d9224a9b9fbf763b5ba921eb9%7D](https://doi.org/10.1145/1468075.1468082) (visited on 06/10/2020).
- Appel, Arthur (Apr. 1968). "Some techniques for shading machine renderings of solids". In: *Proceedings of the April 30–May 2, 1968, spring joint computer conference*. AFIPS '68 (Spring). Atlantic City, New Jersey: Association for Computing Machinery, pp. 37–45. ISBN: 978-1-4503-7897-0. DOI: 10.1145/1468075.1468082. URL: <https://doi.org/10.1145/1468075.1468082> (visited on 06/30/2020).
- Applied Sciences, Hochschule Bonn-Rhein-Sieg University of (2014). *Big Data on the Big Screen — Hochschule Bonn-Rhein-Sieg (H-BRS)*. URL: <https://www.h-brs.de/en/inf/news/big-data-big-screen> (visited on 04/15/2020).
- Bahill, A. T., D. Adler, et al. (June 1975). "Most naturally occurring human saccades have magnitudes of 15 degrees or less". In: *Investigative Ophthalmology* 14.6, pp. 468–469. ISSN: 0020-9988.
- Bako, Steve, Thijs Vogels, et al. (2017). "Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings". In: *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2017)* 36.4. URL: http://cvc.ucsb.edu/graphics/Papers/SIGGRAPH2017_KPCN/ (visited on 04/16/2018).
- Bala, Kavita, Julie Dorsey, et al. (July 1999). "Radiance interpolants for accelerated bounded-error ray tracing". In: *ACM Transactions on Graphics* 18.3, pp. 213–256. ISSN: 0730-0301. DOI: 10.1145/336414.336417. URL: <https://doi.org/10.1145/336414.336417> (visited on 08/19/2020).
- Bala, Kavita, Bruce Walter, et al. (July 2003). "Combining edges and points for interactive high-quality rendering". In: *ACM Transactions on Graphics (TOG)* 22.3, pp. 631–640. ISSN: 0730-0301. DOI: 10.1145/882262.882318. URL: <https://doi.org/10.1145/882262.882318> (visited on 02/10/2020).
- Baum, D. R., H. E. Rushmeier, et al. (July 1989). "Improving radiosity solutions through the use of analytically determined form-factors". In: *ACM SIGGRAPH Computer Graphics* 23.3, pp. 325–334. ISSN: 0097-8930. DOI: 10.1145/74334.74367. URL: <https://doi.org/10.1145/74334.74367> (visited on 07/20/2020).
- Bauszat, Pablo, Martin Eisemann, et al. (2011). "Guided Image Filtering for Interactive High-quality Global Illumination". In: *Computer Graphics Forum*. Vol. 30, pp. 1361–1368.

- Bekaert, Philippe, László Neumann, et al. (1998). “Hierarchical Monte Carlo Radiosity”. In: *Rendering Techniques '98*. Eurographics. Vienna: Springer, pp. 259–268. ISBN: 978-3-7091-6453-2. DOI: 10.1007/978-3-7091-6453-2_24.
- Binder, Nikolaus, Sascha Fricke, et al. (Aug. 2018). “Fast path space filtering by jittered spatial hashing”. In: *ACM SIGGRAPH 2018 Talks*. SIGGRAPH '18. Vancouver, British Columbia, Canada: Association for Computing Machinery, pp. 1–2. ISBN: 978-1-4503-5820-0. DOI: 10.1145/3214745.3214806. URL: <https://doi.org/10.1145/3214745.3214806> (visited on 06/10/2020).
- Bitterli, Benedikt and Wojciech Jarosz (Nov. 2019). “Selectively metropolised Monte Carlo light transport simulation”. In: *ACM Transactions on Graphics* 38.6, 153:1–153:10. ISSN: 0730-0301. DOI: 10.1145/3355089.3356578. URL: <https://doi.org/10.1145/3355089.3356578> (visited on 08/17/2020).
- Bitterli, Benedikt, Fabrice Rousselle, et al. (2016). “Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings”. In: *Computer Graphics Forum*. Vol. 35. Wiley Online Library, pp. 107–117. URL: <http://onlinelibrary.wiley.com/doi/10.1111/cgf.12954/full> (visited on 11/18/2016).
- Bosse, Sebastian, Dominique Maniry, et al. (Jan. 2018). “Deep Neural Networks for No-Reference and Full-Reference Image Quality Assessment”. In: *IEEE Transactions on Image Processing* 27.1. Conference Name: IEEE Transactions on Image Processing, pp. 206–219. ISSN: 1941-0042. DOI: 10.1109/TIP.2017.2760518.
- Buades, A., B. Coll, et al. (Jan. 2005). “A Review of Image Denoising Algorithms, with a New One”. In: *Multiscale Modeling & Simulation* 4.2, pp. 490–530. ISSN: 1540-3459. DOI: 10.1137/040616024. URL: <http://epubs.siam.org/doi/abs/10.1137/040616024> (visited on 11/16/2015).
- Buades, Antoni, Bartomeu Coll, et al. (Feb. 2008). “Nonlocal Image and Movie Denoising”. In: *International Journal of Computer Vision* 76.2, pp. 123–139. ISSN: 1573-1405. DOI: 10.1007/s11263-007-0052-1. URL: <https://doi.org/10.1007/s11263-007-0052-1> (visited on 08/20/2020).
- Chaitanya, Chakravarty R. Alla, Anton S. Kaplanyan, et al. (July 2017). “Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder”. In: *ACM Transactions on Graphics* 36.4, pp. 1–12. ISSN: 07300301. DOI: 10.1145/3072959.3073601. URL: <http://dl.acm.org/citation.cfm?doid=3072959.3073601> (visited on 08/01/2018).
- Chandler, Damon M. and Sheila S. Hemami (Sept. 2007). “VSNR: A Wavelet-Based Visual Signal-to-Noise Ratio for Natural Images”. In: *IEEE Transactions on Image Processing* 16.9. Conference Name: IEEE Transactions on Image Processing, pp. 2284–2298. ISSN: 1941-0042. DOI: 10.1109/TIP.2007.901820.
- Chen, Jiating, Bin Wang, et al. (2011). “Improved Stochastic Progressive Photon Mapping with Metropolis Sampling”. In: *Computer Graphics Forum* 30.4. ePrint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2011.01979.x>, pp. 1205–1213. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2011.01979.x. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.01979.x> (visited on 07/15/2020).
- Choi, Myung Geol, Eunjung Ju, et al. (2009). “Linkless Octree Using Multi-Level Perfect Hashing”. In: *Computer Graphics Forum*. Vol. 28. Wiley Online Library, pp. 1773–1780.
- Christensen, Per, Andrew Kensler, et al. (2018). “Progressive Multi-Jittered Sample Sequences”. In: *Computer Graphics Forum* 37.4. ePrint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13472>, pp. 21–33. DOI: 10.1111/cgf.

