EFFICIENT METHODS FOR COMPUTATIONAL LIGHT TRANSPORT

JULIO MARCO

SUPERVISED BY DIEGO GUTIERREZ AND ADRIAN JARABO

Tesis Doctoral — Ingeniería Informática
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

October 2018

To my mother.

ABSTRACT

In this thesis we present contributions to different challenges of computational light transport. Light transport algorithms are present in many modern applications, from image generation for visual effects to real-time object detection. Light is a rich source of information that allows us to understand and represent our surroundings, but obtaining and processing this information presents many challenges due to its complex interactions with matter. This thesis provides advances in this subject from two different perspectives: *steady-state* algorithms, where the speed of light is assumed infinite, and *transient-state* algorithms, which deal with light as it travels not only through space but also *time*. Our steady-state contributions address problems in both offline and real-time rendering. We target variance reduction in offline rendering by proposing a new efficient method for participating media rendering. In real-time rendering, we target energy constraints of mobile devices by proposing a power-efficient rendering framework for real-time graphics applications. In transient-state we first formalize light transport simulation under this domain, and present new efficient sampling methods and algorithms for transient rendering. We finally demonstrate the potential of simulated data to correct multipath interference in Time-of-Flight cameras, one of the pathological problems in transient imaging.

MEASURABLE CONTRIBUTIONS

This thesis has led to the following contributions and achievements:

- 5 JCR-indexed journal publications (4 of them at ACM Transactions on Graphics, 1 accepted under minor revisions at Computer Graphics Forum) [111, 179, 181, 263].
- 1 peer-reviewed article at Visual Informatics journal [113].
- 1 peer-reviewed conference publication [180].
- 1 research internship (five months) at *Disney Research Los Angeles*, Glendale, CA.
- 1 research internship (two months) at *Microsoft Research Asia*, Beijing, China.
- 1 research internship (three months) at *Adobe Research*, San Jose, CA.
- 1 patent application as a result of the internship at *Adobe Research*, San Jose, CA.
- 1 best paper award at CEIG, and 1 semifinalist poster at ACM Student Research Competition in SIGGRAPH 2017.
- Recipient of a predoctoral researcher 4-year contract from the Gobierno de Aragón, Spain.
- Reviewer for 4 journals and 12 conferences. Member of the program committee in one conference. Member of the local committee in another conference.
- Supervisor of one international undergraduate student, and another undergraduate student still in progress.
- Participation in 3 research projects.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the work, support, and courage of many people. Here I would like to mention the most meaningful ones.

Diego, for creating opportunity, for making me a better engineer and researcher, and for teaching me to unleash my hidden potential.

Adrian, for being an infinite source of motivation and knowledge, for all the the hard work and patience, and for sticking around on endless deadlines.

All my co-authors and collaborators, I thank them for their help on driving this thesis through the right course. I particularly want to thank *Adolfo*, for the morning coffees and fruitful discussions, and *Wojciech*, for all his valuable work and transmitted expertise, and for being a reference on how to do research.

My hosts during the internships. *Carol O'Sullivan*, for being my first contact with the big industry, and for placing her trust in me when I was just a lad. *Xin Tong*, for all the passion and motivation he always showed. *Xin Sun*, for his dedication and spirit, and for all the discussions beyond the main focus. I also want to thank all the fellows I met during my internships, with some special mentions. My *co-interns at Disney*, for making it an awesome place to work. *Crystal*, for bringing color and joy to my life in Beijing. And *Leo*, *Lillo*, *Julio*, and *Qi* in Adobe, for the countless "dudes" and all the shared experiences.

My colleagues at *Graphics & Imaging Lab*, past and present ones, which have helped to create an amazing research environment. Special thanks to *Carlos*, for all the good times in and out of the lab, *Ibón* for the hard work, *Manu* for all the shared knowledge, and *Quercus* for the patience. And the students I supervised, *Suraj* and *Miguel Ángel*, for their dedication.

Many people out of the workplace also deserve a mention here. All my close buddies, for being the best source of getaways from the desk. *Balma* and *Paz*, for all the many good moments since we met in the lab five years ago. All the members of *The Dry Aubergines* and our unconditional troop of fans, for all the gigs, parties, and endless practice hours. *Ani*, for sharing her craziness with me, cheering me up, and being an essential part of me this last year. My yaya *Sara*, for her eternal care and love, and for her supernatural ability to keep track of my deadlines and internships. My sister *Clara*, for not losing her spirit and being an example of hope.

Finally, I am forever in debt with my parents. My mother *Carmen*, a living example of utmost courage, resilience, and selflessness. She has greatly endured the ups and downs in my career, and my departures to distant countries, but always nurturing my development with wholehearted commitment. And my father *Joaquín*, who sparked my curiosity in computer graphics when I was just a kid, and would have loved to see what it has led to. I owe them all of this.

This thesis has been funded by the Gobierno de Aragón, the European Social Fund, the European Research Council (Consolidator Grant, project Chameleon, ref. 682080), DARPA (project REVEAL, HR0011-16-C-0025), and the Spanish Ministry of Economy and Competitiveness (project LIGHTSPEED, ref. TIN2016-78753-P).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

INTRODUCTION & OVERVIEW

1 INTRODUCTION

Computational methods for light transport have been extensively researched in computer graphics and vision, with numerous applications in industries such as entertainment, architecture, robotics, astronomy, or medicine. Light transport simulation is the basis of both offline and real-time rendering, providing synthetic imagery of many sorts for movie production, video games, or product prototyping. But beyond the generation of classic 2D images for visualization, it can also be a powerful forward model to predict and analyze light behavior in other complex problems. Since the emergence of computational imaging [271], many imaging systems operate with *extra* information about light propagation: Light field cameras are able to capture multiple views of a scene in a single shot, hyperspectral devices capture richer information in the electromagnetic spectrum, and high-dynamic range imaging systems can increase the range of light intensity supported in both capture and display [94, 232, 248]. Like many other systems whose main source of information is light, these rely on some sort of light transport processing, and therefore providing accurate and efficient methods for light transport simulation and analysis is an important task.

Monte Carlo methods are nowadays the cornerstone for offline realistic image synthesis [205, 292], and have become a robust workhorse in modern production rendering engines [53]. Despite Monte Carlo rendering being a mature field, research in this direction is still thriving, comprising novel methods that improve computational efficiency, increase accuracy in the results, or allow handling complex light transport phenomena unexplored before [109]. Although variance reduction remains as one of the perpetual challenges in Monte Carlo methods, some other research trends include for example lifting long-standing rendering assumptions of classic radiative transfer [114], or devising practical models for complex appearance such as multi-layered materials [12, 288].

While offline rendering still demands solutions to long-standing challenges such as variance reduction, during the last years the advances in hardware—either for imaging, computation, or display—have opened new challenges and possibilities in many light transport applications. The increase of computational power and memory in GPUs, and their integration in consumer-level devices have escalated to a point where we can generate high-quality renders in real time with a simple swipe of our fingers. The availability of computationally-intensive hardware in our pockets opens new challenges beyond the usual rendering problems, such as high battery consumption in rendering applications. On a more general scope, modern GPUs have made deep learning a viable tool to address many problems in computer vision and graphics. Both offline and real-time rendering benefit from these optimization approaches, for example for denoising [11], real-time shading [199], or rendering participating media [136]. With the combination of deep learning and rendering [141], the need of synthetic data for training faces challenging problems such as massively generation of data and proper domain exploration.

Breaking through traditional imaging, devices that are capable of capturing light at frame-rates comparable to its speed have become more accessible to both the research community and the general public. This has given rise to a vast amount of methods within the field of *transient imaging* [113], which leverage information in the temporal domain of light propagation for applications such as material recognition [195, 278] or hidden-object detection [79, 257]. These methods have led to a high demand of reliable *transient* light transport data, in contrast to the *steady-state* data obtained with conventional cameras and traditional synthesis methods. However, capturing transient light transport data is either slow, expensive, or its accuracy is limited by the available hardware. Simulating light in motion is of key importance for the development of the field, allowing to generate accurate data under controlled setups, for benchmarking and prototyping, as a forward model for inverse problems, and as a source for machine learning. This sort of simulations are referred to as *transient rendering*. While research on the latter has been alive for nearly half a century, research on transient rendering is still at an early stage, requiring novel approaches that explore and address problems on the temporal domain of light transport.

GOAL In a moment when research in steady- and transient-state light transport is very active, this thesis presents contributions in both sides, organized in two different parts. In Part II we target both long-standing problems and novel challenges in rendering from a classic steady-state perspective, with contributions in efficient offline rendering of participating media, and energy-efficient rendering for real-time applications. Part III is dedicated to light transport in transient state, where we set the grounds for a principled light transport simulation framework, identify and provide solutions to different challenges in this domain, and demonstrate the potential of synthetic data generation for transient imaging problems. In the following we give a brief overview of these two parts.

1.1 STEADY-STATE LIGHT TRANSPORT

The first ray-casting algorithm for image generation was introduced to computer graphics by Appel in 1968 [7]. Since the formulation of distributed ray tracing for global illumination by Cook et al. in 1984 [37], and the subsequent introduction of the rendering equation by Kajiya in 1986 [133], research on physically-based rendering has come a long way. Aiming to mimic the behavior of a conventional camera, steady-state rendering methods generate synthetic 2D images by computing light transport on a scene based on mathematical definitions of objects, materials, media, cameras, and light sources. High-quality 2D imagery is omnipresent in our lives, in fields such as movie production, digital prototyping, architecture, video games, or medical imaging. With ray-optics as the common ground in offline and real-time rendering, each of these have followed different paradigms in order to synthesize realistic 2D imagery. In the following we give an overview of different challenges and trends in offline and real-time rendering, and summarize the contributions of this thesis related to each one of them.

OFFLINE RENDERING Offline methods are mainly based on Monte Carlo integration. By stochastically sampling light transport equations, they are able to produce hyper-realistic results that accurately represent light interacting with matter. One of the pathological problems of these methods is the visible noise in the resulting images, consequence of variance in the Monte Carlo estimators. This variance is not uniform in the whole image, and its behavior depends on the scene configuration and the methods used to sample the transport equations. Increasing the sampling rate uniformly in the whole image is usually a bad practice, since some regions may converge faster than others, specially in the case of participating media. Research on variance reduction methods is therefore one of the main trends in rendering [292]: By leveraging information implicit on the light transport equations or obtained during the rendering process we can devise smarter sampling and reconstruction algorithms that adaptively render arbitrary scenarios. In this thesis we present contributions in offline rendering of participating media by proposing a new algorithm that adaptively samples media and performs error-bounded interpolations based on radiance derivatives (see Chapter 2).

REAL-TIME RENDERING Real-time methods in the other hand, rely mainly on rasterization pipelines, where geometry is projected towards the image plane and the pixel color is computed in a shading operation. This shading operation may or not be based on stochastic sampling of light transport equations, but in general makes strong assumptions about light transport to guarantee real-time frame rates. Nonetheless, a great amount of graphics processing units (GPU) are purposely designed for this sort of processes, and their computational power is able to generate graphically rich content. Moreover, GPUs have already flooded smartphones, tablets, and hand-held game consoles, targeted to convey real-time high-quality imagery under battery-powered devices. In a world where battery life has equaled in importance to computational power and memory, power usage in graphic processors comes into the fold of efficiency standards. GPU designers have already put power efficiency as a prerequisite when building their architectures, and not only targeted for battery life optimization, but also aiming for environmental-friendly hardware [99, 194]. From a software perspective, however, most research has mainly focused in providing faster rendering pipelines, while reducing their energy consumption is still an unexplored subject. In this thesis, we present contributions on this aspect by introducing a software-based real-time framework for energy-aware rendering (see Chapter 3). Our framework precomputes energy maps of virtual scenarios for different effects in the rendering pipeline, and computes the optimal settings in runtime to minimize power usage and maximize image quality.

1.2 TRANSIENT LIGHT TRANSPORT

Transient imaging refers to a series of methods that make use of light transport at temporal resolutions comparable to its speed, focusing on capture and simulation of light in motion, and exploiting the temporal information of light transport for scene understanding and reconstruction. Some of these applications, such as real-time depth estimation, are widely available through off-the-shelf products. Other applications, such as reconstruction of *non-line-of-sight* geometry [79, 131, 165, 257], in general require more complex machinery, higher computational power, or are limited to controlled scenarios. While transient imaging devices are becoming more and more available to both researchers and end-users, the hardware characteristics usually limit the accuracy on the captured data. Providing practical methods that increase the quality of the results without hindering the capture process or the range of applicability is one of the main challenges in transient imaging. In that sense, simulating transient light transport has proved to be an effective tool for transient imaging problems, either as a source of reliable ground-truth data [206], as a forward model in optimization approaches [57, 58, 123], or for method prototyping and evaluation [198]. Research on transient rendering is therefore of high importance, since well-established steady-state methods may not longer be appropriate in the temporal domain of light propagation.

In the following we first give an overview of transient light transport simulation and its challenges, and briefly describe our contributions to tackle these. Second we summarize the core concepts of one of the long-standing problems in transient imaging, and introduce our contributions to address it.

EFFICIENT METHODS FOR TRANSIENT RENDERING Traditional steady-state rendering simulates how a conventional camera images a scene. Since light propagates at around 300,000 km/s, in a single shot of a conventional camera light travels several hundreds of kilometers, and the frame captures the scene with light fully propagated. Therefore, steady-state rendering assumes speed of light is infinite, discarding any temporal information. Transient rendering breaks this assumption, and "unfolds" light transport over the temporal domain by accounting for the light propagation delays due to the optical path and light-matter scattering events. In practice this implies that every pixel of a regular 2D image is now resolved in time, effectively becoming a 3D volume. But adapting existing steady-state rendering methods to transient-state presents many challenges. First, we are sampling a higher-dimensional space, so it requires a much higher number of samples to achieve the same sampling density at every frame in the temporal domain. Second, steady-state methods for light transport simulation are usually radiance-driven, in the sense that they focus on sampling light paths that have higher contribution to the radiance signal. Since light intensity decays at each interaction with matter due to absorption, steady-state algorithms tend to generate more samples at short light paths. In transient rendering, this generates exponentially-decaying distributions of samples over time, and therefore variance becomes aggravated at later timings. To address these issues it is necessary to provide a proper formalization and analysis of Monte Carlo methods in transient-state that allows characterizing the existing challenges in a principled way, and providing a solid framework for upcoming research efforts in transient rendering.

In this thesis we contribute to transient rendering by extending Veach's path integral formulation [254] to the transient domain, analyze and address sampling issues in this extended domain of light transport, and provide a consistent method for signal reconstruction in the temporal domain (see Chapter 5). Next, we focus on transient rendering of participating media, and adapt photon beams methods [118] to increase sampling density in the temporal domain (Chapter 6). Based on our previous framework, we formulate a consistent progressive method for transient light transport in participating media that significantly mitigates variance over time.

MULTIPATH INTERFERENCE IN TOF IMAGING A long-standing problem in transient imaging applications is multi-path interference (MPI). While the propagation of a light path occurs at a single instant in time, the limited temporal resolution of transient devices makes light paths with different timings to fall simultaneously at the sensor, therefore *interfering* with each other.

One common application of transient imaging is depth estimation, where the distance of a visible object can be estimated by emitting light towards an object from a light source co-located with the sensor. The Time-of-Flight of a single light bounce will determine the distance to the object. However, MPI due to surrounding geometry and limited temporal resolution may introduce some errors in this estimation, since longer indirect light paths may arrive in the same frame captured by the sensor. This problem is particularly aggravated in the so-called Time-of-Flight (ToF) depth cameras, which rely on correlating a continuous emission of modulated light, and have exposure times in the order of nanoseconds. These cameras work with baseline depth ranges resolutions from a centimeter to several meters. During a single exposure of these cameras the *interfering* indirect paths may have already propagated across the whole scene, and therefore the introduced MPI leads to a significant depth overestimation. While previous works have addressed this problem, they usually require complex hardware modifications, or computationally intensive methods.

As a final contribution of this thesis we address the problem of multipath interference in ToF depth capture avoiding these two disadvantages (Chapter 7). Parting from the previous contributions of this thesis, we rely on simulated transient data to analyze multipath interference on depth images. Thanks to controlled setups provided by transient rendering, we mimic the behavior of ToF cameras in simulation and obtain MPI-tampered depth maps, along with their reference solutions. We then use these to train real-time corrections of MPI in synthesis using a hybrid approach that combines both real and synthetic data.

1.3 CONTRIBUTIONS AND MEASURABLE RESULTS

In the following we state the publications which support the contributions of this thesis. For the publications in which I am not the lead author, my particular contribution is detailed at the beginning of their corresponding chapter in the thesis.

1.3.1 *Publications*

- *Second-Order Occlusion-Aware Volumetric Radiance Caching* (Chapter 2). This work has been published at ACM Transactions on Graphics [181] in 2018, which has an impact factor of 4.384, ranked 3/104 (Q1) in the "Computer Science, Software Engineering" category of the JCR index (data from 2017).
- *Real-time Rendering on a Power Budget* (Chapter 3). This work has been published at ACM Transactions on Graphics [263] in 2016, with an impact factor of 4.088, ranked 1/106 (Q1) in the "Computer Science, Software Engineering" category of the JCR index.
- *Recent Advances in Transient Imaging: A Computer Graphics and Vision Perspective*. This work has been published at Visual Informatics [113], which is an Elsevier journal recently created in 2017, with no available impact factor. Chapter 4 is written based on the excerpts of this publication that most relate to the topics covered in the remainder of Part III.
- *A Framework for Transient Rendering* (Chapter 5). This work has been published at ACM Transactions on Graphics [111] in 2014, with an impact factor of 4.096, ranked 1/104 (Q1) in the "Computer Science, Software Engineering" category of the JCR index.
- *Progressive Transient Photon Beams* (Chapter 6). This article has been conditionally accepted under minor revisions at Computer Graphics Forum, with an impact factor of 2.046, ranked 22/104 (Q1) in the "Computer Science, Software Engineering" category of the JCR index. It is an extension of the article *Transient Photon Beams* [180] which was accepted at the Spanish Conference in Computer Graphics (CEIG), 2017, and received one of the two *Best paper* awards in the conference. Additionally, a poster based on this article was accepted at SIGGRAPH 2017, and ended semifinalist on the ACM Student Research Competition.
- *DeepToF: Off-the-Shelf Real-Time Correction of Multipath Interference in Time-of-Flight Imaging* (Chapter 7). This work has been published at ACM Transactions on Graphics [179], which has an impact factor of 4.384, ranked 3/104 (Q1) in the "Computer Science, Software Engineering" category of the JCR index. This project started during my internship at Microsoft Research Asia.

1.3.2 *Internships*

This thesis has also led to the following research internships:

- A five-month internship at *Disney Research Los Angeles* (Glendale, CA), supervised by Carol O'Sullivan.
- A two-month internship at *Microsoft Research Asia* (Beijing, China), supervised by Xin Tong. This internship led to the publication of one of the articles of this thesis [179] (Chapter 7).
- A three-month internship at *Adobe Research* (San Jose, CA), supervised by Xin Sun. This internship has resulted in a patent application.

Part II

STEADY-STATE LIGHT TRANSPORT

2 SECOND-ORDER OCCLUSION-AWARE VOLUMETRIC RADIANCE CACHING

Variance reduction is one of the long-standing problems in offline rendering. Storing and reusing the sampled light paths during the rendering process has proved to be an effective approach to reduce it, such as in many-lights [87, 140, 203, 204], photon-density estimation [21, 117, 118, 122], or radiance caching methods [116, 152, 230, 267]. In this chapter we propose an improved radiance caching method for participating media rendering. Our method overcomes many issues of previous radiance caching methods by providing an error metric based in second-order derivatives, and presenting a more accurate derivative computation method that accounts for occlusions in single and multiple scattering. We analyze and illustrate the benefits of our method in 2D media, and demonstrate how these benefits extend to complex 3D scenarios.