13472. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13472>.
- Chua, Hannah Faye, Julie E Boland, et al. (2005). “Cultural variation in eye movements during scene perception”. In: *Proceedings of the National Academy of Sciences* 102.35. Publisher: National Acad Sciences, pp. 12629–12633.
- Cohen, Michael F., Shenchang Eric Chen, et al. (June 1988). “A progressive refinement approach to fast radiosity image generation”. In: *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '88. New York, NY, USA: Association for Computing Machinery, pp. 75–84. ISBN: 978-0-89791-275-4. DOI: 10.1145/54852.378487. URL: <https://doi.org/10.1145/54852.378487> (visited on 07/20/2020).
- Cohen, Michael F. and Donald P. Greenberg (July 1985). “The hemi-cube: a radiosity solution for complex environments”. In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, pp. 31–40. ISBN: 978-0-89791-166-5. DOI: 10.1145/325334.325171. URL: <https://doi.org/10.1145/325334.325171> (visited on 07/17/2020).
- Coombe, Greg, Mark J. Harris, et al. (July 2005). “Radiosity on graphics hardware”. In: *ACM SIGGRAPH 2005 Courses*. SIGGRAPH '05. Los Angeles, California: Association for Computing Machinery, 179–es. ISBN: 978-1-4503-7833-8. DOI: 10.1145/1198555.1198782. URL: <https://doi.org/10.1145/1198555.1198782> (visited on 07/20/2020).
- Crassin, Cyril, Fabrice Neyret, et al. (2011). “Interactive Indirect Illumination Using Voxel Cone Tracing”. In: *Computer Graphics Forum* 30.7, pp. 1921–1930. ISSN: 01677055. DOI: 10.1111/j.1467-8659.2011.02063.x. URL: <http://doi.wiley.com/10.1111/j.1467-8659.2011.02063.x> (visited on 09/29/2014).
- Curcio, Christine A., Kenneth R. Sloan, et al. (1990). “Human photoreceptor topography”. In: *Journal of Comparative Neurology* 292.4. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cne.902920402>, pp. 497–523. ISSN: 1096-9861. DOI: 10.1002/cne.902920402. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cne.902920402> (visited on 08/19/2020).
- Dahm, Ken and Alexander Keller (July 2017). “Learning light transport the reinforced way”. In: *ACM SIGGRAPH 2017 Talks*. SIGGRAPH '17. New York, NY, USA: Association for Computing Machinery, pp. 1–2. ISBN: 978-1-4503-5008-2. DOI: 10.1145/3084363.3085032. URL: <https://doi.org/10.1145/3084363.3085032> (visited on 08/18/2020).
- Dammertz, Holger, Daniel Sewtz, et al. (2010). “Edge-avoiding À-Trous wavelet transform for fast global illumination filtering”. In: *Proceedings of the Conference on High Performance Graphics*, pp. 67–75.
- Debattista, K., P. Dubla, et al. (Dec. 2009). “Instant Caching for Interactive Global Illumination”. In: *Computer Graphics Forum* 28.8, pp. 2216–2228. ISSN: 01677055, 14678659. DOI: 10.1111/j.1467-8659.2009.01435.x. URL: <http://doi.wiley.com/10.1111/j.1467-8659.2009.01435.x> (visited on 08/01/2018).
- Defense, Department of (1999). *MIL-STD-1472 F, Design Criteria Standard, Human Engineering*. URL: http://everyspec.com/MIL-STD/MIL-STD-1400-1499/MIL-STD-1472F_208/ (visited on 05/13/2019).
- Delbracio, Mauricio, Pablo Musé, et al. (Feb. 2014). “Boosting Monte Carlo Rendering by Ray Histogram Fusion”. In: *ACM Trans. Graph.* 33.1, 8:1–8:15. ISSN:

- 0730-0301. DOI: 10.1145/2532708. URL: <http://doi.acm.org/10.1145/2532708> (visited on 06/23/2014).
- Dorr, Michael, Thomas Martinetz, et al. (Aug. 2010). “Variability of eye movements when viewing dynamic natural scenes”. In: *Journal of Vision* 10.10. Publisher: The Association for Research in Vision and Ophthalmology, pp. 28–28. ISSN: 1534-7362. DOI: 10.1167/10.10.28. URL: <https://jov.arvojournals.org/article.aspx?articleid=2121333> (visited on 08/20/2020).
- Duchowski, Andrew T. (2007). *Eye tracking methodology: theory and practice*. 2nd ed. London: Springer. ISBN: 978-1-84628-608-7.
- Duchowski, Andrew T. and Arzu Çöltekin (Dec. 2007). “Foveated Gaze-contingent Displays for Peripheral LOD Management, 3D Visualization, and Stereo Imaging”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 3.4, 6:1–6:18. ISSN: 1551-6857.
- Durand, Frédo, George Drettakis, et al. (Apr. 1999). “Fast and accurate hierarchical radiosity using global visibility”. In: *ACM Transactions on Graphics* 18.2, pp. 128–170. ISSN: 0730-0301. DOI: 10.1145/318009.318012. URL: <https://doi.org/10.1145/318009.318012> (visited on 07/20/2020).
- Eberly, David (Feb. 2019). “Distance from a Point to an Ellipse, an Ellipsoid, or a Hyperellipsoid”. Redmond WA 98052. URL: <https://www.geometrictools.com> (visited on 04/21/2020).
- Elek, Oskar, Manu Mathew Thomas, et al. (2019). “Learning Patterns in Sample Distributions for Monte Carlo Variance Reduction”. In: *CoRR* abs/1906.00124. URL: <https://arxiv.org/abs/1906.00124>.
- Field, A.P. and G. Hole (2003). *How to Design and Report Experiments*. Sage publications Limited. ISBN: 978-0-7619-7383-6.
- Frericks, Philipp, Thorsten Roth, et al. (2017). “A Framework for Cluster-based Rendering and Postprocessing”. In: *presented at the 10th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*.
- Fujita, Masahiro and Takahiro Harada (May 2014). *Foveated Real-Time Ray Tracing for Virtual Reality Headset*. URL: <http://research.lighttransport.com/foveated-real-time-ray-tracing-for-virtual-reality-headset>.
- Gastal, Eduardo S. L. and Manuel M. Oliveira (July 2012). “Adaptive manifolds for real-time high-dimensional filtering”. In: *ACM Transactions on Graphics* 31.4, 33:1–33:13. ISSN: 0730-0301. DOI: 10.1145/2185520.2185529. URL: <https://doi.org/10.1145/2185520.2185529> (visited on 06/10/2020).
- Gautron, Pascal, Jaroslav Krivánek, et al. (2005). *Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm*. Accepted: 2014-01-27T14:48:24Z ISSN: 1727-3463. The Eurographics Association. DOI: 10.2312/EGWR/EGSR05/055-064. URL: <https://diglib.eg.org:443/xmlui/handle/10.2312/EGWR.EGSR05.055-064> (visited on 08/20/2020).
- Geisler, Wilson S. and Jeffrey S. Perry (1998). “A real-time foveated multiresolution system for low-bandwidth video communication”. In: *in Proc. SPIE*, pp. 294–305.
- Georgiev, Iliyan and Marcos Fajardo (July 2016). “Blue-noise dithered sampling”. In: *ACM SIGGRAPH 2016 Talks*. SIGGRAPH ’16. New York, NY, USA: Association for Computing Machinery, p. 1. ISBN: 978-1-4503-4282-7. DOI: 10.1145/2897839.2927430. URL: <https://doi.org/10.1145/2897839.2927430> (visited on 08/17/2020).