This work was published in ACM Transactions on Graphics and presented at SIGGRAPH 2018. A preliminary version of this work was also presented as a poster at SIGGRAPH 2017.

J.Marco, A. Jarabo, W. Jarosz & D. Gutierrez
Second-Order Occlusion-Aware Volumetric Radiance Caching
ACM Transactions on Graphics, Vol.37(2)
(Presented at SIGGRAPH 2018)

J. Marco, A. Jarabo, W. Jarosz & D. Gutierrez
Second-Order Occlusion-Aware Volumetric Radiance Caching
SIGGRAPH 2017 Posters

2.1 INTRODUCTION

Accurately simulating the complex lighting effects produced by participating media in the presence of arbitrary geometry remains a challenging task. Monte Carlo-based methods like path tracing numerically approximate the radiative transfer equation (RTE) [28] by stochastically sampling radiance in the medium. These approaches can handle complex geometry and general scattering properties, but since they lack memory and are largely blind to the radiance signal, they perform many redundant computations leading to high cost. A common strategy to increase efficiency is to adaptively sample radiance based on its frequency content, limiting the sampling density in regions where radiance barely changes, and placing more samples in regions with higher frequency variation [292].

Based on this principle, *volumetric radiance caching* [116] computes and stores radiance at sparse cache points in the medium, and uses these samples to reconstruct radiance at nearby locations whenever possible. The method is based on first-order translational derivatives of the radiance, which are used to i) determine how far away a cache point can be reused while controlling error, and ii) improve reconstruction quality by extrapolating the cached radiance values along their gradients. Unfortunately, since the gradient derivations ignore occlusion/visibility changes, the method fails in scenes containing occluders where changes in visibility are the dominant factor in local radiance behavior. Moreover, the reconstruction and error metric both rely on the same gradient estimates and ignore variations caused by higher-order derivatives. These factors lead to suboptimal cache point distributions, which fail to properly sample high-frequency features such as occlusions, while simultaneously oversampling other regions of the scene. This results in reduced efficiency and visible rendering artifacts.

Second-order illumination derivatives have proven to be a powerful and principled tool for sparsely sampling and interpolating surface irradiance [120, 230], as well as controlling error in density estimation techniques [13, 84, 137]. Inspired by these recent developments, we propose a new second-order, *occlusion-aware* radiance caching method for participating media which overcomes the limitations of current state-of-the-art methods.

To this end, we introduce a novel approach to compute first- and second-order occlusion-aware derivatives of both single and multiple scattering, and generalize the Hessian-based metric of Schwarzhaupt et al. [230] for controlling the error introduced by first-order extrapolation of media radiance. In addition, we extend recent work on 2D radiometry, currently limited to surfaces [120], and derive a 2D theory of light transport in participating media. We use this framework to illustrate and analyze the limitations of the state of the art, as well as the benefits of our proposed method. We demonstrate the generality of our approach by deriving occlusion-aware derivatives of 3D media radiance and applying our Hessian-based metric to 3D cache distributions, showing that the benefits predicted by our 2D analysis hold equally in 3D. Our approach improves volumetric cache point distributions in isotropic homogeneous media, providing a significantly more accurate reconstruction of difficult high-frequency features, as Figure 2.8 shows.

2.2 RELATED WORK

We summarize here existing work on radiance caching methods as well as other techniques that leverage illumination derivatives to improve Monte Carlo rendering. For a general overview of scattering and existing adaptive sampling and reconstruction techniques, we refer the reader to other recent sources of information [81, 292].

Radiance caching: Irradiance caching was originally proposed by Ward et al. [267] to accelerate indirect illumination in Lambertian scenes. The method computes and caches indirect irradiance only at a sparse set of points in the scene, and extrapolates or interpolates these values whenever possible from cache points deemed to be sufficiently close by. Since indirect illumination changes slowly across Lambertian surfaces, the costly irradiance calculation can often be reused over large parts of the image, substantially accelerating rendering. There has been a wealth of improvements to irradiance caching, but we discuss only the most relevant follow-up work and refer to the work by Křivánek and Gautron [152] for a more complete survey.

Ward and Heckbert [266] significantly improved reconstruction by leveraging gradient information, and Krivánek et al. [155] incorporated heuristics to improve error estimation (and therefore quality) during adaptive caching. Křivánek and colleagues [153, 154] also extended irradiance caching to handle moderately glossy, non-Lambertian surfaces. Herzog et al. [95] used anisotropic cache points based on the orientation of the illumination gradient. All these methods only considered *surface* light transport.

Jarosz et al. [116] proposed *volumetric* radiance caching, which accelerates single and multiple scattering in participating media. They proposed an error metric based on the first-order derivative of the radiance, but their formulation ignored volumetric occlusion changes. In follow-up work, Jarosz et al. [115] derived occlusion-aware gradients, but only of surface illumination in the presence of absorbing and scattering media, ignoring gradients of the media radiance itself. Both approaches are prone to suboptimal cache point distributions and visible artifacts since they ignore higher order derivatives or occlusion changes in media. Our work addresses both of these issues. Ribardière et al. [223] proposed using anisotropic cache points and a second-order expansion for radiance reconstruction. Their approach, however, did not consider visibility changes due to their point-to-point computation of derivatives.

Recently, Jarosz et al. [120] and follow-up work [230] made significant progress in heuristics-free error control for surface irradiance caching by formulating error in terms of second-order derivatives. In particular, Schwarzhaupt et al. [230] proposed a novel radiometrically equivalent formulation of irradiance gradients and Hessians, which properly accounted for occlusions. The authors used these for extrapolation and principled error control, respectively. We extend these ideas and apply them to light transport in participating media, deriving first- and second-order occlusion-aware derivatives for improved reconstruction and principled error control in volumetric radiance caching.

Differential domain: Arvo [10] derived closed form expressions for irradiance derivatives in polygonal environments, and Holzschuch and Sillion [97] and Holzschuch and Sillion [98] derived second-order illumination derivatives for error control in the radiosity algorithm. Local differentials have also proven useful for texture filtering [103, 246], photon density estimation [118, 228], and spectral rendering [50]. Ramamoorthi et al. [220] analyzed gradients of various surface lighting effects, including occlusions, and showed how these can be used for adaptive sampling and interpolation in image space. Lehtinen et al. [168] and follow-up work [176], proposed to compute image gradients instead of actual luminance values in Metropolis light transport (MLT), and feed a Poisson solver with these gradients to reconstruct the final image. Later work [144, 177] extended the applicability of this gradient domain idea to simpler Monte Carlo path tracing methods, and demonstrated how solving light transport in the gradient domain improves over primal space, while remaining unbiased. Rousselle et al. [225] showed how such Poisson-based reconstruction approaches can be directly formulated as control-variate estimators. Kaplanyan and Dachsbacher [137] leveraged second-order derivatives of irradiance to estimate optimal kernel bandwidth in progressive photon mapping, focusing on surface light transport only.

Closely related to our work, Belcour et al. [13] performed a frequency analysis of light fields within participating media. They summarize the local light field using covariance matrices, which provides Hessians of fluence (up to sign) due to scattering and absorption. Their approach explicitly accounts for radiance changes only in the plane perpendicular to ray propagation, needing to average the per-light-path information from many rays to compute the 3D fluence spectrum. To account for visibility changes, they also require precomputing the covariance matrices in a finite neighborhood, sacrificing locality and incurring the cost of scene voxelization. In contrast, we provide a fully local method for computing first- and second-order derivatives of media radiance, without requiring voxelization, all while accounting for changes due to visibility, scattering, and transmittance.

2D spaces: Simplification to lower-dimensional spaces is a recurring tool used in problem analysis. In image synthesis, reduction to hypothetical 2D worlds has been used to obtain insights and illustrate the benefits of more complex 3D approaches [91, 207]. More recent analyses of derivative and frequency domains [46, 185, 220], as well as recent work on complex reflectance filtering [283, 284] reduce the complexity

Table 2.1: Notation for the optical properties of participating media, and their differences in 3D and 2D.

		3D		2D	
Quantity	Symbol	Expression	Units	Expression	Units
Particle density	ρ	Particles per unit volume	$1/\mathrm{m}^3$	Particles per unit area	$1/\mathrm{m}^2$
Cross-section	σ	Area	m^2	Length	m
Scattering coefficient	μ_s	Probability density per differential length	1/m	Probability density per differential length	1/m
Absorption coefficient	μ_a	Probability density per differential length	1/m	Probability density per differential length	1/m
Extinction coefficient	μ_t	$\mu_t = \mu_a + \mu_s$	1/m	$\mu_t = \mu_a + \mu_s$	1/m
Transmittance	T_r	$T_r(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\int_{\mathbf{x}_1}^{\mathbf{x}_2} \mu_t(\mathbf{x}) \mathrm{d}\mathbf{x})$	unitless	$T_r(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\int_{\mathbf{x}_1}^{\mathbf{x}_2} \mu_t(\mathbf{x}) \mathrm{d}\mathbf{x})$	unitless
Phase Function	$f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)$	Angular scattering of light at a point	1/sr	Angular scattering of light at a point	1/rad

of their derivations by performing them in 2D, before showing how the gained insights generalize to 3D. Jarosz et al. [120] introduced a 2D surface radiometry and global illumination framework, and showed how this allows for a more practical analysis of 2D versions of standard rendering algorithms due to faster computation and simpler visualization. Other fields such as acoustic rendering have recently benefited from 2D reduction to provide interactive simulations [4]. Two-dimensional simulations have also been proved useful to synthesize higher-dimensional light transport, as in transient rendering [20, 111]. In this work we follow a similar methodology as Jarosz et al. [120], providing a novel 2D radiometry framework for participating media.

2.3 2D AND 3D LIGHT TRANSPORT IN PARTICIPATING MEDIA

We describe here the main radiometric aspects of working in a two-dimensional domain, compared to 3D. Similar to Jarosz et al. [120], we assume an *intrinsic model* where light is generated, scattered, and absorbed within a plane, thus ensuring energy conservation.

The outgoing radiance at a point $\mathbf{x}$ in a medium is defined as the angular integral of the incident radiance $L_i(\mathbf{x}, \vec{\omega}_i)$, modulated by the scattering phase function $f_s(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)$:

$$L(\mathbf{x}, \vec{\omega}_o) = \int_\Omega f_s(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)\, L_i(\mathbf{x}, \vec{\omega}_i)\, \mathrm{d}\vec{\omega}_i, \tag{2.1}$$

where $\vec{\omega}_i$ and $\vec{\omega}_o$ are directions over the spherical domain Ω pointing into and out of the point $\mathbf{x}$ respectively. The incident radiance $L_i = L_m + L_s$ is the sum of radiance arriving from the surrounding medium (L_m) and from surfaces (L_s):

$$L_m(\mathbf{x}, \vec{\omega}_i) = \int_0^s \mu_s(\mathbf{y}(t))\, T_r(\mathbf{x}, \mathbf{y}(t))\, L(\mathbf{y}(t), \vec{\omega}_i)\, \mathrm{d}t, \tag{2.2}$$

$$L_s(\mathbf{x}, \vec{\omega}_i) = T_r(\mathbf{x}, \mathbf{y}_s)\, L_o(\mathbf{y}_s, \vec{\omega}_i), \tag{2.3}$$

where $\mathbf{y}(t) = \mathbf{x} - t\vec{\omega}_i$ is a point in the medium, and $\mathbf{y}_s$ is a point on a surface at distance s with outgoing radiance L_o modeled by the rendering equation [133]. The transmittance T_r models the attenuation due to scattering and absorption between two points, and $\mu_s(\mathbf{x}) = \rho\sigma_s$ is the scattering coefficient at $\mathbf{x}$, with ρ and σ_s the density and scattering cross-section in the medium, respectively. We detail our notation in Table 2.1, and highlight the main radiometric differences between self-contained 2D and 3D worlds, described below.

Differences in 2D: When moving to a 2D world, the intrinsic radiometric model implies that all radiance travels within a planar medium, scattering therefore over angle instead of solid angle. This means that radiance falls off with the inverse distance instead of inverse squared

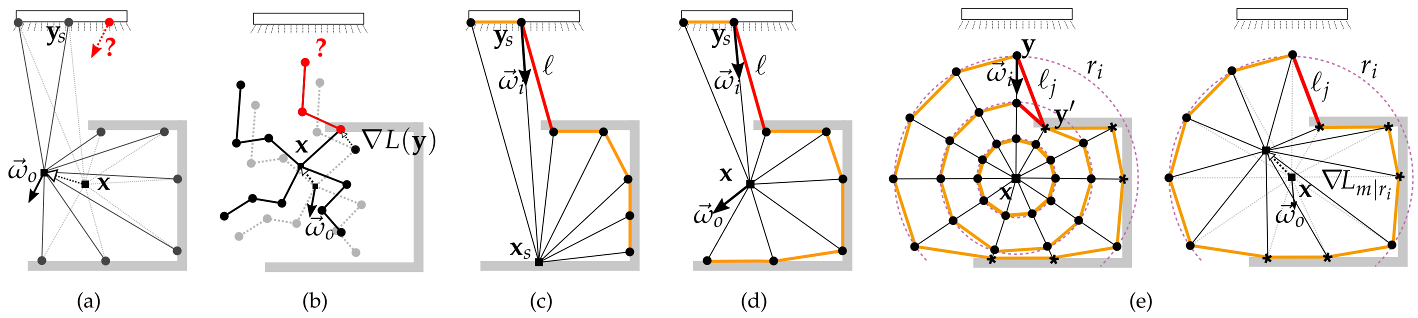


Figure 2.1: Jarosz et al. [116] point-to-point approach for computing first-order derivatives of single (a) and multiple (b) scattering ignores radiance that becomes occluded/disoccluded (red) as $\mathbf{x}$ is translated. Schwarzhaupt et al. [230] compute occlusion-aware derivatives (c) of diffuse surface irradiance by considering the occlusion-free subdivision (orange) of surrounding geometry as seen from $\mathbf{x}_s$. We compute occlusion-aware first- and second-order derivatives (d,e) by constructing such occlusion-free subdivisions (orange) of the scene, both at surface locations for single scattering (similar to Schwarzhaupt's work), and also at ray-marched media locations for multiple scattering. Red segments represent approximations of both single and multiple scattering occlusions. Starred points $\star$ in (e) represent black samples at surfaces that occlude radiance from media.

distance [120]; this will become important in our analysis of first- and second-order derivatives.

The main changes when applying Equations (2.1–2.3) in 2D are:

- The integration domain Ω of Equation (2.3) becomes circular instead of spherical.
- The phase functions in 2D must be normalized over the circle, not the sphere, of incident directions.
- $L_o(\mathbf{y}_s)$ now indicates radiance from the closest curve (the 2D equivalent of a 3D surface).

In the next sections, we use this self-contained 2D world to better depict and reason about the improvements of our new occlusion-aware gradients and Hessians for media (Section 2.4), and our second-order error metric (Section 2.5), before extending them to a more practical three-dimensional world. Working in 2D also allows us to avoid collapsing a 3D scene into a 2D image for visualization, where information from many media points would contribute to a single image pixel. This allows us to illustrate the performance of our algorithm in a more intuitive way (Section 2.6) and to depict the introduced errors more clearly.

2.3.1 *Radiance Caching in Participating Media*

Before deriving our second-order, occlusion-aware volumetric radiance caching approach, we first summarize Jarosz et al.'s [116] original formulation. To determine the radiance at any point $\mathbf{x}'$ in the medium[1], their algorithm first tries to approximate this value by extrapolating (in the log domain) the cached radiance L_k from nearby cache point locations $\mathbf{x}_k$ along their respective gradients:

$$L(\mathbf{x}', \vec{\omega}_o) \approx \exp\left[\frac{\sum_{k\in C} (\ln L_k + \nabla \ln L_k \cdot \Delta_{\mathbf{x}'})\, \mathrm{w}(\mathbf{x}_k, \mathbf{x}')}{\sum_{k\in C} \mathrm{w}(\mathbf{x}_k, \mathbf{x}')}\right], \qquad (2.4)$$

with $\Delta_{\mathbf{x}'} = (\mathbf{x}' - \mathbf{x}_k)$. Here $\nabla \ln L_k = \nabla L_k / L_k$ is the log-space translational gradient of cache point $\mathbf{x}_k$, and $\mathrm{w}(\mathbf{x}_k, \mathbf{x}')$ is a weighting function that diminishes the influence of a cache point to zero as $\mathbf{x}'$ approaches the cache point's valid radius. The collection of nearby cache points C consists of all cache points whose valid radii contain $\mathbf{x}'$. If no nearby cache points are found, then the algorithm computes radiance using Monte Carlo sampling and inserts the value and its gradient into the cache for future reuse.

[1] Throughout the text, $\mathbf{x}'$ represents points where we approximate radiance by interpolating the cache points, while $\mathbf{x}$ represents points where we compute radiance and its derivatives explicitly.

Jarosz et al. [116] proposed to compute the valid radii using a metric based on the local log-space radiance gradient:

$$R = \varepsilon \frac{\sum L_j}{\sum \|\nabla L_j\|}, \tag{2.5}$$

where ε is a global error tolerance parameter and L_j and ∇L_j are the individual Monte Carlo samples of radiance and translational gradient respectively. Unfortunately, this error metric is an ad-hoc approximation of the error in the log-scale interpolation, which can lead to difficulty predicting the error in the sample distribution and suboptimal cache distributions.

Jarosz et al. maintain a separate cache for single/surface scattering and multiple scattering. They compute single-scattering gradients by Monte Carlo sampling the first translational derivative of Equations (2.1) and (2.3) in surface-area form. They trace out many rays in the sphere of directions around point $\mathbf{x}$ to obtain a number of surface hit points $\mathbf{y}_s$. Their gradient calculation, in essence, considers how the radiance L_s from each of these hit points would change (due to changes of transmittance and geometry terms, but not visibility) as $\mathbf{x}$ translates, but the surface hit points $\mathbf{y}_s$ remain fixed (see Figure 2.1a). For multiple-scattering gradients, they Monte Carlo sample the first derivative of Equations (2.1) and (2.2), where the whole set of sampled paths is assumed to move rigidly (see Figure 2.1b), accounting for translational derivatives at each scattering vertex.

This gradient formulation can efficiently compute the local change in radiance of any single Monte Carlo sample, but—by operating independently on each radiance sample—it is not able to capture *global* effects such as visibility gradients. As a consequence, changes in radiance that becomes occluded/unoccluded as the shaded point is translated are not taken into account (see Figures 2.1a and 2.1b, red). As an illustrative example, Figure 2.2 shows how ignoring occlusions (purple line) leads to incorrect single- and multiple-scattering gradients in the penumbra region beneath the occluder.

In the remainder of this work we describe our novel Hessian-based radiance caching method for participating media that overcomes the aforementioned limitations. In Section 2.4 we introduce our approach for computing occlusion-aware first- and second-order derivatives of media radiance. Then, in Section 2.5 we introduce our Hessian-based error metric and extrapolation method for volumetric radiance caching.