- Georgiev, Iliyan, Jaroslav Krivánek, et al. (2012a). “Importance Caching for Complex Illumination”. In: *Computer Graphics Forum* 31.2pt3. ePrint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03049.x>, pp. 701–710. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2012.03049.x. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03049.x> (visited on 08/20/2020).
- Georgiev, Iliyan, Jaroslav Krivánek, et al. (2012b). “Light transport simulation with vertex connection and merging”. In: *ACM Transactions on Graphics (TOG)* 31.6, p. 192.
- Gibson, S. and R. J. Hubbard (1996). “Efficient Hierarchical Refinement and Clustering for Radiosity in Complex Environments”. In: Accepted: 2014-10-21T07:45:31Z Publisher: Blackwell Science Ltd and the Eurographics Association. ISSN: 1467-8659. URL: <https://diglib.eg.org:443/xmlui/handle/10.2312/7069> (visited on 07/20/2020).
- Gibson, Simon and Roger J. Hubbard (1997). “Perceptually-driven radiosity”. In: *Computer Graphics Forum*. Vol. 16. Issue: 2. Wiley Online Library, pp. 129–141.
- Goldstein, E. (Mar. 2013). *Sensation and Perception*. International ed of 9th revised ed. Belmont, CA: Wadsworth Publishing Co Inc. ISBN: 978-1-285-08514-2.
- Goral, Cindy M., Kenneth E. Torrance, et al. (Jan. 1984). “Modeling the interaction of light between diffuse surfaces”. In: *ACM SIGGRAPH Computer Graphics* 18.3, pp. 213–222. ISSN: 0097-8930. DOI: 10.1145/964965.808601. URL: <https://doi.org/10.1145/964965.808601> (visited on 07/17/2020).
- Gortler, Steven J., Peter Schröder, et al. (Sept. 1993). “Wavelet radiosity”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '93. Anaheim, CA: Association for Computing Machinery, pp. 221–230. ISBN: 978-0-89791-601-1. DOI: 10.1145/166117.166146. URL: <https://doi.org/10.1145/166117.166146> (visited on 07/20/2020).
- Greger, Gene, Peter Shirley, et al. (Mar. 1998). “The Irradiance Volume”. In: *IEEE Computer Graphics and Applications* 18.2, pp. 32–43. ISSN: 0272-1716. DOI: 10.1109/38.656788. URL: <https://doi.org/10.1109/38.656788> (visited on 08/19/2020).
- Gruson, Adrien, Binh-Son Hua, et al. (July 2018). “Gradient-domain volumetric photon density estimation”. In: *ACM Transactions on Graphics* 37.4, 82:1–82:13. ISSN: 0730-0301. DOI: 10.1145/3197517.3201363. URL: <https://doi.org/10.1145/3197517.3201363> (visited on 07/15/2020).
- Guenter, Brian, Mark Finch, et al. (Nov. 2012). “Foveated 3D Graphics”. In: *ACM Trans. Graph.* 31.6, 164:1–164:10. ISSN: 0730-0301. DOI: 10.1145/2366145.2366183. URL: <http://doi.acm.org/10.1145/2366145.2366183> (visited on 09/07/2015).
- Hachisuka, Toshiya, Wojciech Jarosz, et al. (2008). “Multidimensional adaptive sampling and reconstruction for ray tracing”. In: *ACM Transactions on Graphics (TOG)*. Vol. 27, p. 33.
- Hachisuka, Toshiya, Wojciech Jarosz, et al. (Aug. 2012). “State of the art in photon density estimation”. In: *ACM SIGGRAPH 2012 Courses*. SIGGRAPH '12. Los Angeles, California: Association for Computing Machinery, pp. 1–469. ISBN: 978-1-4503-1678-1. DOI: 10.1145/2343483.2343489. URL: <https://doi.org/10.1145/2343483.2343489> (visited on 07/15/2020).
- Hachisuka, Toshiya and Henrik Wann Jensen (Dec. 2009). “Stochastic progressive photon mapping”. In: *ACM Trans. Graph.* 28.5, 141:1–141:8. ISSN: 0730-0301.

- DOI: 10.1145/1618452.1618487. URL: <http://doi.acm.org/10.1145/1618452.1618487> (visited on 09/23/2013).
- Hachisuka, Toshiya and Henrik Wann Jensen (2010). “Parallel Progressive Photon Mapping on GPUs”. In: *ACM SIGGRAPH ASIA 2010 Sketches*. SA '10. New York, NY, USA: ACM, 54:1–54:1. ISBN: 978-1-4503-0523-5. DOI: 10.1145/1899950.1900004. URL: <http://doi.acm.org/10.1145/1899950.1900004>.
- Hachisuka, Toshiya, Shinji Ogaki, et al. (Dec. 2008). “Progressive photon mapping”. In: *ACM Transactions on Graphics* 27.5, p. 1. ISSN: 07300301. DOI: 10.1145/1409060.1409083. URL: <http://portal.acm.org/citation.cfm?doid=1409060.1409083> (visited on 08/01/2018).
- Hale, Kelly S. and Kay M. Stanney (2014). *Handbook of Virtual Environments: Design, Implementation, and Applications*. 2nd. Boca Raton, FL, USA: CRC Press, Inc. ISBN: 978-1-4665-1184-2.
- Hanrahan, Pat, David Salzman, et al. (July 1991). “A rapid hierarchical radiosity algorithm”. In: *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '91. New York, NY, USA: Association for Computing Machinery, pp. 197–206. ISBN: 978-0-89791-436-9. DOI: 10.1145/122718.122740. URL: <https://doi.org/10.1145/122718.122740> (visited on 07/20/2020).
- Havran, Vlastimil, Robert Herzog, et al. (2005). “Fast Final Gathering via Reverse Photon Mapping”. In: *Computer Graphics Forum* 24.3. ePrint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2005.00857.x>, pp. 323–332. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2005.00857.x. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2005.00857.x> (visited on 07/13/2020).
- He, Yong, Yan Gu, et al. (July 2014). “Extending the graphics pipeline with adaptive, multi-rate shading”. In: *ACM Transactions on Graphics* 33.4, 142:1–142:12. ISSN: 0730-0301. DOI: 10.1145/2601097.2601105. URL: <https://doi.org/10.1145/2601097.2601105> (visited on 08/20/2020).
- Heckbert, Paul S. (Sept. 1990). “Adaptive radiosity textures for bidirectional ray tracing”. In: *ACM SIGGRAPH Computer Graphics* 24.4, pp. 145–154. ISSN: 0097-8930. DOI: 10.1145/97880.97895. URL: <https://doi.org/10.1145/97880.97895> (visited on 07/20/2020).
- Hedman, Peter, Tero Karras, et al. (Feb. 2016). “Sequential Monte Carlo instant radiosity”. In: *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '16. Redmond, Washington: Association for Computing Machinery, pp. 121–128. ISBN: 978-1-4503-4043-4. DOI: 10.1145/2856400.2856406. URL: <https://doi.org/10.1145/2856400.2856406> (visited on 07/20/2020).
- Herholz, Sebastian, Oskar Elek, et al. (July 2016). “Product Importance Sampling for Light Transport Path Guiding”. In: *Computer Graphics Forum* 35.4, pp. 67–77. ISSN: 1467-8659. DOI: 10.1111/cgf.12950. URL: <http://onlinelibrary.wiley.com/doi/10.1111/cgf.12950/abstract> (visited on 09/01/2016).
- Hinkenjann, André, Oliver Jato, et al. (Mar. 2015). *High Quality Rendering and Visualization at the Institute of Visual Computing, Sankt Augustin, Germany*. Published: IEEE Virtual Reality (IEEE VR).
- Hirvonen, Antti, Atte Seppälä, et al. (2019). “Accurate Real-Time Specular Reflections with Radiance Caching”. In: *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Berkeley, CA: Apress, pp. 571–

607. ISBN: 978-1-4842-4427-2. DOI: 10.1007/978-1-4842-4427-2_32. URL: https://doi.org/10.1007/978-1-4842-4427-2_32 (visited on 02/10/2020).
- Hua, Binh-Son, Adrien Gruson, et al. (May 2017). “Gradient-Domain Photon Density Estimation”. In: *Computer Graphics Forum* 36.2, pp. 31–38. ISSN: 0167-7055.
- Hubel, David H. (1988). *Eye, brain, and vision*. Open Library ID: OL2393916M. New York: Scientific American Library. ISBN: 978-0-7167-5020-8.
- Hunt, Warren (Aug. 2015). *Virtual Reality: The Next Great Graphics Revolution*. Published: Keynote Talk HPG. URL: <http://www.highperformancegraphics.org/wp-content/uploads/2015/Keynote1/WarrenHuntHPGKeynote.pptx>.
- Huynh-Thu, Q. and M. Ghanbari (June 2008). “Scope of validity of PSNR in image/video quality assessment”. In: *Electronics Letters* 44.13. Conference Name: Electronics Letters, pp. 800–801. ISSN: 0013-5194. DOI: 10.1049/e1:20080522.
- Immel, David S., Michael F. Cohen, et al. (Aug. 1986). “A radiosity method for non-diffuse environments”. In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’86. New York, NY, USA: Association for Computing Machinery, pp. 133–142. ISBN: 978-0-89791-196-2. DOI: 10.1145/15922.15901. URL: <https://doi.org/10.1145/15922.15901> (visited on 07/17/2020).
- Jakob, Wenzel and Steve Marschner (2012). “Manifold Exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport”. In: *ACM Transactions on Graphics (TOG)* 31.4, p. 58.
- Jato, Oliver, Martin Weier, et al. (Sept. 2016). “OLIVE: Simulation within Human-Centric Lighting Environments”. In: *Proceedings of the 13. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR*.
- Jensen, Henrik Wann (1995). “Importance driven path tracing using the photon map”. In: *Rendering Techniques ’95*, pp. 326–335.
- (1996). “Global illumination using photon maps”. In: *Rendering Techniques’ 96*. Springer, pp. 21–30. ISBN: 3-211-82883-4.
- Jimenez, Jorge, Jose I. Echevarria, et al. (2012). “SMAA: Enhanced subpixel morphological antialiasing”. In: *Computer Graphics Forum*. Vol. 31. Wiley Online Library, pp. 355–364.
- Jin, Bongjun, Insung Ihm, et al. (Aug. 2009). “Selective and adaptive supersampling for real-time ray tracing”. In: *Proceedings of the Conference on High Performance Graphics 2009*. HPG ’09. New York, NY, USA: Association for Computing Machinery, pp. 117–125. ISBN: 978-1-60558-603-8. DOI: 10.1145/1572769.1572788. URL: <https://doi.org/10.1145/1572769.1572788> (visited on 08/16/2020).
- Kahn, Herman (1955). *Use of Different Monte Carlo Sampling Techniques*: Product Page. Library Catalog: www.rand.org Publisher: RAND Corporation. URL: <https://www.rand.org/pubs/papers/P766.html> (visited on 07/27/2020).
- Kajiya, James T. (1986). “The rendering equation”. In: *ACM SIGGRAPH Computer Graphics*. Vol. 20, pp. 143–150.
- Kang, Chun-meng, Lu Wang, et al. (Mar. 2016). “A survey of photon mapping state-of-the-art research and future challenges”. In: *Frontiers of Information Technology & Electronic Engineering* 17.3, pp. 185–199. ISSN: 2095-9230. DOI: 10.1631/FITEE.1500251. URL: <https://doi.org/10.1631/FITEE.1500251> (visited on 07/15/2020).

- Kaplanyan, Anton S. and Carsten Dachsbacher (Apr. 2013). “Adaptive progressive photon mapping”. In: *ACM Transactions on Graphics* 32.2, 16:1–16:13. ISSN: 0730-0301. DOI: 10.1145/2451236.2451242. URL: <https://doi.org/10.1145/2451236.2451242> (visited on 07/15/2020).
- Keller, Alexander (1996). “Quasi-Monte Carlo Radiosity”. In: *Rendering Techniques '96*. Eurographics. Vienna: Springer, pp. 101–110. ISBN: 978-3-7091-7484-5. DOI: 10.1007/978-3-7091-7484-5_11.
- (Aug. 1997). “Instant radiosity”. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., pp. 49–56. ISBN: 978-0-89791-896-1. DOI: 10.1145/258734.258769. URL: <https://doi.org/10.1145/258734.258769> (visited on 07/20/2020).
- Keller, Alexander, Ken Dahm, et al. (2014). “Path Space Filtering”. In: *ACM SIGGRAPH 2014 Talks*. SIGGRAPH '14. New York, NY, USA: ACM, 68:1–68:1. ISBN: 978-1-4503-2960-6. DOI: 10.1145/2614106.2614149. URL: <http://doi.acm.org/10.1145/2614106.2614149> (visited on 11/16/2015).
- Kettunen, Markus, Marco Manzi, et al. (2015). “Gradient-Domain Path Tracing”. In: *ACM Trans. Graph., to appear*.
- Khademi Kalantari, Nima, Steve Bako, et al. (2015). “A Machine Learning Approach for Filtering Monte Carlo Noise”. In: *Proceedings of SIGGRAPH 2015*. ACM Transactions on Graphics 4.34.
- Kim, Hyuk (2019). “Caustics Using Screen-Space Photon Mapping”. In: *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Berkeley, CA: Apress, pp. 543–555. ISBN: 978-1-4842-4427-2. DOI: 10.1007/978-1-4842-4427-2_30. URL: https://doi.org/10.1007/978-1-4842-4427-2_30 (visited on 07/15/2020).
- Knaus, Claude and Matthias Zwicker (May 2011). “Progressive photon mapping: A probabilistic approach”. In: *ACM Transactions on Graphics* 30.3, 25:1–25:13. ISSN: 0730-0301. DOI: 10.1145/1966394.1966404. URL: <https://doi.org/10.1145/1966394.1966404> (visited on 07/15/2020).
- Koskela, Matias, Kalle Immonen, et al. (2017). “Foveated Instant Preview for Progressive Rendering”. In: *SIGGRAPH Asia 2017 Technical Briefs*. SA '17. New York, NY, USA: ACM, 10:1–10:4. ISBN: 978-1-4503-5406-6. DOI: 10.1145/3145749.3149423. URL: <http://doi.acm.org/10.1145/3145749.3149423> (visited on 03/02/2018).
- Koskela, Matias, Kalle Immonen, et al. (June 2019). “Blockwise Multi-Order Feature Regression for Real-Time Path-Tracing Reconstruction”. In: *ACM Transactions on Graphics* 38.5, 138:1–138:14. ISSN: 0730-0301. DOI: 10.1145/3269978. URL: <https://doi.org/10.1145/3269978> (visited on 08/20/2020).
- Koskela, Matias, Atro Lotvonen, et al. (2019). *Foveated Real-Time Path Tracing in Visual-Polar Space*. Accepted: 2019-07-14T19:22:28Z ISSN: 1727-3463. The Eurographics Association. ISBN: 978-3-03868-095-6. DOI: 10.2312/sr20191219. URL: <https://diglib.eg.org:443/xmlui/handle/10.2312/sr20191219> (visited on 08/20/2020).
- Kowler, Eileen (July 2011). “Eye movements: the past 25 years”. In: *Vision Research* 51.13, pp. 1457–1483. ISSN: 1878-5646. DOI: 10.1016/j.visres.2010.12.014.
- Krivanek, J., P. Gautron, et al. (Sept. 2005). “Radiance caching for efficient global illumination computation”. In: *IEEE Transactions on Visualization and Com-*