2.4 RADIOMETRIC DERIVATIVES IN MEDIA

Following the work of Schwarzhaupt et al. [230] on global illumination on surfaces, we formulate the radiance at $\mathbf{x}$ as a piecewise linear

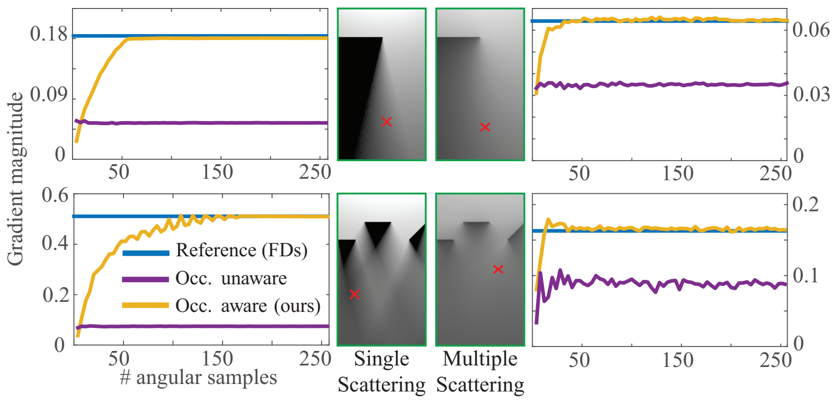


Figure 2.2: Compared to prior occlusion-unaware gradients (purple), our gradients (yellow) converge to the reference solution (blue) with increasing angular sample count both for single scattering (left) and multiple scattering (right). The convergence plots are computed in the red crosses in the respective middle images.

representation of the incoming radiance. Conceptually, we build an approximated coarse representation of the scene as seen from the media point $\mathbf{x}$ by triangulating adjacent stochastic angular samples $\mathbf{y}_s$ (see Figures 2.1c and 2.1d). The interesting property of this triangulation is that the geometry term for each triangle (segments in 2D) models the attenuation due to the solid angle; as a consequence, changes in the geometry term (due to translation of $\mathbf{x}$) model changes in the observed radiance.

We extend Schwarzhaupt et al.'s [230] formulation to handle not only light transport from surfaces, but also from media. In the case of surfaces, the sample points $\mathbf{y}_s$ are located at the first surface point as seen from $\mathbf{x}$ in direction $\overrightarrow{\mathbf{yx}}$ (Figure 2.1d). For points in a participating medium, however, radiance arrives from multiple distances along each direction. We therefore consider a set of concentric triangulations at increasing distances r_i, each representing the outgoing radiance at that particular distance in the medium. If occluding geometry exists closer than the distance r_i, we place a zero-radiance sample at the surface intersection (points marked with $\star$ in Figure 2.1e).

Handling Occlusions and Transmittance: In essence, we are approximating the integration along Ω, by transforming the scene into a discrete set of *virtual* piecewise linear representations of the geometry and media around $\mathbf{x}$. As noted by Schwarzhaupt et al., this representation implicitly encodes changes in visibility by means of the geometry term. Our approach for media, however, requires taking transmittance into account and using different geometry terms (see Figure 2.1c), since surface-medium light transport only has a cosine term at the source

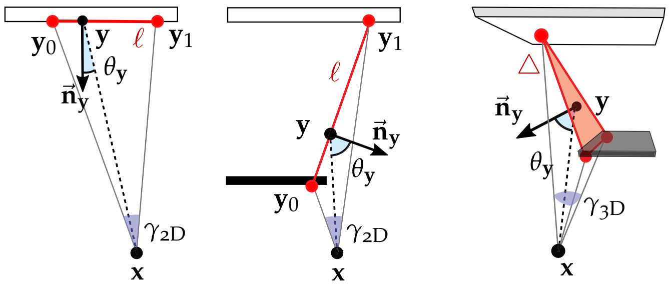


Figure 2.3: Left and center: Visible and occluded cases for 2D surface-media radiance for an angle γ. Red segment represent the piecewise-linear construction as seen from $\mathbf{x}$. Right: 3D interpretation, where occlusions are represented by slanted triangles, and visibility changes are modeled as changes in the 3D geometry term between the triangle points $\mathbf{y} \in \triangle$, and $\mathbf{x}$.

$\mathbf{y}_s$. We illustrate this with a 2D example in Figure 2.3, left and center: Assuming a constant angle γ between vectors $\overrightarrow{\mathbf{x}\mathbf{y}_0}$ and $\overrightarrow{\mathbf{x}\mathbf{y}_1}$, occlusions generate segments $\ell = \mathbf{y}_1 - \mathbf{y}_0$ at grazing angles, with derivatives proportional to the steepness of the segment. When moving within the medium, the projected angle of ℓ towards $\mathbf{x}$ is proportional to $\cos\theta_\mathbf{y}$, and therefore the radiance from ℓ increases with $\cos\theta_\mathbf{y}$. This allows modeling the visibility changes as a change on the 2D geometry term $G = \cos\theta_\mathbf{y} / \|\overrightarrow{\mathbf{x}\mathbf{y}}\|$. This principle holds also for 3D, as Figure 2.3, right, shows: Occlusions are represented by slanted triangular faces, and visibility changes are modeled as changes in the 3D geometry term between the triangle points $\mathbf{y} \in \triangle$ and $\mathbf{x}$. We leverage this equivalence to provide a unified formulation for radiance derivatives, applicable both to 2D and 3D[2].

Using the formulation presented before, we approximate $L(\mathbf{x}, \vec{\omega}_o)$ by discretizing the space into a set of concentric rings $\mathcal{R}$ as:

$$L(\mathbf{x}, \vec{\omega}_o) \approx \sum_{r_i \in \mathcal{R}} \frac{1}{\mathrm{pdf}(r_i)} \sum_{\ell_j \in \mathcal{L}_i} L_j(\mathbf{x}, \vec{\omega}_o), \tag{2.6}$$

where the last ring $r_s \in \mathcal{R}$ has all its vertices on surfaces, $\mathcal{L}_i$ is the set of segments for ring r_i, and $\mathrm{pdf}(r_i)$ is the probability of sampling a particular distance when building the ring (for the surface ring, we have $\mathrm{pdf}(r_s) = 1$). L_j is the radiance contributed by each segment $\ell_j \in \mathcal{L}_i$, defined by the integral:

$$L_j(\mathbf{x}, \vec{\omega}_o) = \int_{\ell_j} f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)\, G(\mathbf{x}, \mathbf{y})\, T_r(\mathbf{x}, \mathbf{y})\, L(\mathbf{y}, \vec{\omega}_i)\, \mathrm{d}\mathbf{y}. \tag{2.7}$$

[2]For convenience, we formulate all the equations in terms of 2D media and geometry subdivisions in segments ℓ, but all formulae are equally applicable in 3D by substituting segments ℓ by triangles $\triangle$.

By construction, the visibility between $\mathbf{x}$ and $\mathbf{y}$ is $V(\mathbf{x},\mathbf{y}) = 1$, and $\mathbf{y}$ is a point on a *virtual* surface; we thus need to account for the foreshortening at $\mathbf{y}$. This allows for a unified formulation of both surface-to-medium and medium-to-medium radiance derivatives, using the same geometry term in both cases. Note that we have merged together the phase function $f_s(\mathbf{x},\vec{\omega}_i,\vec{\omega}_o)$ and scattering coefficient $\mu_s(\mathbf{x})$ as a directional scattering function $f(\mathbf{x},\vec{\omega}_i,\vec{\omega}_o) = \mu_s(\mathbf{x}) f_s(\mathbf{x},\vec{\omega}_i,\vec{\omega}_o)$, to make the following derivations simpler.

Differentiating Equation (2.6) with respect to $\mathbf{x}$ provides approximations for the first and second order derivatives:

$$\nabla L(\mathbf{x},\vec{\omega}_o) \approx \sum_{r_i\in\mathcal{R}} \sum_{\ell_j\in\mathcal{L}_i} \frac{\nabla L_j(\mathbf{x},\vec{\omega}_o)}{\mathrm{pdf}(r_i)}, \tag{2.8}$$

$$\mathbf{H}L(\mathbf{x},\vec{\omega}_o) \approx \sum_{r_i\in\mathcal{R}} \sum_{\ell_j\in\mathcal{L}_i} \frac{\mathbf{H}L_j(\mathbf{x},\vec{\omega}_o)}{\mathrm{pdf}(r_i)}, \tag{2.9}$$

which in turn require differentiating the radiance from each segment.

Unfortunately, we cannot compute Equation (2.7) and its derivatives analytically in closed-form, while computing it numerically would be prohibitively expensive. We instead introduce a set of assumptions to build a closed-form approximation:

- For a sufficiently fine subdivision the angle γ tends to 0, so $\vec{\omega}_i$ can be regarded as constant for the whole segment, and $f(\mathbf{x},\vec{\omega}_i,\vec{\omega}_o) = f(\mathbf{x},\vec{\omega}_\ell,\vec{\omega}_o)$, with $\vec{\omega}_\ell$ a fixed direction from $\mathbf{x}$ to a point in segment ℓ.

- For all $\mathbf{y} \in \ell$, we assume constant $T_r(\mathbf{x},\mathbf{y}) = T_r(\mathbf{x},\mathbf{y}_\ell)$, and $L(\mathbf{y},\vec{\omega}_i) = L(\mathbf{y}_\ell,\vec{\omega}_i)$. Following existing approaches for surface irradiance, we choose $\mathbf{y}_\ell$ as the furthest point in the segment ℓ, which will be the first to be occluded/unoccluded.

These assumptions allow us to significantly simplify the integral in Equation (2.7) to:

$$\begin{aligned} L_j(\mathbf{x},\vec{\omega}_o) &\approx f(\mathbf{x},\vec{\omega}_{\ell_j},\vec{\omega}_o)\, T_r(\mathbf{x},\mathbf{y}_{\ell_j})\, L(\mathbf{y}_{\ell_j},\vec{\omega}_i) \int_{\ell_j} G(\mathbf{x},\mathbf{y})\, \mathrm{d}\mathbf{y} \\ &= f(\mathbf{x},\vec{\omega}_{\ell_j},\vec{\omega}_o)\, T_r(\mathbf{x},\mathbf{y}_{\ell_j})\, L(\mathbf{y}_{\ell_j},\vec{\omega}_i)\, F_{\ell_j}(\mathbf{x}), \end{aligned} \tag{2.10}$$

which now admits a closed-form solution in both 2D and 3D (see Appendices 2.B and 2.C). More importantly, this allows us to approximate the derivatives of L_j in closed form as:

$$\nabla L_j \approx L_F \nabla f + \nabla L_F f, \tag{2.11}$$

$$\mathbf{H}L_j \approx L_F \mathbf{H}f + \nabla L_F \nabla^\mathsf{T} f + \nabla f \nabla^\mathsf{T} L_F + \mathbf{H}L_F f, \tag{2.12}$$

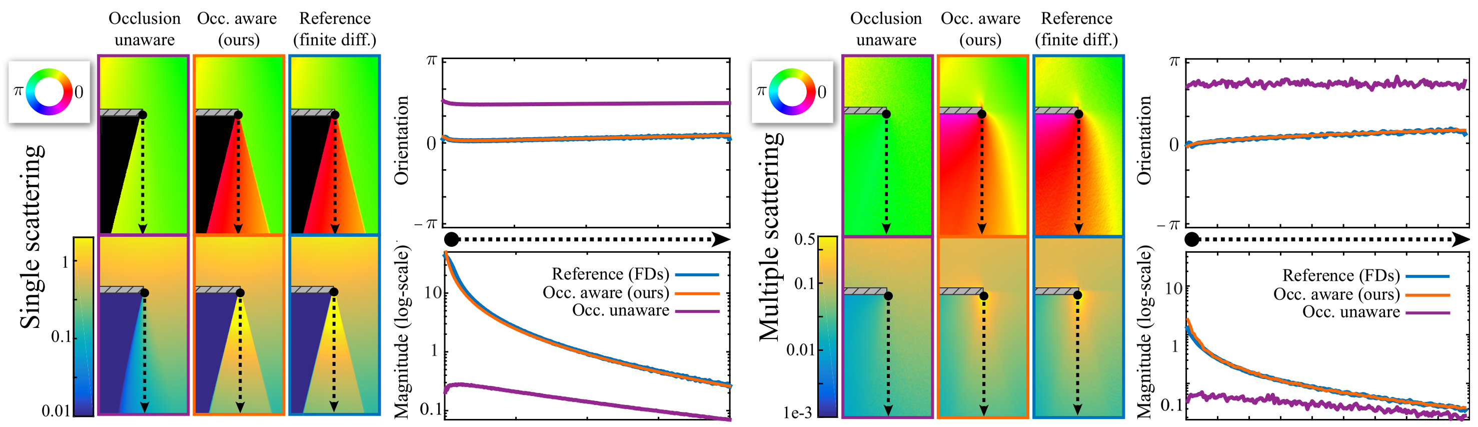


Figure 2.4: 2D single and multiple scattering gradients in similar setup to Figure 2.2, top. Compared against an occlusion-unaware reference solution [116], our method correctly captures both gradients orientation (color-coded angle), and magnitude. The graphs show the evolution of the gradient across the dotted black line for both methods (purple, orange), and the reference solution (blue).

where

$$\nabla L_F = L_r \nabla F_\ell + \nabla L_r F_\ell, \tag{2.13}$$
$$\mathbf{H} L_F = L_r \mathbf{H} F_\ell + \nabla L_r \nabla^\intercal F_\ell + \nabla F_\ell \nabla^\intercal L_r + \mathbf{H} L_r F_\ell, \tag{2.14}$$
$$\nabla L_r = L \nabla T_r + \nabla L T_r, \tag{2.15}$$
$$\mathbf{H} L_r = L \mathbf{H} T_r + \nabla L \nabla^\intercal T_r + \nabla T_r \nabla^\intercal L + \mathbf{H} L T_r. \tag{2.16}$$

For brevity we have omitted function parameters, and we express gradients and Hessians in terms of the scaled radiance $L_F = F_\ell L_r$, and the reduced radiance $L_r = L T_r$. While Equations (2.10–2.16) are general, we restrict our work to Lambertian surfaces and isotropic, homogeneous media (in Section 2.7 we discuss how to extend it to anisotropic and heterogeneous media). This means that both L and f are constant, and therefore their derivatives cancel out as $\nabla L = \mathbf{H} L = \nabla f = \mathbf{H} f = 0$, removing directional dependences; this allows us to simplify Equations (2.11) and (2.12) to:

$$\nabla L_j \approx L f \left(T_r \nabla F_\ell + \nabla T_r F_\ell \right), \tag{2.17}$$
$$\mathbf{H} L_j \approx L f \left(T_r \mathbf{H} F_\ell + \nabla T_r \nabla^\intercal F_\ell + \nabla F_\ell \nabla^\intercal T_r + \mathbf{H} T_r F_\ell \right). \tag{2.18}$$

We refer to Appendices 2.A, 2.B and 2.C for all the terms.

By construction, our formulation in Equation (2.6) and its derivatives (Equations (2.8) and (2.9)) are biased but consistent estimators of $L(\mathbf{x}, \vec{\omega}_o)$, $\nabla L(\mathbf{x}, \vec{\omega}_o)$, and $\mathbf{H} L(\mathbf{x}, \vec{\omega}_o)$, respectively. In addition the assumptions imposed in Equation (2.10) introduce some additional bias due to the piecewise assumption in the scattering f, transmittance T_r, and radiance terms L. However, as shown in Figure 2.2 our formulation converges accurately to the actual derivatives. Note that we use this biased but consistent approximation only to compute first- and second-order derivatives of media radiance (Equations (2.8) and (2.9)), while computing actual radiance values (Equation (2.1)) using the standard unbiased Monte Carlo estimator. In the following, we describe how to use the derivatives in Equations (2.8) and (2.9) for interpolating radiance from a set of cache points, and define an error metric for such interpolation.

2.5 SECOND-ORDER ERROR CONTROL FOR MEDIA RADIANCE EXTRAPOLATION

The error in radiance caching is controlled by a tolerance value ε, and depends both on how radiance is extrapolated, and on the radiance moments at cache point $\mathbf{x}$. These moments define a valid bounding region $\aleph$ where a point $\mathbf{x}'$ can be used for extrapolation. We provide here the key ideas and resulting equations for the valid regions in the context of 2D and 3D participating media and provide detailed derivations in the supplementary material[3].

Existing work on radiance caching for participating media estimates the relative error using radiance gradients at $\mathbf{x}$. However, ignoring higher-order derivatives creates suboptimal cache distributions that often oversample regions near surfaces and light sources. Given the radiance and the first n derivatives at a media point $\mathbf{x}$, we can approximate radiance at point $\mathbf{x}' \in \aleph$ using an n^{th}-order Taylor expansion. Following previous work [230] we truncate to order one, approximating $L(\mathbf{x}', \vec{\omega}_o)$ as:

$$L(\mathbf{x}', \vec{\omega}_o) \approx L(\mathbf{x}, \vec{\omega}_o) + \nabla L(\mathbf{x}, \vec{\omega}_o)\Delta_{\mathbf{x}'}. \tag{2.19}$$

Since we focus on isotropic media, we remove the directional dependence in the following derivations to simplify notation. By using a second order expansion of $L(\mathbf{x})$ as our oracle, we can approximate the relative error $\hat{\epsilon}'(\mathbf{x}')$ of the extrapolation as:

$$\hat{\epsilon}'(\mathbf{x}') \approx \frac{|\Delta_{\mathbf{x}'}^{\mathsf{T}} \mathbf{H}L(\mathbf{x})\Delta_{\mathbf{x}'}|}{2\,L(\mathbf{x})}, \tag{2.20}$$

with $\mathbf{H}L(\mathbf{x})$ the Hessian matrix of $L(\mathbf{x})$. This expression is similar to the second-order error metric proposed by Jarosz et al. [120] and follow-up work by Schwarzhaupt et al. [230], although these works dealt with surfaces only.

By integrating Equation (2.20) in the neighborhood of $\mathbf{x}$ for a given error threshold ε, we can express the valid region in two-dimensional media as an ellipse with principal radii $R_{2\text{D}}^{\lambda_i}$ (see the supplemental document[3] for the complete derivation)

$$R_{2\text{D}}^{\lambda_i} = \sqrt[4]{\frac{4L(\mathbf{x})\varepsilon}{\pi|\lambda_i|}}, \tag{2.21}$$

where λ_i is the i-th eigenvalue of the radiance Hessian $\mathbf{H}L(\mathbf{x})$. This formula is analogous to the relative error metric presented by Schwarzhaupt and colleagues [230] for surfaces, but here the radii are computed by taking the principal components of the *volumetric* radiance Hessian. Adding the third dimension, the valid region for a cache point becomes a 3D ellipsoid, whose principal radii are:

$$R_{3\text{D}}^{\lambda_i} = \sqrt[5]{\frac{15L(\mathbf{x})\varepsilon}{4\pi|\lambda_i|}}. \tag{2.22}$$

Our second-order error metric and its derived radius assume knowledge of the radiance and its derivatives at $\mathbf{x}$. In practice, these are

[3] http://webdiis.unizar.es/~juliom/pubs/2018TOG-SecondOrderMediaCaching/2018TOG_SecondOrderMediaCaching_supp.pdf

usually computed by Monte Carlo techniques, which lead to other sources of error such as variance (inherent to Monte Carlo sampling), or bias (due to inaccuracies computing the derivatives).

The presented metric describes the error introduced by extrapolation from a single cache point in participating media. However, at render time, we compute radiance at each shaded point by interpolating from *multiple* cached points, as:

$$L(\mathbf{x}') \approx \frac{\sum_{k \in C} \left[L(\mathbf{x}_k) + \nabla L(\mathbf{x}_k) \cdot \Delta_{\mathbf{x}'} \right] \mathrm{w}(\mathbf{x}_k, \mathbf{x}')}{\sum_{k \in C} \mathrm{w}(\mathbf{x}_k, \mathbf{x}')}, \tag{2.23}$$

with C the set of cache points whose radii include $\mathbf{x}'$, and $\mathrm{w}(\mathbf{x}_k, \mathbf{x}')$ the interpolation kernel. Following Jarosz et al. [116], we use a cubic interpolation kernel $\mathrm{w}(\mathbf{x}_k, \mathbf{x}') = 3d^2 - 2d^3$ with $d = 1 - \|\mathbf{x}' - \mathbf{x}_k\| \, R_k^{-1}$. Since Equation (2.23) only interpolates from cache points which predict a maximum error $\hat{\epsilon}' < \epsilon$ at $\mathbf{x}'$, the error of the weighted sum is equally upper-bounded by ϵ. Note that, as opposed to Jarosz et al. [116] (2.4), we interpolate in linear space, where the error is more accurately predicted by our Hessian-based metric described in Equation (2.20).

2.6 RESULTS

In the following we illustrate the accuracy and benefits of our method. We start showing our results in a two-dimensional world, and compare it against a 2D version of the current state-of-the-art method [116]. We refer the reader to the supplementary material for the additional expressions to compute two-dimensional occlusion-unaware gradients. Then, we move to 3D, to demonstrate that our results are also consistent in a more practical three-dimensional scenario. For comparison purposes, all 3D insets show only single and multiple scattering in media, discarding surface radiance. Unless it is explicitly mentioned, we use isotropic points with the smallest principal axis of the Hessian. This is the most costly scenario for our method in comparison to previous work, since we cannot adapt to the signal as faithfully as with anisotropic points, and therefore require more points.

Implementation: We compute both radiance and derivatives at point $\mathbf{x}$ by stratified sampling uniformly in the sphere, with equal solid angle strata (in the case of 2D, this stratification is in the circle, using equal angle stratification). This reduces variance compared to pure uniform sampling. More importantly, it allows to very simply build the subdivision using the angular samples, by just connecting samples from adjacent strata [230]. This stratification is used for both media and surfaces, including area light sources, while other direct light sources (such as directional or point lights) must be handled separately. The accuracy of the subdivision for computing the derivatives relies on a dense sampling of the angular domain, and, as in any sampling problem, our sampling rate limits the amount of radiance changes that we can recover. This is especially important when capturing fine details such as small light sources, which are not computed using next event estimation (NEE), but could also be important in high-frequency fluctuations of radiance in media. However, in practice our method presents much better convergence than previous work [116] with increasing number of angular samples, as shown in Figure 2.2. Introducing a NEE-aware subdivision combined with the standard angular one via multiple importance sampling could significantly improve the performance of the derivative computation, although we leave this to future work. We perform the subdivision within the medium by uniformly ray-marching the medium at discrete distances around $\mathbf{x}$, and joining adjacent angular samples within each marching step (see Figure 2.1e).

Unless stated otherwise, single scattering in all compared methods refers to radiance emitted or reflected (first bounce) by surfaces. We limit multiple scattering to the second bounce for all methods. We did this mainly to reduce excessive variance when computing reference derivatives with finite differences. Note that both occlusion-unaware and occlusion-aware methods are equally applicable to higher number of media bounces, although they usually require a high number of samples to obtain noiseless solutions.

Following previous methods [116], we first pre-populate the cache by uniformly sampling a ray from the camera, and ray-marching along the media, placing cache points in case they do not fulfill our error metric (Section 2.5). At render time, we evaluate Equation (2.23) at ray-marched points $\mathbf{x}'$ in the medium, extrapolating radiance from the surrounding valid cache points. If no valid cache points are found for $\mathbf{x}'$ then we compute its radiance and derivatives, and add it to the cache. As in previous methods [116], we separate single and multiple scattering caches, each in a different octree for efficient cache query.

All results were computed on a desktop PC with an Intel Core i7 3.4 GHz CPU and 16GB RAM. Note that all methods used for rendering comparisons of the complex 3D scenes *Whiteroom* and *Staircase* were accelerated with Embree ray-tracing kernels [260], and therefore the performance with respect to the other 3D scenes is higher.

2.6.1 *Results in 2D*

To evaluate the error introduced by our occlusion-aware computations of derivatives in a clear, intuitive way, we rely on their two-dimensional versions. In Figure 2.2 we showed the convergence of gradient computation with the number of angular samples. Previous approaches not taking into account visibility changes fail to estimate the gradient. In contrast, our derivative formulation converges to the actual gradient, even in areas of penumbra for both single and multiple scattering. The

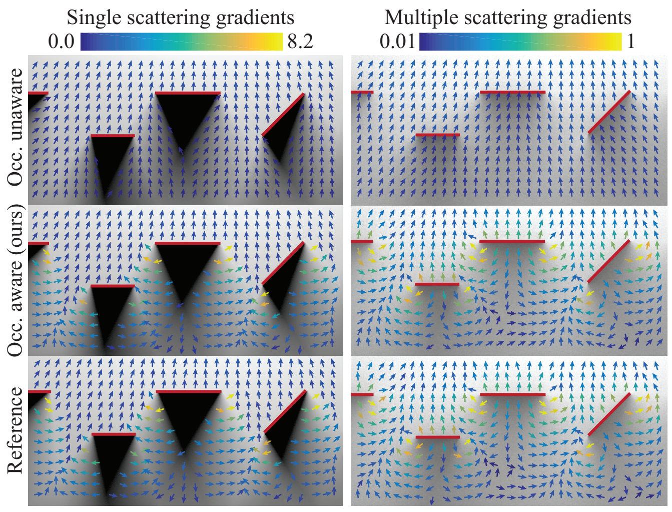


Figure 2.5: Radiance gradients at discrete locations in 2D computed with occlusion unaware, and our occlusion aware methods, compared against reference gradients (bottom row) computed with path traced finite differences using 4M samples/gradient. Left column, top and middle, show single scattering gradients computed with 256 angular samples/gradient. Right column, top and middle, show multiple scattering gradients computed with 65536 samples/gradient (256 angular $\times$ 256 ray samples).

quality of our estimated derivatives increases with the number of angular samples, since the approximations introduced by our assumptions vanish as the strata size diminishes.

In Figure 2.4 we compare the evolution of single and multiple scattering gradients across a penumbra region, computed with our method and previous work. We illustrate them in polar coordinates (magnitude and orientation) in a simple scene with a medium illuminated by an area light on top, and a line acting as an occluder within the medium. We compute reference gradients with path traced finite differences. Our approach manages to correctly compute both gradients magnitude and orientation in the penumbra region. The right graphs show a progression of gradients along the dotted line. The graphs show that our method is able to match the ground-truth, while the occlusion-unaware method both underestimates the magnitude of the gradient and computes an incorrect direction.

Figure 2.5 shows a comparison of gradients (shown as a vector field) with the occlusion-unaware method, our technique, and a ground-truth solution computed with finite differences. Our method correctly captures complex radiance changes, including strong changes near occluder boundaries, closely matching the ground-truth reference.

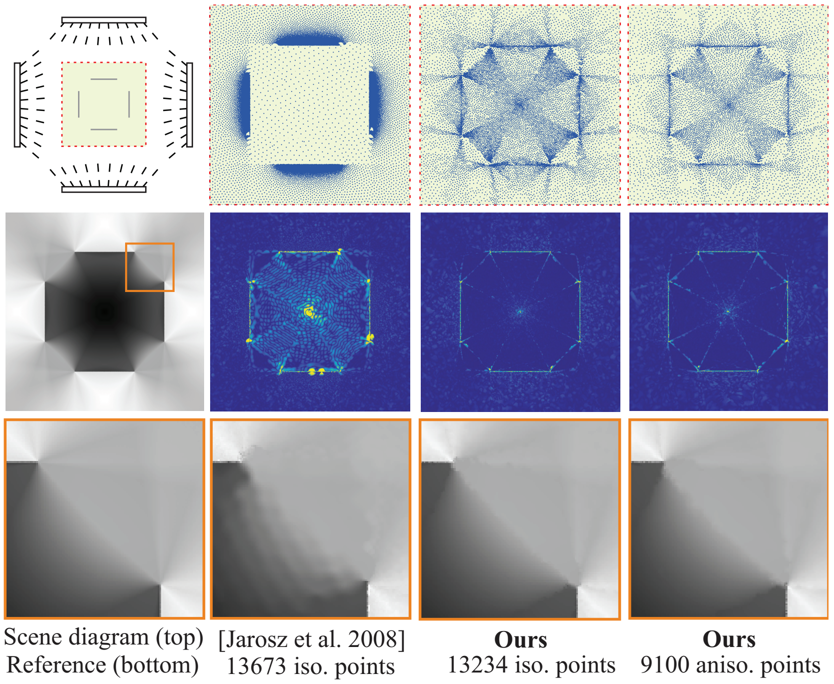


Figure 2.6: Single scattering in a 2D setup with four line lights and four occluders. Point distributions (top row) show how a occlusion-unaware gradient metric [116] fails to estimate the correct radiance changes in complex shadows, while tending to concentrate cache points near reflecting geometry. In contrast, our algorithm distributes points according to occlusion-aware, second-order derivatives of radiance, capturing complex light patterns more accurately. Leveraging curvature information in the Hessians enables anisotropic cache points that further reduce the number of required cache points while maintaining quality (see error maps).

Our error metric takes into account second-order derivatives to drive sample-point density in the scene. Since we use the estimated occlusion-aware Hessians as an oracle of the error, this allows us to place more cache points in areas with higher frequencies. Additionally, our improved gradients allow for a more accurate extrapolation within the valid region of the cache points. Figure 2.6 (top) shows a scene with overlapping shadows, created by four lights and four occluders (top-left diagram indicates the shaded region in green). Previous work (second column) drives point density based on the log-space gradient of radiance; in practice this tends to drastically increase point density near light-reflecting geometry, failing to efficiently sample shadowed regions. This can only be mitigated by radius-clamping heuristics (in this case based on the pixel size), thus breaking the principled properties of the

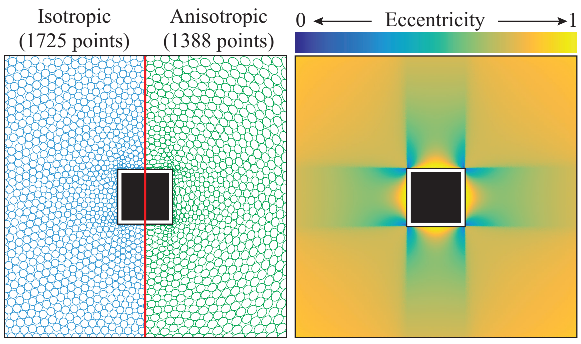


Figure 2.7: 2D point distributions in a medium illuminated by a square-shaped light, using our Hessian-based error metric with isotropic and anisotropic points (left) using the same relative error threshold. Eccentricity of radiance curvature (right) determines anisotropy of cache points (left, green), stretching circular points to ellipses along the direction of lower change.

approach. In contrast, our method (last two columns) does not rely on heuristics and manages to correctly capture shadows by placing more points near shadow boundaries.

By computing principal components of radiance Hessians, we can use the *radiance eccentricity* (i.e. the eccentricity of the ellipse defined by the Hessian of the radiance) to stretch media cache points along the components with lower radiance variation, obtaining elliptic (2D) or ellipsoidal (3D) cache points. In Figure 2.6 (bottom) we compare previous work with our isotropic and anisotropic cache distributions. Even with a similar number of isotropic points ($\sim$13k), our improved derivatives manage to capture the overlapping shadows much better; using our anisotropic technique, we manage to reduce cache size by 32%, while keeping the same error threshold. Figure 2.7 illustrates eccentricity across a 2D scene with a square light emitter in the center. By keeping the same error threshold, our anisotropic cache reduces the number of cache points by up to 20%.

2.6.2 *Results in 3D*

Here we further analyze occlusion-unaware gradients and our occlusion-aware Hessians on four 3D scenes: *Strips*, *Statues*, *Patio* and *Cornell holes*. Unless stated otherwise, all renders are taken using 16 samples per pixel, and performing uniform ray marching with a step size of 0.1.

The *Statues* scene shown in Figure 2.8 combines both surface-to-media single scattering, and media-to-media (two-bounce) multiple scattering. The scene includes distant and local light sources (side windows and ceiling, respectively). Occlusion-unaware single and mul-

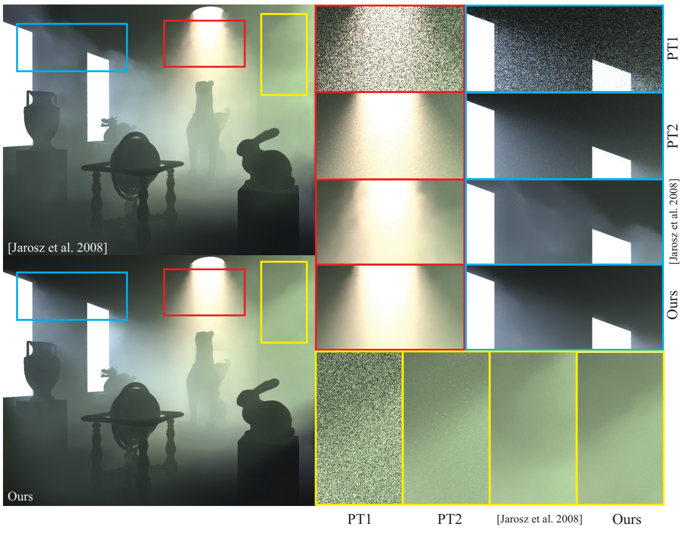


Figure 2.8: *Statues* scene rendered with both single and multiple scattering. Radiance at surfaces is excluded for illustration purposes (please refer to the digital version for accurate visualization). **PT1:** Path tracing, 2k samples/pixel, 2 hours. **PT2:** Path tracing, 500k samples/pixel, 500 hours. **[116]:** Occlusion-unaware, gradient-based error metric, ~19k cache points, 16k samples/cache, 155 minutes. **Ours:** Occlusion-aware, Hessian-based metric, ~19k cache points, 16k samples/cache, 154 minutes. Ignoring visibility derivatives fails at representing high-frequency shadows from the windows (a, blue and yellow) due to poor cache distribution, as well as other rapid radiance changes (a, red) in areas with good cache distribution, due to imprecise extrapolation during reconstruction. In contrast, our occlusion-aware Hessian-based method correctly handles these higher-frequency features by improving the sample distribution, as well as the reconstruction.

tiple scattering gradients lead to big splotches on the boundaries of light beams coming through the windows. In the case of light coming through the ceiling, while the point distribution captures shadow contours fairly well, extrapolation fails since occlusion-unaware gradients ignore light effects produced in the penumbra region. Moreover, occlusion-unaware techniques concentrate most cache points near light sources and reflecting surfaces (Figure 2.9, top-left), as seen previously in 2D. Since the gradients are large in these areas, this results in very small valid radii for the cache points. Histograms show how for previous work nearly 8000 points (Figure 2.9, top-right, leftmost bin of the blue histogram) on single scattering reach the minimum radius, which is close to a 40% of the total number of points. This implies that

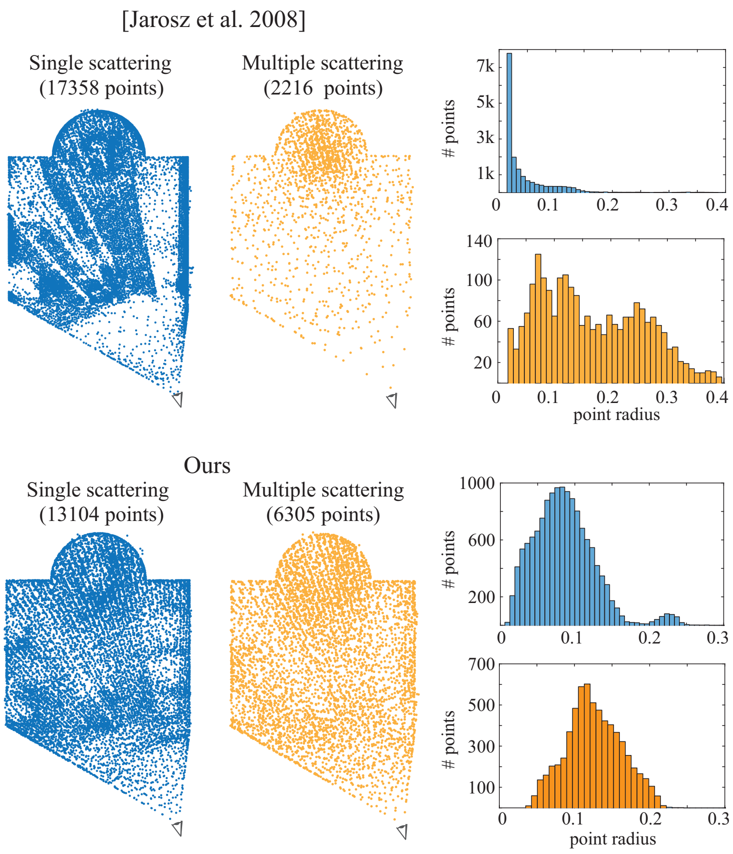


Figure 2.9: Cached point distributions of the results in Figure 2.8 as seen from above for both single and multiple scattering. The occlusion-unaware approach (top) concentrates the samples excessively near the surfaces, usually reaching the cache minimum radius (see top-right histograms), but ignoring occlusion changes throughout the scene. Using occlusion-aware first- and second-order derivatives, our method predicts the error introduced by gradient extrapolation more robustly, increasing cache density in regions where gradients change rapidly (bottom).

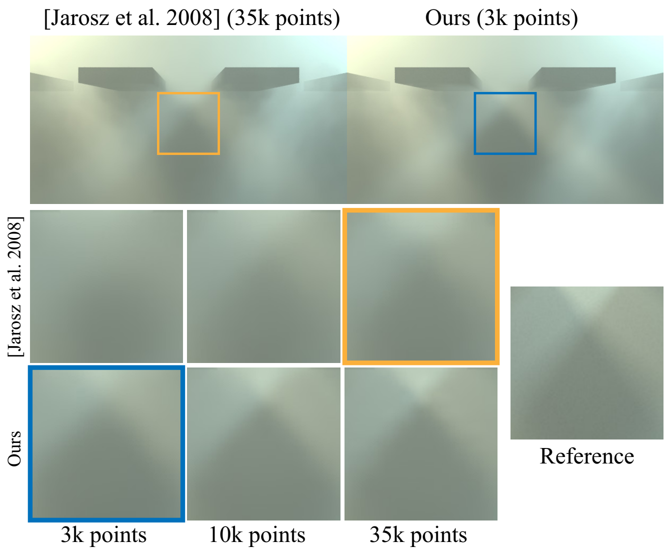


Figure 2.10: *Strips* scene comparing single scattering for an increasing number of cache points, and showing relative error with respect to the reference image. While occlusion-unaware gradients method requires 35k cache points to fairly capture occlusions, our occlusion-aware Hessians produce similar results with just 3k points.

the performance of this approach is highly dependent on the value of such minimum radius, which undermines the principled basis of its error metric. In contrast our method generates better point distributions, which correctly capture light gradients while avoiding additional heuristics to control oversampling in certain regions.

The *Strips* scene (Figure 2.10) shows surface-to-medium single scattering, for an increasing number of cache sizes. Surface radiance is excluded for illustration purposes. The occlusion-unaware method needs an order of magnitude more cache points to get comparable results to ours (see progression insets). This implies that we have to significantly drop the tolerance parameter to create sufficiently fine point distributions in occluded regions. As we can observe in Figure 2.10, top row, our method yields better sampling density and extrapolation from the sampled points, achieving similar results with an order of magnitude less points.