- puter Graphics* 11.5, pp. 550–561. ISSN: 1077-2626. DOI: 10.1109/TVCG.2005.83.
- Křivánek, Jaroslav, Kadi Bouatouch, et al. (2006). “Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping”. In: *Proceedings of the 17th Eurographics Conference on Rendering Techniques*. EGSR '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, pp. 127–138. ISBN: 3-905673-35-5. DOI: 10.2312/EGWR/EGSR06/127-138. URL: <http://dx.doi.org/10.2312/EGWR/EGSR06/127-138> (visited on 11/06/2015).
- Krivaneck, Jaroslav and Pascal Gautron (Jan. 2009). “Practical Global Illumination with Irradiance Caching”. In: *Synthesis Lectures on Computer Graphics and Animation* 4.1, pp. 1–148. ISSN: 1933-8996, 1933-9003. DOI: 10.2200/S00180ED1V01Y200903CGR010. URL: <http://www.morganclaypool.com/doi/abs/10.2200/S00180ED1V01Y200903CGR010> (visited on 08/20/2020).
- Krull, Alexander, Tim-Oliver Buchholz, et al. (Apr. 2019). “Noise2Void - Learning Denoising from Single Noisy Images”. In: *arXiv*. arXiv: 1811.10980. URL: <http://arxiv.org/abs/1811.10980> (visited on 08/20/2020).
- Lafortune, Eric P. (1993). “Bi-Directional Path Tracing”. In: *Proceedings of Computergraphics '93*, pp. 145–153.
- Lafortune, Eric P. and Yves D. Willems (1995). “A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing”. In: *Rendering Techniques '95*, pp. 11–20. URL: <http://www.graphics.cornell.edu/%20eric/Dublin.html>.
- Laine, Samuli and Tero Karras (2011). “Efficient Sparse Voxel Octrees”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.8, pp. 1048–1059. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.240.
- Laine, Samuli, Hannu Saransaari, et al. (June 2007). “Incremental instant radiosity for real-time indirect illumination”. In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*. EGSR'07. Grenoble, France: Eurographics Association, pp. 277–286. ISBN: 978-3-905673-52-4. (Visited on 07/20/2020).
- Ledda, Patrick, Luis Paulo Santos, et al. (Nov. 2004). “A local model of eye adaptation for high dynamic range images”. In: *Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. AFRIGRAPH '04. New York, NY, USA: Association for Computing Machinery, pp. 151–160. ISBN: 978-1-58113-863-4. DOI: 10.1145/1029949.1029978. URL: <https://doi.org/10.1145/1029949.1029978> (visited on 08/25/2020).
- Lefebvre, Sylvain and Hugues Hoppe (2006). “Perfect spatial hashing”. In: *ACM Transactions on Graphics (TOG)*. Vol. 25. ACM, pp. 579–588. URL: <http://dl.acm.org/citation.cfm?id=1141926> (visited on 03/24/2015).
- Legge, Gordon E. and Daniel Kersten (Aug. 1987). “Contrast discrimination in peripheral vision”. In: *J. Opt. Soc. Am. A* 4.8, pp. 1594–1598.
- Lehtinen, Jaakko, Timo Aila, et al. (2012). “Reconstructing the indirect light field for global illumination”. In: *ACM Transactions on Graphics (TOG)* 31.4, p. 51.
- Lehtinen, Jaakko, Tero Karras, et al. (July 2013). “Gradient-domain Metropolis Light Transport”. In: *ACM Trans. Graph.* 32.4, 95:1–95:12. ISSN: 0730-0301. DOI: 10.1145/2461912.2461943. URL: <http://doi.acm.org/10.1145/2461912.2461943> (visited on 03/28/2014).
- Lehtinen, Jaakko, Jacob Munkberg, et al. (Oct. 2018). “Noise2Noise: Learning Image Restoration without Clean Data”. In: *arXiv*. arXiv: 1803.04189. URL: <http://arxiv.org/abs/1803.04189> (visited on 08/20/2020).

- Levoy, Marc and Ross Whitaker (1990). “Gaze-directed Volume Rendering”. In: *Proceedings of the 1990 Symposium on Interactive 3D Graphics*. I3D '90. New York, NY, USA: ACM, pp. 217–223. ISBN: 978-0-89791-351-5. DOI: 10.1145/91385.91449. URL: <http://doi.acm.org/10.1145/91385.91449> (visited on 02/01/2019).
- Li, Tzu-Mao, Yu-Ting Wu, et al. (Nov. 2012). “SURE-based Optimization for Adaptive Sampling and Reconstruction”. In: *ACM Trans. Graph.* 31.6, 194:1–194:9. ISSN: 0730-0301. DOI: 10.1145/2366145.2366213. URL: <http://doi.acm.org/10.1145/2366145.2366213> (visited on 05/27/2015).
- Lischinski, Dani, Filippo Tampieri, et al. (Nov. 1992). “Discontinuity Meshing for Accurate Radiosity”. In: *IEEE Computer Graphics and Applications* 12.6, pp. 25–39. ISSN: 0272-1716. DOI: 10.1109/38.163622. URL: <https://doi.org/10.1109/38.163622> (visited on 07/20/2020).
- Liu, Tao, Jin Gao, et al. (2020). “An Approach to Global Illumination Calculation Based on Hybrid Cone Tracing”. In: *IEEE Access* 8. Conference Name: IEEE Access, pp. 92061–92071. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2994597.
- Loschky, Lester C. and Gary S. Wolverson (Dec. 2007). “How Late Can You Update Gaze-contingent Multiresolutional Displays Without Detection?” In: *ACM Trans. Multimedia Comput. Commun. Appl.* 3.4, 7:1–7:10. ISSN: 1551-6857. DOI: 10.1145/1314303.1314310. URL: <http://doi.acm.org/10.1145/1314303.1314310> (visited on 05/13/2019).
- Lou, Chin Ian, Daria Migotina, et al. (2012). “Object Recognition Test in Peripheral Vision: A Study on the Influence of Object Color, Pattern and Shape”. In: *2012 International Conference on Brain Informatics*. Macau, China: Springer-Verlag, pp. 18–26. ISBN: 978-3-642-35138-9.
- Ma, Vincent C. H. and Michael D. McCool (Sept. 2002). “Low latency photon mapping using block hashing”. In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. HWWS '02. Saarbrücken, Germany: Eurographics Association, pp. 89–99. ISBN: 978-1-58113-580-0. (Visited on 07/13/2020).
- Mantiuk, Rafat, Kil Joong Kim, et al. (2011). “HDR-VDP-2: A Calibrated Visual Metric for Visibility and Quality Predictions in All Luminance Conditions”. In: *ACM SIGGRAPH 2011 Papers*. SIGGRAPH '11. event-place: Vancouver, British Columbia, Canada. New York, NY, USA: ACM, 40:1–40:14. ISBN: 978-1-4503-0943-1. DOI: 10.1145/1964921.1964935. URL: <http://doi.acm.org/10.1145/1964921.1964935>.
- Manzi, Marco, Markus Kettunen, et al. (Nov. 2016). “Temporal gradient-domain path tracing”. In: *ACM Transactions on Graphics* 35.6, 246:1–246:9. ISSN: 0730-0301. DOI: 10.1145/2980179.2980256. URL: <https://doi.org/10.1145/2980179.2980256> (visited on 08/17/2020).
- Mara, Michael, David Luebke, et al. (Mar. 2013). “Toward practical real-time photon mapping: efficient GPU density estimation”. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '13. Orlando, Florida: Association for Computing Machinery, pp. 71–78. ISBN: 978-1-4503-1956-0. DOI: 10.1145/2448196.2448207. URL: <https://doi.org/10.1145/2448196.2448207> (visited on 07/15/2020).
- Mark, William R., Leonard McMillan, et al. (1997). “Post-rendering 3D Warping”. In: *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. Providence, Rhode Island, USA: ACM, 7–ff. ISBN: 0-89791-884-3.