Computing derivatives of surface-to-medium form factor involves operating with 3×3 matrices (see Appendix 2.C). Including the cost of scene subdivision, this introduces an overhead per cache point of just 9%, compared to computing only point-to-point first derivatives (see Table 2.2 for the *Patio* scene). Nevertheless, as we can see in Figure 2.11, our method yields better equal-time results with isotropic points. Moreover, our anisotropic approach stretching spherical cache

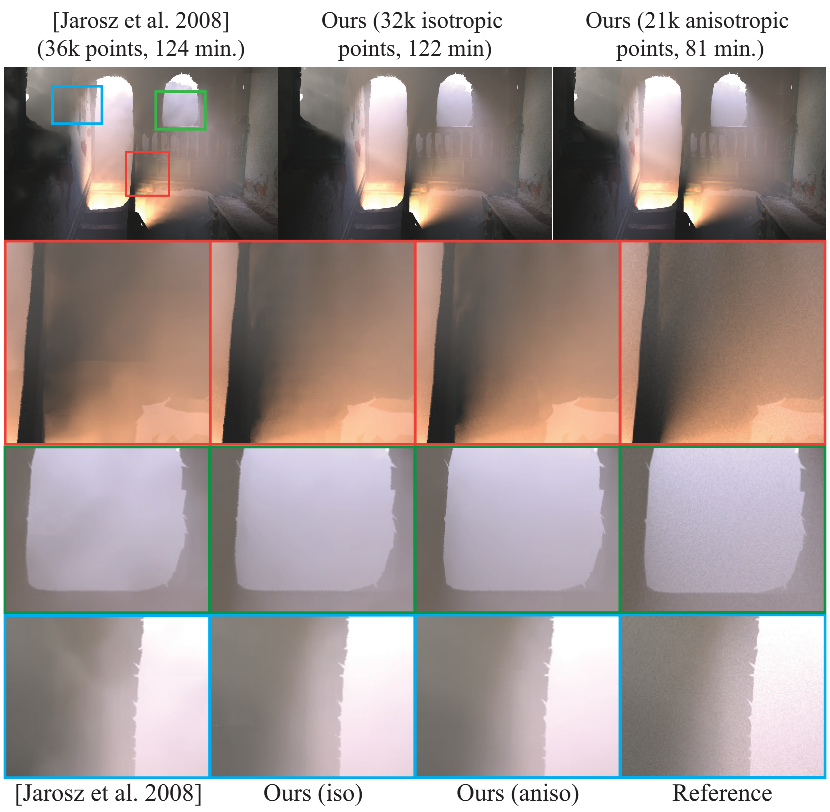

[Jarosz et al. 2008] (36k points, 124 min.)
Ours (32k isotropic points, 122 min)
Ours (21k anisotropic points, 81 min.)
[Jarosz et al. 2008]
Ours (iso)
Ours (aniso)
Reference

Figure 2.11: *Patio* scene with single scattering. Our method outperforms existing occlusion-unaware techniques on an equal-time comparison. Moreover, our anisotropic cache manages to significantly reduce total time under the same error tolerance $\varepsilon = 1.5\text{e}{-4}$ than our isotropic cache, while still retaining shadow details on window boundaries and near thin handrails as shown in the insets.

points along the principal components of radiance, allows to reduce both the number of points and the total computation time by 30% for the same error tolerance.

The *Cornell Holes* scene (Figure 2.12) shows how our method successfully resolves difficult, high-frequency occlusions due to light coming out of the box. Our method provides a built-in mechanism to significantly reduce error in two ways: additional samples reduce variance but also create finer subdivisions, thus improving accuracy when detecting occlusions.

We also demonstrate the benefits our method in scenes of higher complexity. In the *Staircase* scene (Figure 2.13) we show an equal-time

[4]Note that Jarosz et al.'s metric (Equation (2.5)) is different from our Hessian-based integrated error ε, thus tolerance values of both metrics have different meaning.

Table 2.2: Computation data for the *Patio* scene. For the isotropic case, our method yields better results in equal time. Using anisotropic points provides a further 30% computation time reduction at the same low error threshold due to the improved point distributions and larger valid regions.

Method	**Error tol.** [4]	**Cache gen.**	**Time / point**	**Total time**
Jarosz et al. 2008	0.3	124 min / 36k pts	206 ms	136 min
Ours (isotropic)	$\varepsilon = 1.5e{-}4$	122 min / 32k pts	225 ms	135 min
Ours (anisotropic)	$\varepsilon = 1.5e{-}4$	81 min / 21k pts	225 ms	94 min

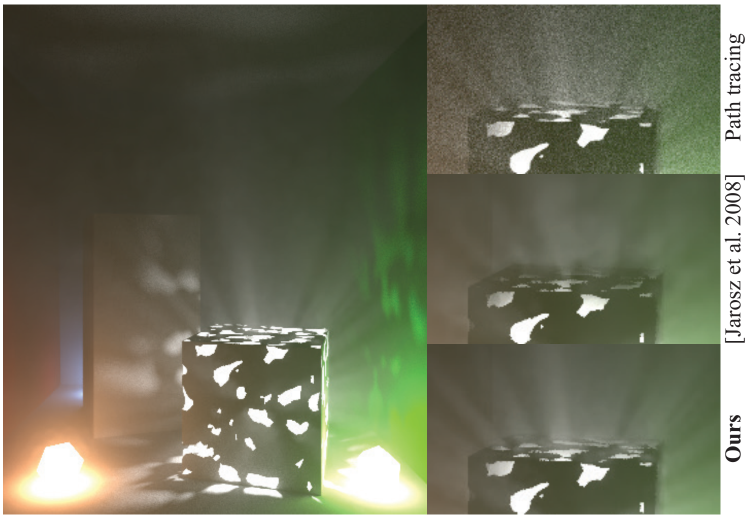


Figure 2.12: *Cornell Holes* scene showing complex high-frequency shadows handled by our method, compared against equal-time path traced, and occlusion-unaware gradients solutions.

comparison with a render time of 90 minutes. Path tracing has not fully converged to the reference solution in that time, and while the point distributions of occlusion-unaware methods manage to capture the main shadow boundaries, occlusion-unaware gradients still create visible artifacts on the shadow patterns created by light coming from different windows. Progressive photon beams [119] manages to capture high frequency changes, but fails to densely sample the medium due to distant lighting. In equal time, our method manages to get the closest match to the reference by correctly capturing complex shadow configurations. In Figure 2.14 we also illustrate convergence of our occlusion-aware gradients in the same scene by analyzing the changes on a XZ-aligned slice of the media crossing through the light shafts. We compare our gradients against finite differences gradients on two

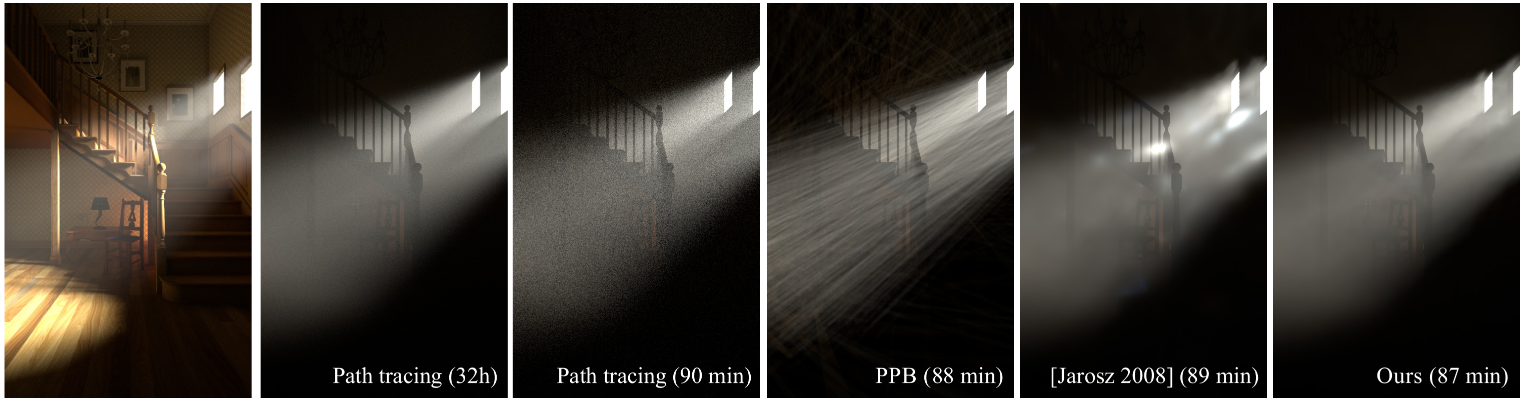


Figure 2.13: *Staircase* scene showing equal-time comparisons of path tracing, progressive photon beams (PPB), occlusion-unaware gradients [116], and our second-order occlusion-aware solution. We include a fully converged solution for path tracing. Each cache in both our method and Jarosz et al. is computed using 16k stratified angular samples, and rendered using 16 samples per pixel. The progressive photon beams solution was obtained using the publicly available Tungsten rendering engine [19]. Note how the occlusion unaware method creates visible artifacts in the patterns created by the shadows crossing from different windows, while our method correctly captures those details in equal time. Due to distant lighting, progressive photon beams fails to densely sample the light shafts coming through the windows, resulting in visible variance after 400 iterations of 1M beams/iteration.

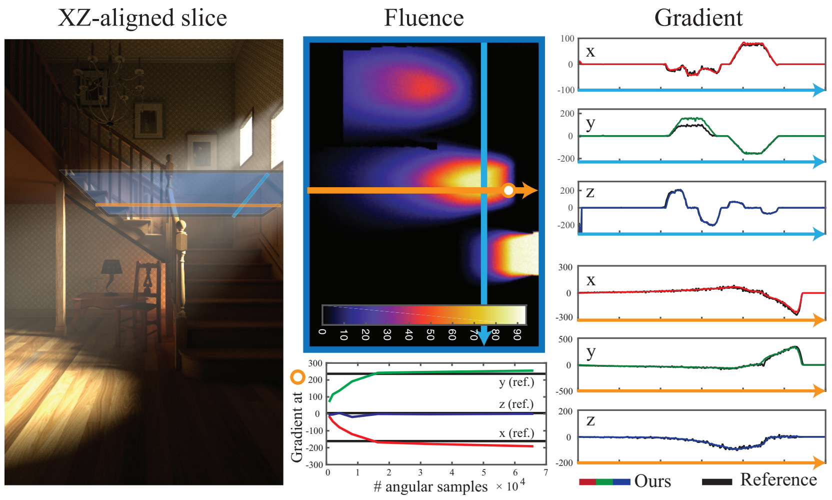


Figure 2.14: We demonstrate the convergence of our occlusion aware derivatives in complex 3D scenarios like *Staircase*. We illustrate this using an XZ-aligned slice of the media that captures the occlusion changes produced by the light shafts through the windows. Right graphs show our computed gradients across two orthogonal scan lines of the slice, where we can observe how our method matches the reference derivatives computed with finite differences. In the bottom graph we also illustrate convergence at the white dot respect to the number of angular samples. Higher number of angular samples create finer scene subdivisions and increase the precision of our derivatives, which provide a very good estimation of the actual derivatives.

orthogonal scanlines that cross through the shadows, and demonstrate how our method converges to the reference gradients by creating finer subdivisions with higher number of angular samples.

Finally we perform comparisons up to equal-quality in the *Whiteroom* scene (Figure 2.15), which presents high scattering due to bright white walls and furniture. In a sequence of insets with increasing render time, we show how our method manages to recover high-frequency shadows in much less time than other methods, which also fail to capture thin shadows near window boundaries.

2.7 CONCLUSIONS

We have presented a new occlusion-aware method for efficiently computing light transport in homogeneous isotropic media, including both single and multiple scattering. At the core of our method lies an efficient computation of radiance derivatives for both surface-to-medium and medium-to-medium light transport. Our radiance derivatives, in-

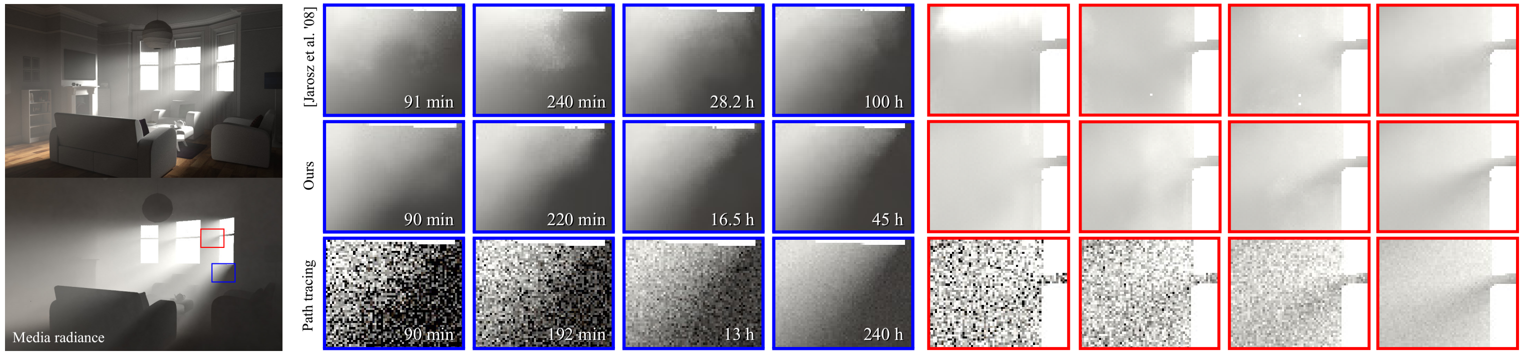


Figure 2.15: We illustrate convergence to an equal-quality reference solution with our algorithm, Jarosz et al. method, and path tracing for the *Whiteroom* scene. In 3D scenes of higher complexity, our method presents much better convergence properties than previous ones, being able to reconstruct the shadow boundaries near the window frames in much less time.

cluding visibility changes for single and multiple scattering, improve both the placement of cache points, as well as their interpolation using a Taylor expansion.

We have additionally formalized light transport in participating media in a self-contained 2D world; we hope that this framework becomes a valuable contribution for the graphics community as a testbed for novel algorithms. Our results (2D and 3D) demonstrate a significant improvement over the current state of the art, both in equal-time and equal-error comparisons.

LIMITATIONS & FUTURE WORK Our work shares some of the limitations of traditional radiance caching algorithms, namely the assumption of relatively low frequency transport with finite derivatives. High-frequency illumination due to e.g. small light sources would require a very fine-grained subdivision to accurately find shadow boundaries. Other high-frequency effects such as caustics would additionally require departing from the assumption of constant angular radiance L_o in Equation (2.10), which would in turn require computing its translational derivatives.

In our implementation we have assumed isotropic media, which helps reduce the complexity and storage requirements of the cache points. By using an angularly-resolved caching of radiance and its derivatives (by using e.g. spherical harmonics [116, 154]) anisotropic phase functions could be added. Incorporating heterogeneous media would break the assumption of constant scattering term (i.e. $\nabla f \neq \mathbf{H} f \neq 0$) given the variability of μ_s and f_s within the media. This would require us to use the full radiance derivatives (Equations (2.11) and (2.12)), instead of the simplified Equations (2.17) and (2.18). Moreover, it would require changing our derivatives of transmittance T_r; given our marching procedure for subdividing the media, a similar approach to Jarosz et al.'s [116] for single scattering could be used. Finally, high-frequency heterogeneity in the medium would require a very fine subdivision, which would potentially make our approach impractical.

Our error metric assumes that the error is due to extrapolation only, with perfect radiance samples and derivatives. However, both are computed stochastically, which introduces variance (in the case of radiance), and bias (on the derivatives). Developing new metrics taking into account these additional sources of error, as well as accurately characterizing them, are interesting avenues of future work. In this regard, analyzing other consistent approaches to compute derivatives (e.g. using photon mapping [137]) might be helpful. Evaluating whether using our biased estimator of radiance (Equation (2.6)) instead of our Monte Carlo estimate of Equation (2.1) would be interesting too, making our cache points more robust by reducing variance (at the price of additional bias). Finally, it may be possible to use our first- and second-order derivatives to accurately estimate the optimal kernel in density estimation algorithms for participating media [86], as well as to guide sampling in media or to improve quadrature-based ray-marching methods [191].

APPENDICES

In the following we summarize 2D and 3D expressions of translational derivatives of transmittance and form factors needed for our method. We box all relevant final expressions that to the best of our knowledge are new to the literature. We define column vectors as $\vec{\mathbf{v}}$ and row vectors as $\vec{\mathbf{v}}^\intercal$. Expressions such as $\vec{\mathbf{r}}_1 \cdot \vec{\mathbf{r}}_2$ denote dot (inner) products, while expressions such as $\vec{\mathbf{r}}_1 \vec{\mathbf{r}}_2^\intercal$, $\nabla(\ldots)\nabla^\intercal(\ldots)$, and $(\ldots)(\ldots)^\intercal$ denote vector *outer* products.

2.A HOMOGENEOUS TRANSMITTANCE DERIVATIVES

Homogeneous transmittance is modeled by the exponential decay due to extinction,

$$T_r = e^{-\mu_t \|\overrightarrow{\mathbf{yx}}\|} \tag{2.24}$$

where $\|\overrightarrow{\mathbf{xy}}\|$ denotes distance between source $\mathbf{y}$ and shaded point $\mathbf{x}$. Its gradient and Hessian with respect to a translation of $\mathbf{x}$ are

$$\nabla T_r = -\mu_t \frac{\overrightarrow{\mathbf{yx}}}{r} T_r, \tag{2.25}$$

$$\boxed{\mathbf{H} T_r = -\mu_t \left(\frac{\mathbf{J}(\overrightarrow{\mathbf{yx}})}{r} - \frac{1}{r^3} \overrightarrow{\mathbf{yx}}\,\overrightarrow{\mathbf{yx}}^\intercal - \frac{\mu_t}{r^2} \overrightarrow{\mathbf{yx}}\,\overrightarrow{\mathbf{yx}}^\intercal \right) T_r. \qquad (2.26)}$$

2.B 2D SEGMENT-MEDIA FORM FACTOR DERIVATIVES

The form factor between a 2D segment ℓ and a media point $\mathbf{x}$ (Figure 2.16, left) is defined as the integrated curve-media geometry term along all segment points. This is equivalent to the angular ratio covered by ℓ as seen from $\mathbf{x}$

$$F_\ell(\mathbf{x}) = \frac{1}{2\pi} \int_{\mathbf{y}_0}^{\mathbf{y}_1} \frac{\cos \theta_{\mathbf{y}}}{\|\mathbf{x} - \mathbf{y}\|} \mathrm{d}\ell(\mathbf{y}) = \frac{1}{2\pi} \arccos \left(\frac{\overrightarrow{\mathbf{xy}}_0}{r_0} \cdot \frac{\overrightarrow{\mathbf{xy}}_1}{r_1} \right).$$

where $r_i = \|\overrightarrow{\mathbf{xy}}_i\|$. The form factor gradient and Hessian become

$$\boxed{\nabla F_\ell(\mathbf{x}) = -\frac{1}{2\pi} \frac{\nabla \cos \theta'}{\sqrt{1 - \cos^2 \theta'}} \qquad (2.27)}$$

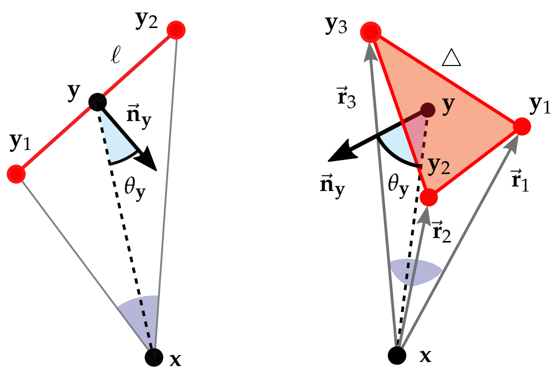


Figure 2.16: Setups for segment-to-media (2D, left) and triangle-to-media (3D, right) form factors.

$$\boxed{\begin{aligned}\mathbf{H}F_\ell(\mathbf{x}) = -\frac{1}{2\pi}\Bigg(&\frac{\mathbf{J}(\nabla\cos\theta')}{\sqrt{1-\cos^2\theta'}} \\ &+\frac{\cos\theta'}{(1-\cos^2\theta')^{3/2}}\nabla\cos\theta'\nabla^\intercal\cos\theta'\Bigg) \qquad (2.28)\end{aligned}}$$

where $\mathbf{J}$ is the Jacobian operator, and:

$$\nabla\cos\theta' = \frac{\cos\theta'}{r_0^2}\overrightarrow{\mathbf{xy}}_0 + \frac{\cos\theta'}{r_1^2}\overrightarrow{\mathbf{xy}}_1 \tag{2.29}$$

$$- \frac{(\overrightarrow{\mathbf{xy}}_0 + \overrightarrow{\mathbf{xy}}_1)}{r_0 r_1}, \tag{2.30}$$

$$\begin{aligned}\mathbf{J}(\nabla\cos\theta') = &-\mathbf{J}\left(\frac{\overrightarrow{\mathbf{xy}}_0}{r_0r_1}\right) - \mathbf{J}\left(\frac{\overrightarrow{\mathbf{xy}}_1}{r_0r_1}\right) \\ &+\mathbf{J}\left(\frac{\cos\theta'}{r_0^2}\overrightarrow{\mathbf{xy}}_0\right) + \mathbf{J}\left(\frac{\cos\theta'}{r_1^2}\overrightarrow{\mathbf{xy}}_1\right),\end{aligned} \tag{2.31}$$

$$\mathbf{J}\left(\frac{\overrightarrow{\mathbf{xy}}_i}{r_0r_1}\right) = \frac{\mathbf{J}(\overrightarrow{\mathbf{xy}}_i)}{r_0r_1} + \frac{\overrightarrow{\mathbf{xy}}_i\overrightarrow{\mathbf{xy}}_0^\intercal}{r_0^3r_1} + \frac{\overrightarrow{\mathbf{xy}}_i\overrightarrow{\mathbf{xy}}_1^\intercal}{r_0r_1^3}, \tag{2.32}$$

$$\begin{aligned}\mathbf{J}\left(\frac{\cos\theta'}{r_i^2}\overrightarrow{\mathbf{xy}}_i\right) = &\frac{\cos\theta'}{r_i^2}\mathbf{J}(\overrightarrow{\mathbf{xy}}_i) + \frac{\overrightarrow{\mathbf{xy}}_i}{r_i^2}\nabla^\intercal\cos\theta' \\ &+\frac{2\cos\theta'}{r_i^4}\overrightarrow{\mathbf{xy}}_i\overrightarrow{\mathbf{xy}}_i^\intercal.\end{aligned} \tag{2.33}$$

2.C 3D TRIANGLE-MEDIA FORM FACTOR DERIVATIVES

The form factor between a 3D triangular face $\triangle$ and a media point $\mathbf{x}$ (see Figure 2.16, right) is defined as the integrated surface-media geometry term along all points in the triangle. Analogous to 2D, this has analytical solution equal to the ratio of solid angle covered by the triangle as seen from $\mathbf{x}$,

$$F_{\triangle}(\mathbf{x}) = \frac{1}{4\pi} \int\limits_{\mathbf{y}\in\triangle} \frac{\cos\theta_{\mathbf{y}}}{\|\mathbf{x}-\mathbf{y}\|^2} \mathrm{d}\triangle(\mathbf{y}) = \frac{\Omega}{4\pi}. \tag{2.34}$$

Solid angle Ω of a triangle can be computed as [252],

$$\Omega = 2\arctan\frac{|A|}{B} \tag{2.35}$$

with

$$A = \vec{\mathbf{r}}_1 \cdot (\vec{\mathbf{r}}_2 \times \vec{\mathbf{r}}_3) \tag{2.36}$$

$$B = \mathrm{r}_1\mathrm{r}_2\mathrm{r}_3 + (\vec{\mathbf{r}}_1\cdot\vec{\mathbf{r}}_2)\,\mathrm{r}_3 + (\vec{\mathbf{r}}_2\cdot\vec{\mathbf{r}}_3)\,\mathrm{r}_1 + (\vec{\mathbf{r}}_1\cdot\vec{\mathbf{r}}_3)\,\mathrm{r}_2 \tag{2.37}$$

where $\vec{\mathbf{r}}_i = \overrightarrow{\mathbf{x}\mathbf{y}_i}$, and $\mathrm{r}_i = \|\vec{\mathbf{r}}_i\|$ (see Figure 2.16, right). Note that the numerator A requires an absolute value to ensure positive vector order (i.e. triangle winding) with respect to $\mathbf{x}$. Also, when obtaining negative arctangent values, π must be added to the obtained solid angle.

The gradient of the form factor with respect to a translation of $\mathbf{x}$ becomes

$$\nabla F_{\triangle}(\mathbf{x}) = \frac{1}{2\pi}\nabla\arctan\frac{|A|}{B} \tag{2.38}$$

$$= \frac{1}{2\pi}\frac{B\nabla|A| - |A|\nabla B}{|A|^2 + B^2}, \tag{2.39}$$

and its Hessian yields

$$\mathbf{H}F_{\triangle}(\mathbf{x}) = \frac{1}{2\pi}\left(\frac{\nabla(|A|)\nabla^{\intercal}B - \nabla B\nabla^{\intercal}(|A|)}{|A|^2+B^2} + \frac{B\mathbf{J}(\nabla(|A|)) - |A|\mathbf{J}(\nabla B)}{|A|^2+B^2} - \frac{(B\nabla(|A|) - |A|\nabla B)\left(\nabla(|A|^2) + \nabla(B^2)\right)^{\intercal}}{\left(|A|^2+B^2\right)^2}\right). \tag{2.40}$$

Note that for computing the terms $\nabla(|A|)$ and $\mathbf{J}(\nabla|A|)$, we can apply the derivatives of the absolute value of a vector function:

$$\nabla(|A|) = \frac{A}{|A|}\nabla A, \tag{2.41}$$

$$\mathbf{J}(\nabla|A|) = \frac{A\mathbf{J}(\nabla A) + \nabla A\nabla^{\intercal}A}{|A|} - \frac{A^2(\nabla A\nabla^{\intercal}A)}{|A|^3}. \tag{2.42}$$

The gradient of A becomes

$$\nabla A = \mathbf{J}\,(\vec{\mathbf{r}}_2\times\vec{\mathbf{r}}_3)\vec{\mathbf{r}}_1 + \mathbf{J}\,(\vec{\mathbf{r}}_1)\,(\vec{\mathbf{r}}_2\times\vec{\mathbf{r}}_3)\,. \tag{2.43}$$

By the Jacobi identity we have that

$$\mathbf{J}\left(\vec{\mathbf{r}}_2 \times \vec{\mathbf{r}}_3\right) = \vec{\mathbf{r}}_2 \times \mathbf{J}(\vec{\mathbf{r}}_3) - \vec{\mathbf{r}}_3 \times \mathbf{J}(\vec{\mathbf{r}}_2) \tag{2.44}$$

where any vector-matrix cross product $\vec{\mathbf{v}} \times \mathbf{J}(\bullet)$ can be expressed by means of the matrix multiplication form

$$\vec{\mathbf{v}} \times \mathbf{J}(\bullet) = \langle\vec{\mathbf{v}}\rangle\mathbf{J}(\bullet) \tag{2.45}$$

$$\vec{\mathbf{v}} = \begin{pmatrix} v_{(1)} \\ v_{(2)} \\ v_{(3)} \end{pmatrix}, \qquad \langle\vec{\mathbf{v}}\rangle = \begin{pmatrix} 0 & -v_{(3)} & v_{(2)} \\ v_{(3)} & 0 & -v_{(1)} \\ -v_{(2)} & v_{(1)} & 0 \end{pmatrix}. \tag{2.46}$$

Since $\mathbf{J}(\vec{\mathbf{r}}_1) = \mathbf{J}(\vec{\mathbf{r}}_2) = \mathbf{J}(\vec{\mathbf{r}}_3) = -I_3$, we have that

$$\boxed{\begin{aligned} \nabla A &= \left(\langle\vec{\mathbf{r}}_2\rangle\mathbf{J}(\vec{\mathbf{r}}_3) - \langle\vec{\mathbf{r}}_3\rangle\mathbf{J}(\vec{\mathbf{r}}_2)\right)\vec{\mathbf{r}}_1 - \left(\vec{\mathbf{r}}_2 \times \vec{\mathbf{r}}_3\right) \\ &= \langle\vec{\mathbf{r}}_3 - \vec{\mathbf{r}}_2\rangle\vec{\mathbf{r}}_1 - \left(\vec{\mathbf{r}}_2 \times \vec{\mathbf{r}}_3\right). \end{aligned}} \tag{2.47}$$

Note that $\langle\vec{\mathbf{r}}_3 - \vec{\mathbf{r}}_2\rangle = \langle\mathbf{y}_3 - \mathbf{y}_2\rangle$ and therefore does not depend on $\mathbf{x}$, and $\langle\vec{\mathbf{v}}\rangle^\intercal = \langle-\vec{\mathbf{v}}\rangle$ (see Equation (2.46)). As a result, the Jacobian of ∇A becomes a zero matrix

$$\boxed{\begin{aligned} \mathbf{J}\left(\nabla A\right) &= \mathbf{J}(\vec{\mathbf{r}}_1)\langle\vec{\mathbf{r}}_3 - \vec{\mathbf{r}}_2\rangle^\intercal - \mathbf{J}\left(\vec{\mathbf{r}}_2 \times \vec{\mathbf{r}}_3\right) \\ &= \langle\vec{\mathbf{r}}_3 - \vec{\mathbf{r}}_2\rangle - \langle\vec{\mathbf{r}}_3 - \vec{\mathbf{r}}_2\rangle \\ &= 0. \end{aligned}} \tag{2.48}$$

The gradient of B becomes

$$\boxed{\begin{aligned} \nabla B = &\nabla(\mathrm{r}_1\mathrm{r}_2\mathrm{r}_3) + \nabla\left(\left(\vec{\mathbf{r}}_1 \cdot \vec{\mathbf{r}}_2\right)\mathrm{r}_3\right) \\ &+ \nabla\left(\left(\vec{\mathbf{r}}_2 \cdot \vec{\mathbf{r}}_3\right)\mathrm{r}_1\right) + \nabla\left(\left(\vec{\mathbf{r}}_1 \cdot \vec{\mathbf{r}}_3\right)\mathrm{r}_2\right) \end{aligned}} \tag{2.49}$$

where

$$\nabla(\mathrm{r}_1\mathrm{r}_2\mathrm{r}_3) = \mathrm{r}_2\mathrm{r}_3\nabla\mathrm{r}_1 + \mathrm{r}_1\mathrm{r}_3\nabla\mathrm{r}_2 + \mathrm{r}_1\mathrm{r}_2\nabla\mathrm{r}_3 \tag{2.50}$$

$$\nabla\left(\left(\vec{\mathbf{r}}_i \cdot \vec{\mathbf{r}}_j\right)\mathrm{r}_k\right) = \left(\vec{\mathbf{r}}_i \cdot \vec{\mathbf{r}}_j\right)\nabla\mathrm{r}_k - \mathrm{r}_k(\vec{\mathbf{r}}_i + \vec{\mathbf{r}}_j) \tag{2.51}$$

$$\nabla\mathrm{r} = -\frac{\vec{\mathbf{r}}}{\mathrm{r}}. \tag{2.52}$$

Jacobian of ∇B yields

$$\boxed{\begin{aligned} \mathbf{J}(\nabla B) = &\mathbf{J}(\nabla(\mathrm{r}_1\mathrm{r}_2\mathrm{r}_3)) + \mathbf{J}\left(\nabla\left(\left(\vec{\mathbf{r}}_1 \cdot \vec{\mathbf{r}}_2\right)\mathrm{r}_3\right)\right) \\ &+ \mathbf{J}\left(\nabla\left(\left(\vec{\mathbf{r}}_2 \cdot \vec{\mathbf{r}}_3\right)\mathrm{r}_1\right)\right) + \mathbf{J}\left(\nabla\left(\left(\vec{\mathbf{r}}_1 \cdot \vec{\mathbf{r}}_3\right)\mathrm{r}_2\right)\right) \end{aligned}} \tag{2.53}$$

where

$$\begin{aligned}\mathbf{J}(\nabla(\mathrm{r}_1\mathrm{r}_2\mathrm{r}_3)) = {} & \mathrm{r}_2\mathrm{r}_3\mathbf{J}(\nabla\mathrm{r}_1) + \mathrm{r}_1(\nabla\mathrm{r}_3\nabla^\intercal\mathrm{r}_2 + \nabla\mathrm{r}_2\nabla^\intercal\mathrm{r}_3) \\ & + \mathrm{r}_1\mathrm{r}_3\mathbf{J}(\nabla\mathrm{r}_2) + \mathrm{r}_2(\nabla\mathrm{r}_3\nabla^\intercal\mathrm{r}_1 + \nabla\mathrm{r}_1\nabla^\intercal\mathrm{r}_3) \\ & + \mathrm{r}_1\mathrm{r}_2\mathbf{J}(\nabla\mathrm{r}_3) + \mathrm{r}_3(\nabla\mathrm{r}_2\nabla^\intercal\mathrm{r}_1 + \nabla\mathrm{r}_1\nabla^\intercal\mathrm{r}_2)\end{aligned} \tag{2.54}$$

$$\begin{aligned}\mathbf{J}\left(\nabla\left(\left(\vec{\mathbf{r}}_i\cdot\vec{\mathbf{r}}_j\right)\mathrm{r}_k\right)\right) = {} & \left(\vec{\mathbf{r}}_i\cdot\vec{\mathbf{r}}_j\right)\mathbf{J}(\nabla\mathrm{r}_k) + 2\mathrm{r}_k I_3 \\ & - \nabla\mathrm{r}_k(\vec{\mathbf{r}}_i + \vec{\mathbf{r}}_j)^\intercal - (\vec{\mathbf{r}}_i + \vec{\mathbf{r}}_j)\nabla^\intercal\mathrm{r}_k\end{aligned} \tag{2.55}$$

$$\mathbf{J}(\nabla\mathrm{r}) = \frac{I_3}{\mathrm{r}} - \frac{\vec{\mathbf{r}}\,\vec{\mathbf{r}}^\intercal}{\mathrm{r}^3}. \tag{2.56}$$

3 REAL-TIME RENDERING ON A POWER BUDGET

With the inclusion of computationally-intensive graphics units in mobile devices, power consumption has be come a limiting constraint for many real-time graphics applications. This chapter presents efforts on this matter by proposing the first software-based power-optimal framework for real-time rendering. Our framework efficiently finds the optimal rendering settings that minimize image error for a chosen power requirement based on pre-computed energy maps of virtual scenarios. We demonstrate the benefits of this framework in desktop and mobile platforms, running in our own OpenGL rendering engine, and in the commercially available Unreal Engine [61].

This work was published in ACM Transactions on Graphics and presented at SIGGRAPH 2016. My particular role as a third author of this work was devising a practical real-time integration of NVIDIA GPU power measurement libraries in Unreal Engine, and performing energy consumption tests through parameter exploration within that framework.

R. Wang, B. Yu, J. Marco, T. Hu, D. Gutierrez & H. Bao
Real-time Rendering on a Power Budget
ACM Transactions on Graphics, Vol.35(4) (Presented at SIGGRAPH 2016)

3.1 INTRODUCTION

The increasing incorporation of GPUs on mobile, battery-powered devices during the last years has led to the emergence of many real-time rendering applications. These applications and the required computations, however, demand a high energy consumption. This has a significant impact on battery life, which becomes a limiting constraint for mobile devices. As a consequence, lowering the energy requirements on rendering applications has been recently identified as one of the next challenges in computer graphics [210]. However, a generalized methodology does not exist yet, and its possibilities remain largely unexplored.

Among the explored strategies to reduce energy consumption for graphics applications running on battery-powered devices, reducing the number of computations on the rendering pipeline has proved to be an effective solution (e.g., [9, 126, 217, 241, 275]). However, most existing solutions are based on ad-hoc decisions, tailored to a particular application. While previous works aiming to reduce computations in real-time rendering have relied on multi-objective cost functions defined by visual error, rendering time, or memory consumption [89, 212, 238, 262], we introduce a new cost model based on visual quality and *power usage*.

An ideal power-saving framework should have the following characteristics: 1) It guarantees an *optimal* tradeoff between the quality of the results and the target energy footprint; 2) The user can *adjust* both a target quality or a target energy consumption to prolong battery life; 3) It is *real-time*, and transparent to the user; 4) It *generalizes* across platforms and applications.

Finding the optimal settings from the usually huge set of rendering parameters available in graphics applications is a very challenging task, which requires an intelligent exploration of the large power-error space. This is further complicated by the desired real-time and multi-platform requirements. In this work, we address these challenges and present a real-time, power-optimal rendering framework that automatically finds the optimal tradeoffs between power consumption and image quality, and adapts the required rendering settings dynamically at run-time. We demonstrate how our adaptive exploration of the energy footprint of a rendering application can be leveraged to reduce power usage while preserving quality on the results. In particular our contributions are:

- We formally formulate the power vs. error tradeoff as an optimization problem, and present a multi-objective cost model defined in a novel power-error space.
- Based on this model, we present a new two-stage rendering framework that efficiently explores the power-error space, and adaptively reduces rendering costs at run-time.
- We demonstrate the flexibility and effectiveness of our framework using both a custom-built, OpenGL rendering system, tested on a smartphone and a desktop PC, and the commercial renderer integrated in the Unreal Engine, running on a desktop PC.

3.2 RELATED WORK

Energy-aware devices and algorithms are becoming a prolific research topic, with recent examples in fields like data management [15], systems design [159], cloud photo enhancement [65], or display technology [183], to name a few. This last one is maybe the field where energy consumption has been more thoroughly researched, while power-efficient rendering algorithms are increasingly drawing attention, largely motivated by the widespread adoption of mobile devices.

POWER SAVING FOR DISPLAYS In the last decade or so, many existing works have focused on reducing energy consumption in displays [190, 233]. For back-lit LCD displays, most of the light is converted to heat, a problem that is aggravated for HDR displays [183]. Dimming is the most common energy-saving strategy, e.g., simply by reducing the intensity of the background light [201], darkening inactive regions [105], or by concurrent brightness and contrast scaling [31]. More modern OLED displays allow energy control at individual pixel level [40, 55], which enables more sophisticated strategies like saliency-based dimming [29]. Energy-efficient color schemes have been proposed, for instance as a set of distinguishable iso-lightness colors guided by perceptual principles [33], or by finding a suitable color palette by means of energy minimization [41]. Chen et al. [30] present an optimization approach for volume rendering, optimizing color sets in object space instead of image space. Vallerio and colleagues [251] and Ranganathan et al. [221] explore energy efficiency for displays in the context of designing user interfaces.