- McKee, S. and K. Nakayama (Jan. 1984). “The detection of motion in the peripheral visual field”. In: *Vision research* 24.1, pp. 25–32. ISSN: 0042-6989.
- Meng, Xiaoxu, Ruofei Du, et al. (July 2018). “Kernel Foveated Rendering”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1, 5:1–5:20. DOI: 10.1145/3203199. URL: <https://doi.org/10.1145/3203199> (visited on 08/20/2020).
- Metropolis, Nicholas, Arianna W. Rosenbluth, et al. (1953). “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21, p. 1087.
- Mir, Sermet (2020). “A SURVEY OF MONTE CARLO DENOISING: CHALLENGES AND POSSIBLE SOLUTIONS”. In: *Journal of Modern Technology and Engineering* 5.1, pp. 85–96.
- Miura, T. (1986). “Coping with situational demands: A study of eye movements and peripheral vision performance”. In: *Vision in Vehicles*, pp. 206–216.
- Müller, Thomas, M. Gross, et al. (2017). “Practical Path Guiding for Efficient Light-Transport Simulation”. In: *Comput. Graph. Forum*. DOI: 10.1111/cgf.13227.
- Müller, Thomas, Brian McWilliams, et al. (Oct. 2019). “Neural Importance Sampling”. In: *ACM Transactions on Graphics* 38.5, 145:1–145:19. ISSN: 0730-0301. DOI: 10.1145/3341156. URL: <https://doi.org/10.1145/3341156> (visited on 08/18/2020).
- Murphy, Hunter and Andrew T. Duchowski (2001). “Gaze-Contingent Level Of Detail Rendering”. In: — (2007). “Hybrid Image-/Model-based Gaze-contingent Rendering”. In: *4th Symposium on Applied Perception in Graphics and Visualization*. Tubingen, Germany: ACM, pp. 107–114. ISBN: 978-1-59593-670-7.
- Nathan, Reed (2015). *NVIDIA GameWorks VR*. Technical Report. SIGGRAPH 2015.
- Nehab, Diego, Pedro V. Sander, et al. (2007). “Accelerating Real-time Shading with Reverse Reprojection Caching”. In: *22Nd ACM Symposium on Graphics Hardware*. San Diego, California: Eurographics Association, pp. 25–35. ISBN: 978-1-59593-625-7.
- Neumann, Laszlo (1995). “Monte carlo radiosity”. In: *Computing* 55.1. Publisher: Springer, pp. 23–42.
- Nichols, Greg, Jeremy Shopf, et al. (June 2009). “Hierarchical image-space radiosity for interactive global illumination”. In: *Proceedings of the Twentieth Eurographics conference on Rendering*. EGSR’09. Girona, Spain: Eurographics Association, pp. 1141–1149. DOI: 10.1111/j.1467-8659.2009.01491.x. URL: <https://doi.org/10.1111/j.1467-8659.2009.01491.x> (visited on 07/20/2020).
- Noorlander, C., J. J. Koenderink, et al. (1983). “Sensitivity to spatiotemporal colour contrast in the peripheral visual field”. In: *Vision Research* 23.1, pp. 1–11. ISSN: 0042-6989. DOI: 10.1016/0042-6989(83)90035-4.
- Ohshima, T., H. Yamamoto, et al. (Mar. 1996). “Gaze-directed adaptive rendering for interacting with virtual space”. In: *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pp. 103–110. DOI: 10.1109/VRAIS.1996.490517.
- Omidvar, Mahmoud, Mickaël Ribardière, et al. (Oct. 2015). “A radiance cache method for highly glossy surfaces”. In: *The Visual Computer*, pp. 1–12. ISSN: 0178-2789, 1432-2315. DOI: 10.1007/s00371-015-1159-y. URL: <http://>

- link.springer.com/article/10.1007/s00371-015-1159-y (visited on 10/28/2015).
- Owen, Art B. (Jan. 1998). “Latin supercube sampling for very high-dimensional simulations”. In: *ACM Transactions on Modeling and Computer Simulation* 8.1, pp. 71–102. ISSN: 1049-3301. DOI: 10.1145/272991.273010. URL: <https://doi.org/10.1145/272991.273010> (visited on 07/27/2020).
- (2003). “Quasi-Monte Carlo Sampling”. In: *ACM SIGGRAPH 2003 Courses*. SIGGRAPH ’03. Los Angeles, California: Association for Computing Machinery.
- Papaioannou, Georgios (2011). “Real-time Diffuse Global Illumination Using Radiance Hints”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. HPG ’11. New York, NY, USA: ACM, pp. 15–24. ISBN: 978-1-4503-0896-0. DOI: 10.1145/2018323.2018326. URL: <http://doi.acm.org/10.1145/2018323.2018326> (visited on 11/16/2015).
- Patney, Anjul, Joohwan Kim, et al. (2016). “Perceptually-based Foveated Virtual Reality”. In: *ACM SIGGRAPH 2016 Emerging Technologies*. SIGGRAPH ’16. New York, NY, USA: ACM, 17:1–17:2. ISBN: 978-1-4503-4372-5. DOI: 10.1145/2929464.2929472. URL: <http://doi.acm.org/10.1145/2929464.2929472> (visited on 09/01/2016).
- Patney, Anjul, Marco Salvi, et al. (Nov. 2016). “Towards foveated rendering for gaze-tracked virtual reality”. In: *ACM Transactions on Graphics* 35.6, pp. 1–12. ISSN: 07300301. DOI: 10.1145/2980179.2980246. URL: <http://dl.acm.org/citation.cfm?doid=2980179.2980246> (visited on 08/01/2018).
- Pérard-Gayot, Arsène, Javor Kalojanov, et al. (2017). “GPU Ray Tracing using Irregular Grids”. In: *Computer Graphics Forum* 36.2, pp. 477–486. ISSN: 0167-7055. DOI: 10.1111/cgf.13142. URL: <https://doi.org/10.1111/cgf.13142> (visited on 06/10/2020).
- Peter, Ingmar and Georg Pietrek (1998). “Importance Driven Construction of Photon Maps”. In: *Rendering Techniques ’98*. Eurographics. Vienna: Springer, pp. 269–280. ISBN: 978-3-7091-6453-2. DOI: 10.1007/978-3-7091-6453-2_25.
- Pharr, Matt and Greg Humphreys (2010). *Physically based rendering: From theory to implementation*. Morgan Kaufmann. ISBN: 0-12-375079-2.
- Pohl, Daniel, Xucong Zhang, et al. (Mar. 2016). “Combining eye tracking with optimizations for lens astigmatism in modern wide-angle HMDs”. In: *2016 IEEE Virtual Reality (VR)*. ISSN: 2375-5334, pp. 269–270. DOI: 10.1109/VR.2016.7504757.
- Rawat, Waseem and Zenghui Wang (Sept. 2017). “Deep convolutional neural networks for image classification: A comprehensive review”. In: *Neural Computation* 29.9, pp. 2352–2449. ISSN: 0899-7667. DOI: 10.1162/neco_a_00990. URL: https://doi.org/10.1162/neco_a_00990 (visited on 08/19/2020).
- Roth, Scott D (Feb. 1982). “Ray casting for modeling solids”. In: *Computer Graphics and Image Processing* 18.2, pp. 109–144. ISSN: 0146-664X. DOI: 10.1016/0146-664X(82)90169-1. URL: <http://www.sciencedirect.com/science/article/pii/0146664X82901691> (visited on 06/30/2020).
- Roth, Thorsten, Martin Weier, et al. (2015). “Guided High-Quality Rendering”. In: *11th International Symposium on Visual Computing (ISVC)*.
- Roth, Thorsten, Martin Weier, et al. (2016). “An Analysis of Eye-Tracking Data in Foveated Ray Tracing”. In: *Proceedings of the 2016 Workshop on Eye Tracking and Visualization*. ETVIS.