POWER SAVING FOR GPUS With the establishment of GPUs and mobile devices, several specific pipeline designs and hardware implementations have been developed to optimize resources and reduce power usage during rendering (see for instance [217, 241, 275]). Möller and Ström [3] presented a survey about GPU design, where power consumption plays a key role, while the recent thesis by Johnsson [126] offers for a more detailed discussion of hardware-related aspects concerning power usage. Arnau et al. [9] reduce mobile GPUs energy consumption by removing redundancy of fragment shaders operations at hardware level. Instead, we present a purely software-driven power optimization strategy, agnostic to the underlying hardware being used.

Tile-based deferred rendering (TBDR) [218] identifies the portions of the scenes that can be ignored in the very early stages of rendering, therefore saving GPU computation and power consumption. Johnsson et al. [127] compared the power efficiency of three rendering and three shadow algorithms on different GPUs, although they do not provide new energy-efficient algorithms. Recently, Cohade and Santos [34] presented their efforts on optimizing the power usage in the *Lego Minifigures* game, and Mavridis and Papaioannou [184] reported energy savings on GPU-implementation of coarse shading techniques [250]. Different from these works, our approach makes use of actual energy consumption and error measurements, to drive a real-time power-optimal rendering system.

Complementary to power optimization, other rendering resources such as memory bandwidth or computation time have been the focus of different optimization schemes [89, 262], aiming for a good tradeoff between image quality and rendering budgets. Complementary to these

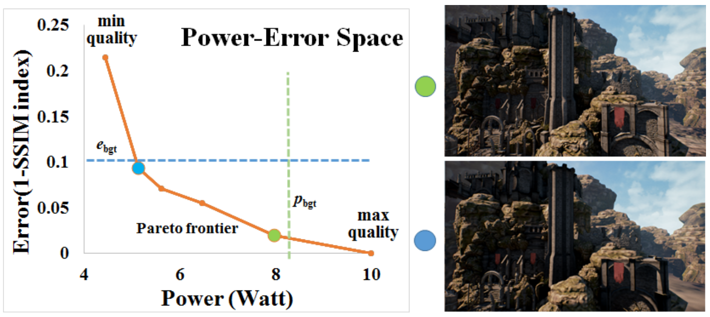


Figure 3.1: Illustration of the power-optimal rendering. With Pareto-optimal rendering settings, it is possible to obtain the optimal tradeoffs between power and visual error. Two rendering settings, marked in blue and green, are the optimal rendering settings with respect to the error budget e_{bgt} and the power budget p_{bgt}. One achieves the minimum power under the error budget, and the other obtain the minimum visual error under the power budget.

works, we aim to find an optimal compromise between image quality and a new challenging and constraining budget: energy consumption.

3.3 PROBLEM DEFINITION

In our context, it is useful to think about the rendering process as a function f that performs multiple rendering passes[1], and returns a color image. Each rendering function takes as input, on the one hand, the rendering settings s, defining the visual effects (shadow mapping, screen-space ambient occlusion, etc) and the specific parameters used for each one (such as map resolutions or kernel sizes); and on the other hand, the camera parameters c (position and view). It is clear that different rendering settings yield images with different quality for a given camera.

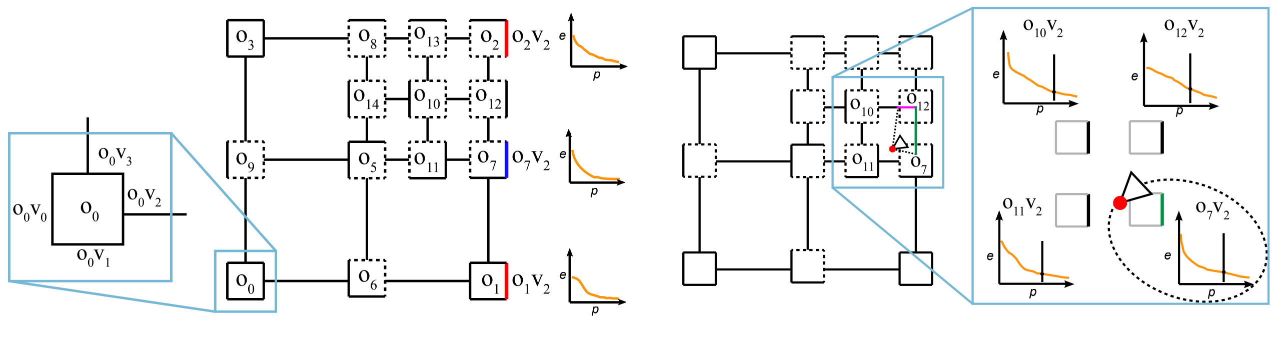
Let s_{best} denote the rendering settings that generate the best quality image. We can define the quality error e of any other image produced by different rendering settings as

$$e(c,s) = \iint_{xy} \| f(c, s_{\mathrm{best}}) - f(c,s) \| \, \mathrm{d}xy \tag{3.1}$$

where x, y define the pixel domain of the image, and $\| \cdot \|$ indicates the chosen norm.

Rendering with different functions $f(c,s)$ also has an impact on power usage. We can denote the power consumed during rendering of one frame as $p(c,s)$. In general, higher-quality images require more

[1]Generalizing the rendering process as a function f allows us to include both forward and deferred rendering frameworks.



(a) Adaptive measurement in an octree

(b) Runtime power optimal rendering

Figure 3.2: Overview of our power-optimal rendering process. For the sake of simplicity, we illustrate a 2D example using a quad-tree. **(a) Adaptive subdivision:** The initial node has four corners $o_{0..3}$, where each corner o_i defines four axis-aligned views $v_0..v_3$ (light blue zoomed-in node). Cameras $c = (o_i, v_j)$ are placed at every position-view pair, where we compute their Pareto frontiers. For each pair of adjacent camera samples looking at the same view —e.g. o_2, v_2 and o_1, v_2, highlighted in red— if either the error or power difference between their Pareto frontiers is larger than a threshold, we subdivide the corresponding node and compute the Pareto frontier of the new camera sample (o_7, v_2), highlighted in blue. Otherwise, the Pareto frontiers of the new cameras are inherited (dashed lines). This process is repeated for each node until a certain depth level or given error and power difference thresholds. **(b) Rendering settings at run-time:** Given a camera position and its node in our structure, we select the closest camera sample and corresponding view (o_7, v_2) (green). The optimal rendering settings are then obtained from its Pareto frontier.

power, while rendering a minimum quality image can save over 50% of the power compared to the maximum quality (see Table 3.3). It is therefore possible to find suitable tradeoffs between quality and power usage, to either obtain the best rendering quality under a given power budget, or to ensure a minimal power consumption given a desired rendering quality. We call this *power-optimal rendering*. The optimization for a given power budget p_{bgt} can be formulated as

$$s = \arg\min_s e(c,s) \qquad \text{subject to} \qquad p(c,s) < p_{\mathrm{bgt}}, \tag{3.2}$$

whereas given a target quality defined by the error budget e_{bgt}, the optimization becomes

$$s = \arg\min_s p(c,s) \qquad \text{subject to} \qquad e(c,s) < e_{\mathrm{bgt}}. \tag{3.3}$$

3.4 POWER-OPTIMAL RENDERING

We formulate our power-optimal rendering approach as a multi-objective optimization in a visual quality and power usage space. In this section we introduce our multi-objective cost model and the basic idea to solve the power-optimal rendering problem.

3.4.1 *Multi-objective Cost Model*

Different from other works [89, 212, 238, 262], our novel multi-objective cost model is based on visual quality and *power usage*. To optimize the rendering settings s of a given camera c, we first introduce a partial order to compare two different rendering settings s_i and s_j, and say that s_i is preferred over s_j (written as $s_i \prec s_j$) if either $e(c,s_i) < e(c,s_j) \wedge p(c,s_i) \leq p(c,s_j)$ or $e(c,s_i) \leq e(c,s_j) \wedge p(c,s_i) < p(c,s_j)$. That is, one rendering setting is preferred over another if it improves in quality or power usage, and is at least as good in the other.

Using our partial order, the Pareto frontier of all rendering settings $P(U) = \{u \in U : \forall u' \in U, u \not\prec u'\}$, can be regarded as the curve defining all power-optimal rendering settings in our two-dimensional cost space defined by (e,p). That is, the rendering settings in the Pareto frontier are preferred over other settings. Working in the domain of the Pareto frontier has one key advantage: given a power budget or an error budget, finding the optimal rendering settings is reduced to a 1D search on the Pareto curve. As we will see, this dimensionality reduction is a crucial aspect which will allows us to select optimal rendering settings at run-time. Figure 3.1 shows an example Pareto frontier, from which two optimal rendering settings have been selected (given a power budget and an error budget respectively). The resulting images are shown on the right.

3.4.2 *Adaptive Partition of The Camera View-Space*

By optimizing our multi-objective cost model, we have the solution of Equations (3.2) and (3.3) for one particular camera. However, given the high dimensionality of the camera view-space (composed of all possible camera positions and views) it would be impractical to carry out all Pareto-optimal optimization at run-time. Therefore, we introduce an adaptive partition of the camera view-space to store precomputed, optimized Pareto frontiers at given positions and views, which will later enable real-time optimal rendering. Such an adaptive partition is based on the observation that at some regions in the camera view-space, the rendering settings on the Pareto frontiers are quite different, but at other regions they are similar.

In particular, we use an octree structure, where one corner of octree node defines a camera position, and defines a discrete set of six views at each position, forming a view cube. Each position-view pair (o_i, v_j) thus describes one camera sample, $c = (o_i, v_j)$. For the sake of clarity, a simplified 2D version is shown in Figure 3.2a. At each position-view, we compute the Pareto frontier representing the optimal tradeoffs between power usage and rendering quality. The differences between these frontiers for adjacent (o_i, v_j) pairs will guide the adaptive partition of the space. In practice, we found that adaptive spatial subdivision along with six view orientations maintains a good tradeoff between structure complexity, temporal smoothness, and computational cost. A complete description of this process is described in Section 3.5.

3.4.3 *Algorithm Overview*

Figure 3.3 shows an overview of our algorithm, based on our multi-objective cost model, and our adaptive partition of the camera view-space. Our input is a 3D scene and a set of rendering effects and parameters (see Table 3.1 for the set used in our implementation). The entire algorithm is split into two main stages: the adaptive subdivision stage and the run-time rendering. As a preprocess, from our initial octree node, we measure the error in visual quality e and the power usage p for each camera c, exploring the space of all possible rendering settings s through Genetic Programming; this yields the Pareto frontier for such camera. We then compare the Pareto frontiers of each pair of adjacent cameras sharing the same view direction, and subdivide the octree if the difference is too large. We iteratively repeat this process until no more subdivisions are needed. At run-time, novel views can be rendered under the given quality or power budget by interpolating the optimal rendering settings at the nearest sample positions and views during user exploration of the scene. The following sections offers details on each of the main steps.

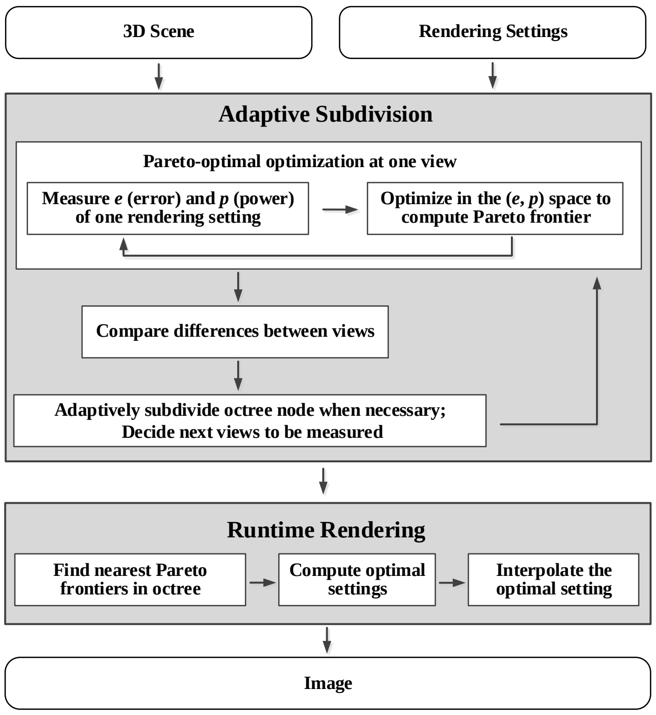

3D Scene
Rendering Settings
Adaptive Subdivision
Pareto-optimal optimization at one view
Measure e (error) and p (power) of one rendering setting
Optimize in the (e, p) space to compute Pareto frontier
Compare differences between views
Adaptively subdivide octree node when necessary; Decide next views to be measured
Runtime Rendering
Find nearest Pareto frontiers in octree
Compute optimal settings
Interpolate the optimal setting
Image

Figure 3.3: Our algorithm is split up into two main stages: the adaptive measurement stage (described in Section 3.5) followed by the runtime rendering stage (described in Section 3.6).

3.5 ADAPTIVE SUBDIVISION

As a preprocess, the adaptive subdivision partitions the camera view-space and stores Pareto-frontiers at sampled camera positions and views. It mainly takes following three steps.

3.5.1 *Pareto-Optimal Optimization at One Camera*

3.5.1.1 *Error and Power Measurement*

Given a camera c, we first render its reference image using the maximum quality settings. This image will then be used to compare the output of all other rendering settings, according to Equation (3.1). Instead of relying on pixel-wise error metrics such as the L^2 norm, we use the Structural Similarity Index (SSIM) [264] to measure the similarity and use one minus the similarity to obtain the error, i.e $e = 1 - \text{SSIM}$, which yields results that better predict human perception.

To measure power usage, we use two different approaches, depending on the target platform (desktop PC or mobile device). For the desktop PC, we use specific APIs provided by GPU vendors to access the hardware's internal power readings, and read back power consumption. For the mobile device we measure it directly instead, since we did not find any reliable APIs to measure power; we remove the battery and use an external metered power source to read power usage. More details are provided in the supplementary document[2].

3.5.1.2 *Exploring Potentially Optimal Settings*

For each camera c, the space of all possible rendering settings s is large. For an efficient exploration of such space, we rely on Genetic Programming (GP), inspired by recent works on shader simplification [238, 262]. Given its speed, we adapt the algorithm proposed by Deb et al. [39], which fits our multi-objective optimization in error and power space well.

First, we randomly combine parameters of rendering settings to generate the initial population. During partition, after every subdivision of the octree, children nodes are initialized by inheriting the optimal rendering settings of the parent nodes. This greatly accelerates the optimization process. To keep the diversity of the population while guiding the selection process towards a good spread of solutions on the Pareto curve, we use similar crowding heuristics to previous work [39]. We use crossover to combine partial solutions from high-fitness variants, along with mutation to avoid local optima. In particular, two rendering settings swap their parameter values to generate two offspring. Newly generated variants are considered and compared together with all preferred variants using our partial order. Newly preferred variants are selected to form the incoming population for the next iteration. The result of this process is a Pareto frontier defined for each camera.

3.5.2 *Comparing Pareto Frontiers*

The next step is to compare the Pareto frontiers of adjacent cameras sharing the same view direction, and evaluate the numerical difference between them, to decide whether the node should be subdivided. Our observation is that these adjacent cameras will cover a similar portion of the scene and produce a similar image, thus having similar Pareto-optimal rendering settings.

Suppose we already have two Pareto frontiers P_{c_0} and P_{c_1} taken from two different cameras c_0 and c_1. We separately measure their difference under both the error and power metrics:

$$D_e(P_{c_0}, P_{c_1}) = d_e(P_{c_0}, P_{c_1}) + d_e(P_{c_1}, P_{c_0}) \tag{3.4}$$
$$D_p(P_{c_0}, P_{c_1}) = d_p(P_{c_0}, P_{c_1}) + d_p(P_{c_1}, P_{c_0}) \tag{3.5}$$

[2] http://www.cad.zju.edu.cn/home/rwang/projects/power-optimization/16power_supp.pdf

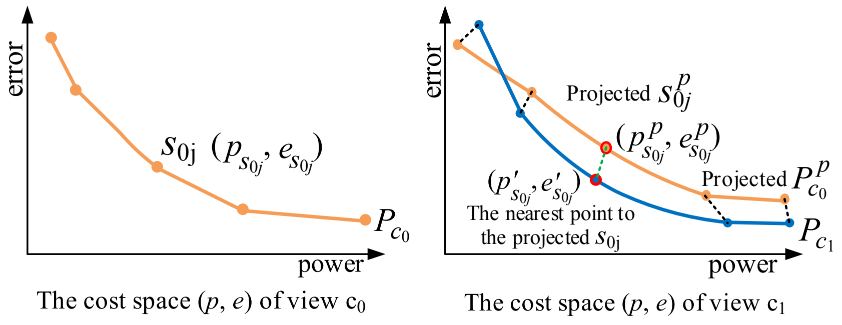


Figure 3.4: Illustration of the distance from the Pareto frontiers P_{c_0} (left, orange) to P_{c_1} (right, blue). One distance between the error and power $(e_{s_{0j}}, p_{s_{0j}})$ of projected points s_{0j} to nearest point $(e'_{s_{0j}}, p'_{s_{0j}})$ is visualized in green dash line (right). The total distance is averaged by all point-wise distance (black dashed lines) to the projected $P_{c_0}^p$ (right, orange).

where $d_e(P_{c_0}, P_{c_1})$ and $d_p(P_{c_0}, P_{c_1})$ are half-distance functions computing error and power differences, respectively, and $D_e(P_{c_0}, P_{c_1})$ and $D_p(P_{c_0}, P_{c_1})$ are the two full distance functions of error and power.

Figure 3.4 shows the process of comparing Pareto curves: To compute the half distance from P_{c_0} to P_{c_1}, we project P_{c_0} to the two-dimensional cost space of P_{c_1}. This can be done by using rendering settings of P_{c_0} to render scenes with camera c_1. Note that both error and power change in the projected curve $P_{c_0}^p$, since it is now related to a different view. Then we compute the distance between $P_{c_0}^p$ and P_{c_1}; for efficiency, we compute the point-wise distances and average them to obtain the total distance. Specifically, given a projected rendering setting s_{0j}^p defining a point $(p_{s_{0j}^p}, e_{s_{0j}^p})$, we find the nearest 2D point on P_{c_1}, defined as $n(c_1, s_{0j}^p) = (p'_{s_{0j}^p}, e'_{s_{0j}^p})$. The error and power distances between these two points are $d_e(s_{0j}^p, n(c_1, s_{0j}^p)) = |e_{s_{0j}^p} - e'_{s_{0j}^p}|$, and $d_p(s_{0j}^p, n(c_1, s_{0j}^p)) = |p_{s_{0j}^p} - p'_{s_{0j}^p}|$, respectively. The total error and power distance function from the projected Pareto frontier $P_{c_0}^p$ to P_{c_1} is given by:

$$d_e(P_{c_0}, P_{c_1}) = \frac{1}{N}\sum_j^N d_e(s_{0j}, P_{c_1}) \approx \frac{1}{N}\sum_j^N |e_{s_{0j}} - e'_{s_{0j}}| \tag{3.6}$$

$$d_p(P_{c_0}, P_{c_1}) = \frac{1}{M}\sum_j^M d_p(s_{0j}, P_{c_1}) \approx \frac{1}{M}\sum_j^M |p_{s_{0j}} - p'_{s_{0j}}| \tag{3.7}$$

where N and M are numbers of rendering settings on P_{c_0} and P_{c_1} respectively (we have removed the super-index p in s_{0j} to simplify notation). If the distances of either the error or power between two Pareto frontiers are larger than a given threshold, we adaptively subdivide our space, as explained in the following subsection.