- Roth, Thorsten, Martin Weier, et al. (2017). “A Quality-Centered Analysis of Eye Tracking Data in Foveated Rendering”. In: *Journal of Eye Movement Research* 10.5. ISSN: 1995-8692. URL: <https://bop.unibe.ch/index.php/JEMR/article/view/3729>.
- Roth, Thorsten, Martin Weier, et al. (2019). *Hash-based Hierarchical Caching for Interactive Previews in Global Illumination Rendering*. Tam, Roberts (Eds.): Computer Graphics and Visual Computing (CGVC 2019). The Eurographics Association. ISBN: 978-3-03868-096-3. DOI: 10.2312/cgvc.20191261.
- (2020). “Hash-Based Hierarchical Caching and Layered Filtering for Interactive Previews in Global Illumination Rendering”. In: *Computers* 9.1. ISSN: 2073-431X. DOI: 10.3390/computers9010017. URL: <https://www.mdpi.com/2073-431X/9/1/17>.
- Roughton, Thomas (2019). “Interactive Generation of Path-Traced Lightmaps”. PhD Thesis. Victoria University of Wellington.
- Rousselle, Fabrice, Claude Knaus, et al. (Nov. 2012). “Adaptive rendering with non-local means filtering”. In: *ACM Transactions on Graphics* 31.6, p. 1. ISSN: 07300301. DOI: 10.1145/2366145.2366214. URL: <http://dl.acm.org/citation.cfm?doid=2366145.2366214> (visited on 08/01/2018).
- Rousselle, Fabrice, Marco Manzi, et al. (Oct. 2013). “Robust Denoising using Feature and Color Information”. In: *Computer Graphics Forum* 32.7, pp. 121–130. ISSN: 1467-8659. DOI: 10.1111/cgf.12219. URL: <http://onlinelibrary.wiley.com/doi/10.1111/cgf.12219/abstract> (visited on 08/18/2015).
- Sanzenbacher, Paul, Lars Mescheder, et al. (2020). *Learning Neural Light Transport*. arXiv: 2006.03427 [cs.CV].
- Saunders, Daniel R. and Russell L. Woods (June 2014). “Direct measurement of the system latency of gaze-contingent displays”. In: *Behavior Research Methods* 46.2, pp. 439–447. ISSN: 1554-3528. DOI: 10.3758/s13428-013-0375-5. URL: <https://doi.org/10.3758/s13428-013-0375-5> (visited on 05/13/2019).
- Scherzer, Daniel, Chuong H. Nguyen, et al. (2012). “Pre-convolved Radiance Caching”. In: *Computer Graphics Forum* 31.4, pp. 1391–1397. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2012.03134.x. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03134.x> (visited on 02/10/2020).
- Schied, Christoph, Anton Kaplanyan, et al. (July 2017). “Spatiotemporal Variance-Guided Filtering: Real-time Reconstruction for Path Traced Global Illumination”. In: *Proceedings of High Performance Graphics*. Los Angeles, California, USA: ACM. ISBN: 978-1-4503-5101-0. DOI: 10.1145/3105762.3105774.
- Schied, Christoph, Christoph Peters, et al. (Aug. 2018). “Gradient Estimation for Real-time Adaptive Temporal Filtering”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.2, 24:1–24:16. DOI: 10.1145/3233301. URL: <https://doi.org/10.1145/3233301> (visited on 04/21/2020).
- Segovia, B., J. C. Iehl, et al. (June 2006). “Bidirectional instant radiosity”. In: *Proceedings of the 17th Eurographics conference on Rendering Techniques*. EGSR ’06. Nicosia, Cyprus: Eurographics Association, pp. 389–397. ISBN: 978-3-905673-35-7. (Visited on 07/20/2020).
- Sen, Pradeep and Soheil Darabi (2012). “On filtering the noise from the random parameters in Monte Carlo rendering”. In: *ACM Trans. Graph.* 31.3, p. 18.
- Sigitov, Anton, Thorsten Roth, et al. (Mar. 2015). “Enabling Global Illumination Rendering on Large, High-Resolution Displays”. In: *8th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*.

- Silvennoinen, Ari and Jaakko Lehtinen (Nov. 2017). “Real-time global illumination by precomputed local reconstruction from sparse radiance probes”. In: *ACM Transactions on Graphics* 36.6, pp. 1–13. ISSN: 07300301. DOI: 10.1145/3130800.3130852. URL: <http://dl.acm.org/citation.cfm?doid=3130800.3130852> (visited on 04/16/2018).
- Simmons, Maryann and Carlo H. Séquin (2000). “Tapestry: A Dynamic Mesh-based Display Representation for Interactive Rendering”. In: *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*. Springer-Verlag, pp. 329–340. ISBN: 3-211-83535-0.
- Sloan, Peter-Pike, Jan Kautz, et al. (2002). “Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments”. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. New York, NY, USA: ACM, pp. 527–536. ISBN: 1-58113-521-1. DOI: 10.1145/566570.566612. URL: <http://doi.acm.org/10.1145/566570.566612> (visited on 09/29/2014).
- Smits, Brian, James Arvo, et al. (July 1994). “A clustering algorithm for radiosity in complex environments”. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. SIGGRAPH '94. New York, NY, USA: Association for Computing Machinery, pp. 435–442. ISBN: 978-0-89791-667-7. DOI: 10.1145/192161.192277. URL: <https://doi.org/10.1145/192161.192277> (visited on 07/20/2020).
- Spencer, B. and M. W. Jones (Jan. 2009). “Hierarchical Photon Mapping”. In: *IEEE Transactions on Visualization and Computer Graphics* 15.1, pp. 49–61. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.67.
- Stamminger, Marc, Annette Scheel, et al. (2000). “Efficient Glossy Global Illumination with Interactive Viewing”. In: *Computer Graphics Forum* 19.1. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00385>, pp. 13–25. ISSN: 1467-8659. DOI: 10.1111/1467-8659.00385. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00385> (visited on 08/19/2020).
- Steinhurst, Joshua, Greg Coombe, et al. (May 2005). “Reordering for cache conscious photon mapping”. In: *Proceedings of Graphics Interface 2005*. GI '05. Victoria, British Columbia: Canadian Human-Computer Communications Society, pp. 97–104. ISBN: 978-1-56881-265-6. (Visited on 07/13/2020).
- Stengel, Michael, Steve Grogorick, et al. (Oct. 2015). “An Affordable Solution for Binocular Eye Tracking and Calibration in Head-mounted Displays”. In: *Proceedings of the 23rd ACM international conference on Multimedia*. MM '15. New York, NY, USA: Association for Computing Machinery, pp. 15–24. ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2806265. URL: <https://doi.org/10.1145/2733373.2806265> (visited on 08/20/2020).
- Stengel, Michael, Steve Grogorick, et al. (June 2016). “Adaptive Image-Space Sampling for Gaze-Contingent Real-time Rendering”. In: *Proc. Eurographics Conference on Rendering Techniques (EGSR) 2016* 35.4. Won the 'EGSR'16 Best Paper Award'.
- Stich, Martin, Heiko Friedrich, et al. (2009). “Spatial Splits in Bounding Volume Hierarchies”. In: *High Performance Graphics 2009*. New Orleans, Louisiana: ACM, pp. 7–13. ISBN: 978-1-60558-603-8.
- Strasburger, Hans, Ingo Rentschler, et al. (Jan. 2011). “Peripheral vision and pattern recognition: A review.” In: *Journal of vision* 11.5, pp. 1–82. ISSN: 1534-7362.

- Suykens, Frank and Yves D. Willems (2000). “Density Control for Photon Maps”. In: *Rendering Techniques 2000*. Eurographics. Vienna: Springer, pp. 23–34. ISBN: 978-3-7091-6303-0. DOI: 10.1007/978-3-7091-6303-0_3.
- Szeracki, Sebastian, Thorsten Roth, et al. (Sept. 2015). “Boosting Histogram-Based Denoising Methods with GPU Optimizations”. In: *12. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR*. Shaker Verlag.
- Tewari, A., O. Fried, et al. (2020). “State of the Art on Neural Rendering”. In: *Computer Graphics Forum* 39.2. ePrint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14022>, pp. 701–727. ISSN: 1467-8659. DOI: 10.1111/cgf.14022. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14022> (visited on 08/18/2020).
- Tole, Parag, Fabio Pellacini, et al. (July 2002). “Interactive global illumination in dynamic scenes”. In: *ACM Transactions on Graphics (TOG)* 21.3, pp. 537–546. ISSN: 0730-0301. DOI: 10.1145/566654.566613. URL: <https://doi.org/10.1145/566654.566613> (visited on 02/10/2020).
- Tracking, ART Advanced Realtime (2020). *Flystick 2 - Interaction - Products - ART Advanced Realtime Tracking*. URL: <https://ar-tracking.com/products/interaction/flystick-2/> (visited on 05/06/2020).
- University of Illinois at Chicago’s Electronic Visualization Laboratory and University of Hawai’i at Manoa’s Laboratory for Advanced Visualization and Applications (2020). *SAGE*. URL: <https://sagecommons.org/> (visited on 04/30/2020).
- Veach, Eric (1997). “Robust Monte Carlo methods for light transport simulation”. PhD thesis. Stanford University.
- Veach, Eric and Leonidas Guibas (1995a). “Bidirectional Estimators for Light Transport”. In: *Photorealistic Rendering Techniques*. Focus on Computer Graphics. Berlin, Heidelberg: Springer, pp. 145–167. ISBN: 978-3-642-87825-1. DOI: 10.1007/978-3-642-87825-1_11.
- Veach, Eric and Leonidas J. Guibas (Sept. 1995b). “Optimally combining sampling techniques for Monte Carlo rendering”. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’95. New York, NY, USA: Association for Computing Machinery, pp. 419–428. ISBN: 978-0-89791-701-8. DOI: 10.1145/218380.218498. URL: <https://doi.org/10.1145/218380.218498> (visited on 07/02/2020).
- (1997). “Metropolis light transport”. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 65–76.
- Vorba, Jiří, Ondřej Karlík, et al. (July 2014). “On-line Learning of Parametric Mixture Models for Light Transport Simulation”. In: *ACM Trans. Graph.* 33.4, 101:1–101:11. ISSN: 0730-0301. DOI: 10.1145/2601097.2601203. URL: <http://doi.acm.org/10.1145/2601097.2601203> (visited on 09/03/2015).
- Waldin, N., M. Waldner, et al. (May 2017). “Flicker Observer Effect: Guiding Attention Through High Frequency Flicker in Images”. In: *Comput. Graph. Forum* 36.2, pp. 467–476. ISSN: 0167-7055. DOI: 10.1111/cgf.13141. URL: <https://doi.org/10.1111/cgf.13141> (visited on 05/13/2019).
- Wallace, J. R., K. A. Elmquist, et al. (July 1989). “A Ray tracing algorithm for progressive radiosity”. In: *ACM SIGGRAPH Computer Graphics* 23.3, pp. 315–324. ISSN: 0097-8930. DOI: 10.1145/74334.74366. URL: <https://doi.org/10.1145/74334.74366> (visited on 07/20/2020).

- Walter, Bruce, George Drettakis, et al. (June 1999). “Interactive Rendering using the Render Cache”. In: *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. Vol. 10. Springer-Verlag, pp. 235–246.
- Wandell, Brian A. (1995). *Foundations of Vision*. Stanford University.
- Wang, Yue, Soufiane Khiat, et al. (May 2019). “Fast non-uniform radiance probe placement and tracing”. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’19. Montreal, Quebec, Canada: Association for Computing Machinery, pp. 1–9. ISBN: 978-1-4503-6310-5. DOI: 10.1145/3306131.3317024. URL: <https://doi.org/10.1145/3306131.3317024> (visited on 02/10/2020).
- Wang, Z., E. P. Simoncelli, et al. (Nov. 2003). “Multiscale structural similarity for image quality assessment”. In: *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*. Vol. 2, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.
- Wang, Zhou, A.C. Bovik, et al. (Apr. 2004). “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4. Conference Name: IEEE Transactions on Image Processing, pp. 600–612. ISSN: 1941-0042. DOI: 10.1109/TIP.2003.819861.
- Ward, Gregory J., Francis M. Rubinstein, et al. (1988). “A Ray Tracing Solution for Diffuse Interreflection”. In: *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’88. New York, NY, USA: ACM, pp. 85–92. ISBN: 0-89791-275-6. DOI: 10.1145/54852.378490. URL: <http://doi.acm.org/10.1145/54852.378490> (visited on 11/16/2015).
- Ward, Gregory and Maryann Simmons (Oct. 1999). “The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments”. In: *ACM Transactions on Graphics (TOG)* 18.4, pp. 361–368. ISSN: 0730-0301. DOI: 10.1145/337680.337722. URL: <https://doi.org/10.1145/337680.337722> (visited on 02/10/2020).
- Weier, M., M. Stengel, et al. (May 2017). “Perception-driven Accelerated Rendering”. In: *Computer Graphics Forum* 36.2, pp. 611–643. ISSN: 0167-7055. DOI: 10.1111/cgf.13150. URL: <https://doi.org/10.1111/cgf.13150> (visited on 04/21/2020).
- Weier, Martin, Jens Maiero, et al. (Sept. 2014a). “Enhancing Rendering Performance with View-Direction-Based Rendering Techniques for Large, High Resolution Multi-Display Systems”. In: *11. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR*.
- (2014b). *Lazy Details for Large High-Resolution Displays*. Published: SIGGRAPH ASIA.
- Weier, Martin, Thorsten Roth, et al. (Oct. 2016). “Foveated Real-Time Ray Tracing for Head-Mounted Displays”. In: *Computer Graphics Forum*. Vol. 35-7. Okinawa, Japan.
- Weier, Martin, Thorsten Roth, et al. (Sept. 2018a). “Foveated Depth-of-Field Filtering in Head-Mounted Displays”. In: *ACM Transactions on Applied Perception* 15.4, 26:1–26:14.
- (2018b). “Foveated Depth-of-field Filtering in Head-mounted Displays”. In: *Proceedings of the 15th ACM Symposium on Applied Perception*. SAP ’18. New York, NY, USA: ACM, 18a:1–18a:1. ISBN: 978-1-4503-5894-1. DOI: 10.1145/3225153.3243894. URL: <http://doi.acm.org/10.1145/3225153.3243894>.

- Weier, Martin, Thorsten Roth, et al. (June 2018c). “Predicting the Gaze Depth in Head-mounted Displays using Multiple Feature Regression”. In: *Proceedings of the 2018 ACM Symposium on Eye Tracking Research and Applications*. ETRA 18. ACM, 19:1–19:9.
- Whitted, Turner (June 1980). “An improved illumination model for shaded display”. In: *Communications of the ACM* 23.6, pp. 343–349. ISSN: 0001-0782. DOI: 10.1145/358876.358882. URL: <https://doi.org/10.1145/358876.358882> (visited on 06/30/2020).
- Yang, Lei, Diego F. Nehab, et al. (2009). “Amortized supersampling”. In: *ACM Trans. Graph.* 28.5, 135:1–135:12.
- Yang, Lei, Yu-Chiu Tse, et al. (2011). “Image-based Bidirectional Scene Reprojection”. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. SA ’11. New York, NY, USA: ACM, 150:1–150:10. ISBN: 978-1-4503-0807-6. DOI: 10.1145/2024156.2024184. URL: <http://doi.acm.org/10.1145/2024156.2024184> (visited on 09/07/2015).
- Zeng, Zheng, Lu Wang, et al. (Apr. 2020). “Denoising Stochastic Progressive Photon Mapping Renderings Using a Multi-Residual Network”. In: *Journal of Computer Science and Technology* 35. DOI: 10.1007/s11390-020-0264-1.
- Zhang, Cecilia (2016). *Assignment 3: PathTracer*. <https://people.eecs.berkeley.edu/~cecilia77/graphics/a3/>. Accessed: 2020-06-30.
- Zhao, Yangyang, Laurent Belcour, et al. (June 2019). “View-dependent Radiance Caching”. In: *Proceedings of the 45th Graphics Interface Conference on Proceedings of Graphics Interface 2019*. GI’19. Kingston, Canada: Canadian Human-Computer Communications Society, pp. 1–9. ISBN: 978-0-9947868-4-5. DOI: 10.20380/GI2019.22. URL: <https://doi.org/10.20380/GI2019.22> (visited on 02/10/2020).
- Zheng, Quan and Matthias Zwicker (2018). *Learning to Importance Sample in Primary Sample Space*. arXiv: 1808.07840 [cs.LG].
- Zhu, Shilin, Zexiang Xu, et al. (Apr. 2020). “Deep Photon Mapping”. In: *arXiv*. URL: <http://arxiv.org/abs/2004.12069> (visited on 07/15/2020).
- Zwicker, M., W. Jarosz, et al. (May 2015). “Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering”. In: *Computer Graphics Forum* 34.2, pp. 667–681. ISSN: 1467-8659. DOI: 10.1111/cgf.12592. URL: <http://onlinelibrary.wiley.com/doi/10.1111/cgf.12592/abstract> (visited on 09/07/2015).