3.5.3 *Adaptive Space Subdivision*

Since computing and comparing all the Pareto curves in the entire camera view-space is intractable, we adaptively partition this space into an octree, storing a set of discrete position-view pairs, according to the distance between Pareto frontiers. Let us consider the position-view pairs, i.e. two cameras (o_1, v_2) and (o_2, v_2) illustrated (as a 2D quadtree) in Figure 3.2.a, right. If either the error or the power difference between their Pareto frontiers is larger than a given threshold, this indicates that the current sampling of v_2 for adjacent camera positions o_1 and o_2 is insufficient to obtain all power-optimal settings in the space in-between, and the node needs to be subdivided. At the newly generated corner point (o_7 in Figure 3.2.a, right), we take (o_7, v_2) (blue) as a new camera and compute a Pareto frontier on it, iteratively repeating the comparison-subdivision steps until no more subdivision is required (adjacent Pareto curves are similar). Note that for other views at the corner point o_7, new optimization are only required on the views whose parent views differ above the given threshold; the rest of the views simply inherit one of the Pareto curves of their parent nodes. For views of corners at the center of node faces or at the node center that have more than two adjacent corners, e.g. o_5 and o_{10} in Figure 3.2.a, we pair their adjacent corners along axes and calculate the corresponding error or power difference. If the difference is larger than the threshold, we then perform optimization on it to compute new Pareto frontier.

3.6 RUNTIME RENDERING

At run-time, we leverage our adaptively partitioned camera view-space with the corresponding optimal rendering settings to ensure rendering power-optimal images. Observe a 2D quadtree example in Figure 3.2b. First, given a new camera position and user view, we traverse the octree to obtain the leaf node where it is located. We project the user's frustum onto the cubemap formed by the sides of the leaf, and select the side with the largest projected portion of the view frustum v_2 (see Figure 3.2b left, green). The corner closest to the camera position o_7, and the selected view v_2 determine a position-view camera sample (o_7, v_2), from which we fetch the precomputed Pareto frontier (see Figure 3.2b, right). Finally, given a power or error budget, and the selected Pareto frontier, we perform binary search along the frontier and obtain the optimal settings that are used to render the image.

TEMPORAL FILTERING OF RENDERING SETTINGS To avoid visible sudden changes in quality when choosing different cameras during real-time navigation of the scene, we apply a smoothing strategy based on temporal filtering. The filtered rendering settings are computed as

$$s_{\text{optimal}} = [(1 - \frac{t}{T})s_{\text{old}} + \frac{t}{T}s_{\text{new}}] \tag{3.8}$$

where the brackets denote the closest integer, s_{old} and s_{new} are the previous and current optimal rendering settings, respectively, t is time after applying a new rendering setting, and T is the time used for interpolation ($T = 2$ seconds as default).

3.7 IMPLEMENTATION

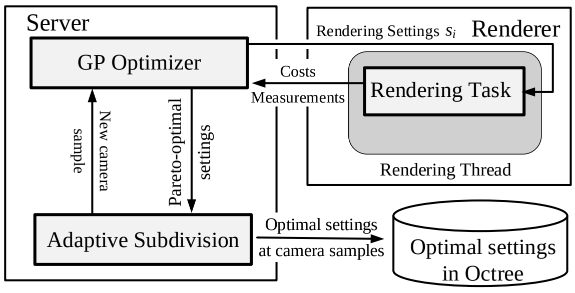
We have implemented an OpenGL-based rendering system, and tested it on two different platforms: One is a desktop PC with Intel Xeon E3-1230 CPU and an NVIDIA Quadro K2200 graphics card, running Microsoft Windows 7. The other is a smartphone with 2.2 GHz 8-core ARM Cortex-A53 CPU and PowerVR G6200 GPU, running Android 5.0.2. Additionally, we also validate our approach on a commercial rendering engine by integrating it into the Unreal Engine [61] on the desktop PC. Please refer to the supplementary material[2] for more details not covered in this section. Our code will be made available through our website.

3.7.1 *Power Measurement*

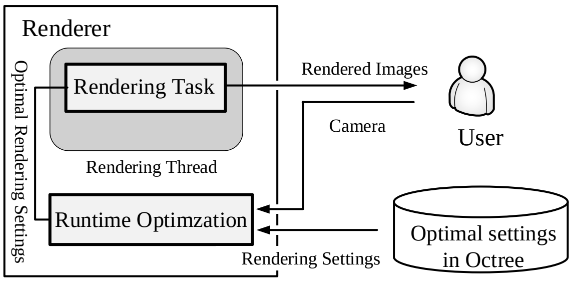
We first set our rendering system to a fixed frames-per-second rate, to guarantee comparable measurements where the only variable are the rendering settings. Then, we combine rendering settings from different cameras, following two different strategies according to the given platform.

DESKTOP PC To measure the power usage of the Quadro graphics card, we use the C-based API, NVIDIA Management Library (NVML) [193], to directly access the power usage of the GPU and its associated circuitry. According to the documentation, measurements are accurate to within ±5% of the current power draw. We average power measurements over a given time period to reduce variance: we generally take 10 seconds to measure the power and read back 10 times per second. Between two different rendering settings, we wait for 3 seconds without measurements to avoid any residual influence of the previous setting.

MOBILE DEVICE For the smartphone, we use an external source meter to directly supply the power of the device. The source meter



(a) Adaptive subdivision



(b) Runtime Rendering

Figure 3.5: Preprocess and runtime workflow of our system. Please refer to the text in Section 3.7 for details.

that we are using is a Keithley A2230-30-1, which allows direct cable connection and provides APIs to access the instantaneous voltage and current consumption. In practice, we set a constant voltage and read back the current, from which we obtain power. Note that in this case, both the CPU and GPU power usages are measured. Before the measurement, we close all unnecessary applications and services to reduce the unpredictable power consumptions of CPU. Since the power measurement of the mobile device has bigger variance than the desktop PC, we average over 25 seconds, and read back 10 times per second. The interval between the measurements of two rendering settings is 5 seconds.

3.7.2 *Rendering Systems*

3.7.2.1 *OpenGL-based Rendering System*

Figure 3.5 shows the architecture and the workflow of our rendering system. It consists of a renderer and a server, connected through sockets. The renderer is developed in C++ and OpengGL ES, to be easily used on different platforms. The server is implemented in C++ and only executed on the desktop PC. Our system has two rendering modes: In the *subdivision* mode, the renderer receives information from the server about the camera position and view to render scenes, to perform the adaptive partition of the view space. After the preprocess, all measured and sampled data are then transferred from the server to the renderer. The *rendering* mode is active during free navigation of the scene. The renderer automatically searches in the stored hierarchy to find the power-optimal rendering settings at run-time.

RENDERING SETTINGS Our OpenGL rendering framework supports GPU-based *importance sampling* [35], *shadow mapping*, *screen-space ambient occlusion* (SSAO) [132], and *morphological antialiasing* (MLAA) [125]. For each, we can choose between different parameters and values to adjust the rendering quality, resulting in a varying power consumption. The combination of all these makes up the space of all rendering settings. For GPU-based importance sampling, the parameter we use is the number of samples generated at run-time; for shadow mapping, we choose the shadow map resolution as parameter; for SSAO, the parameter is the number of sample rays to compute the visibility; last, for MLAA, we vary the steps of the search to find edges in the pixel shader. The complete set of values is given in Table 3.1. To store these settings we use a 32-bit integer, where the index value of each effect takes up eight bits.

ADAPTIVE SUBDIVISION In subdivision mode, the server sends camera information and rendering settings to the renderer, to sample the camera view-space. For each sample, the server first requests the renderer to render the scene with maximum quality and store the image as a reference, to be used to compute quality errors. Then, the server runs the Genetic Programming algorithm to optimize the Pareto frontier. The server sends each of the candidate rendering settings to the renderer, and let the renderer use it to render the scene. The power usage is measured by the server, and the image error is measured by the renderer and sent back to the server. In the next step, the server compares the Pareto frontiers and selects the next camera sample. The process is repeated iteratively until no more subdivisions are needed. Last, the server stores all the Pareto frontiers at the views of corners of the final octree.

RUNTIME RENDERING In the real-time rendering mode, the position and view of the camera are used to guide the search in the octree. Then, the power-optimal rendering setting is retrieved and used to render the scene.

Table 3.1: List of parameters and values forming the space of rendering settings for our two renderers. For details in the Unreal Engine parameters, refer to the supplementary material[2].

Parameters	Values
In-house renderer	
Sample number in GPU sampling	8, 16, 32
Shadow map resolution	256, 512, 1024, 2048
Sample number in SSAO	4, 8, 16, 32
Search steps in MLAA	4, 8, 16, 32, 64
The renderer in Unreal Engine	
Anti aliasing	0, 2, 4, 6
Post processing quality	0, 1, 2, 3
Shadows quality	0, 1, 2, 3
Textures quality	0, 1, 2, 3
Effects quality	0, 1, 2, 3
Resolution scale	70%, 80%, 90%, 100%
View distance	0.1, 0.4, 0.7, 1.0

3.7.2.2 *Rendering System Using the Unreal Engine*

To test how well our framework generalizes to other rendering platforms, we implement it on the Unreal Engine. This framework also consists of two sub-systems, the renderer and the server, with similar roles as before. To integrate our rendering in the Unreal Engine, instead of defining two modes of operation, we develop two plug-ins for the subdivision and rendering tasks, respectively, adapting our in-house code to the Unreal architecture.

RENDERING SETTINGS The Unreal Engine provides a set of predefined settings to allow users to adjust the quality of several features. These can be tweaked at run-time, thus they fit well in our system. We select seven features (Table 3.1): *resolution scale*, *view distance*, *anti-aliasing*, *post-processing* quality, *shadows* quality, *textures* quality, and the *effects*. The complete set of values is given in Table 3.1, defining the space of all rendering settings. To store these settings we also use a 32-bit integer, where the index value of each effect takes up four bits.

3.8 RESULTS

We performed a series of experiments in order to demonstrate the effectiveness of our rendering framework on four different scenes. Our

Table 3.2: Statistics for the tested demos. Scene statistics include number of triangles, and size on disk. Subdivision statistics include number of levels and corners on the octree, number of computed views, size on disk, and preprocess time. Thresholds indicate power and error limits for subdivision of the octree (see Section 3.5 for details).

Demos	Renderer	Platform	Scene		Subdivision						Rendering
			Tris. #	Scene Size	Thld. (power, error)	Time (hrs)	Levels	Corners	Views	Data Size	Duration
Valley	in-house	PC	55.9 k	32.5 MB	(0.5W, 0.002)	29.9	4	279	759	18.1 KB	50 s
Hall	in-house	Mobile	31.7 k	25.8 MB	(0.15W, 0.002)	35.3	4	108	316	9.3 KB	50 s
Elven Ruins	UE4	PC	191.2 k	1.03 GB	(0.5W, 0.005)	67.1	5	393	956	36.6 KB	60 s
Sun Temple	UE4	PC	667.9 k	512 MB	(0.5W, 0.005)	35.4	3	125	511	18 KB	60 s

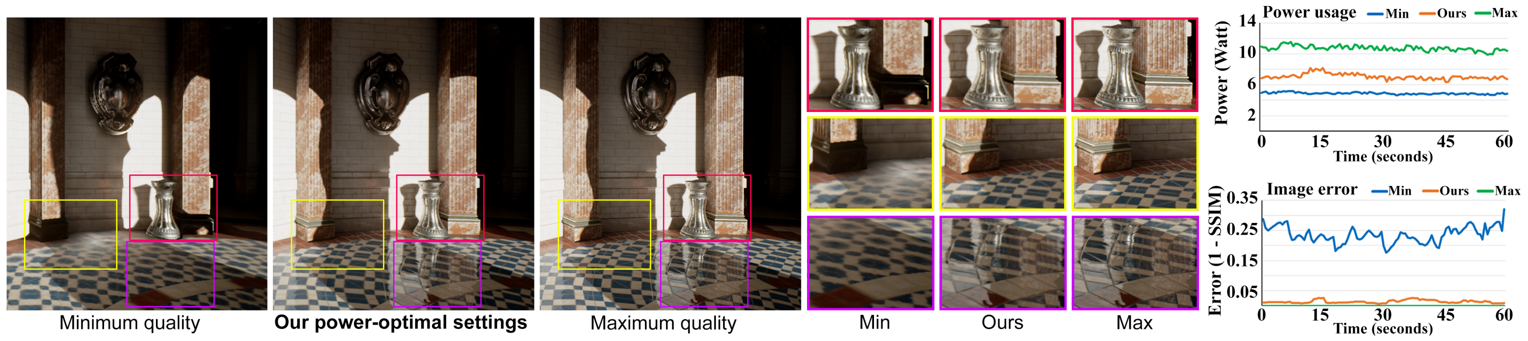


Figure 3.6: Results for the *Sun Temple* scene, where our Power-Optimal settings yield images of similar quality as *Maximum Quality*, but saving up to 34% in power consumption. The charts on the right show power consumption and image quality (measured with the perceptually-based SSIM metric), respectively.

in-house renderer runs on a desktop PC and a smartphone, and the renderer integrated in the Unreal Engine runs on the desktop PC. In Table 3.2, we summarize the statistics of the demo scenes. The FPS is set to 30 in all our experiments on desktop PC, and 10 on the smartphone due to the limited computational power.

VALLEY We use our in-house renderer to render the scene with four different effects on the PC. We set an environment light and a directional light, with GPU-based importance sampling. The directional light casts shadows, with the shadow map resolution as one of the parameters in our optimization. The screen-space ambient occlusion (SSAO) and morphological anti-aliasing (MLAA) are computed as post-processes.

HALL Each polygon has a diffuse map and a specular map. We use our in-house renderer to render the scene on the smartphone. Since the GPU-based importance sampling effect requires environment lighting, we initially render the scene onto a cubemap centered in the hall, and use it as the environment map. A directional light is set to illuminate the scene through the door. As in the previous scene, SSAO and MLAA are all computed in screen space as post-processes.

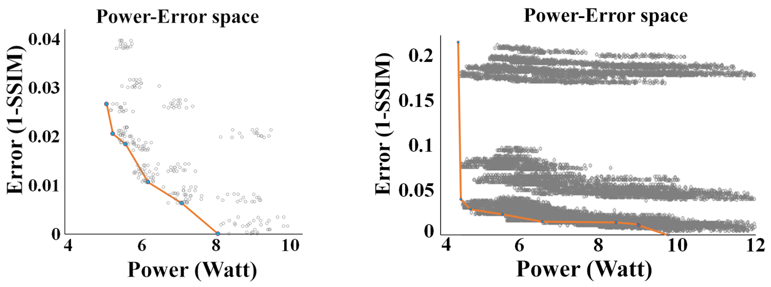
ELVEN RUINS This demo is modified from an example scene shipped with the Unreal Engine. We use the plug-in that we developed in the Unreal Engine to render the scene.

SUN TEMPLE This demo is another example scene shipped with the Unreal Engine. We also use our Unreal Engine plug-in to render it.

3.8.1 *Adaptive Subdivision and Pareto Frontier*

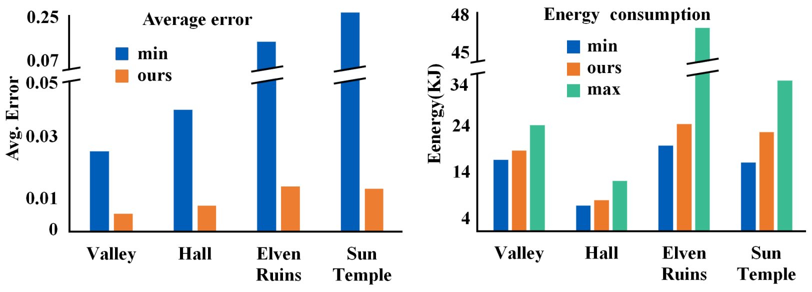
Power and error thresholds used to trigger subdivision of the octree are shown in Table 3.2. Note that since the parameter space is different for our in-house renderer and Unreal Engine, while power consumption also varies on each platform, we set different initial parameters for each platform-renderer pair. For the Genetic Programming (GP) algorithm, we set the maximum iterations to 25 in our in-house renderer. In Unreal Engine, we increase the maximum iterations to 40 due to the higher complexity of the parameter space. As Table 3.2 shows, the extra memory overhead is negligible in both cases, in the order of a few KB.

Figure 3.7 shows two example plots of our entire power-error cost space for one view in the *Valley* and *Elven Ruins* scenes. The Pareto frontiers optimized by our GP algorithm are shown in orange. The combinations of all different rendering settings are shown in dark grey.



(a) One view in *Valley* demo (b) One view in *Elven Ruins* demo

Figure 3.7: The entire power-error cost spaces and Pareto frontiers optimized by our GP algorithm at two example views. Grey dots are rendering settings and the orange line is the Pareto frontier. (a) One view in the *Valley* demo with 300 rendering settings. (b) One view in the *Elven Ruins* demo with 16384 rendering settings.

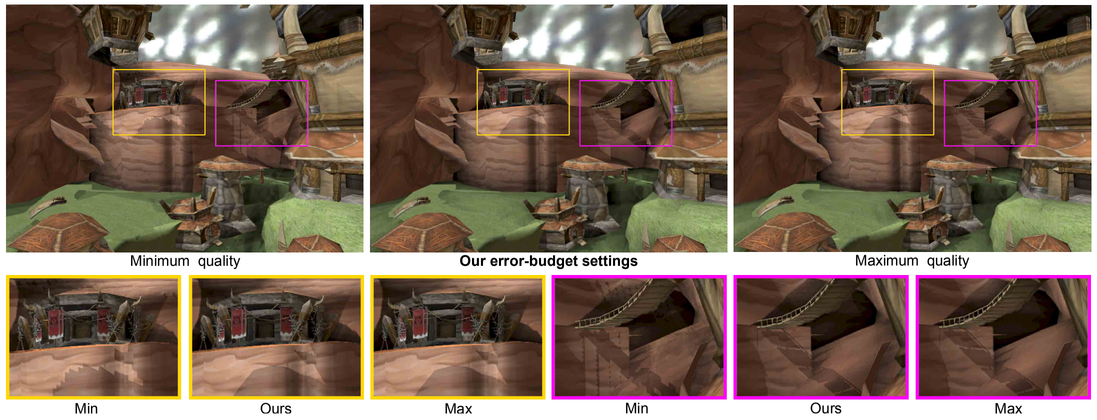


(a) Average visual error (b) Energy consumption

Figure 3.8: Average visual error and total energy consumption under different rendering settings in our four demos.

3.8.2 *Runtime Power-Optimal Rendering*

Although our approach supports run-time free exploration, for comparison purposes we record a camera path and repeat the motion while testing different power or error budgets. To obtain stable reliable measurements, all paths last between 50 and 60 seconds. The maximum quality and the minimum rendering settings are regarded as the baselines. Then, for the different demos, we use different power or error budgets to guide our run-time power-optimal rendering. Figure 3.8 shows the average visual error and total energy consumption we measured. It can be seen how our framework drastically minimizes visual error, while keeping power consumption very close to the minimum-quality settings. We describe our demos in this section, and refer the



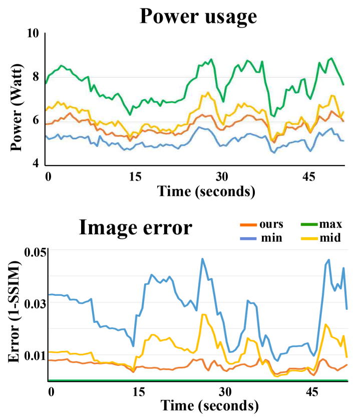


Figure 3.9: Real-time demo for the *Valley* scene, using our in-house render engine on a desktop PC. We compare minimum, middle and maximum quality settings against our power-optimal rendering framework, selecting an error threshold of 0.01 (renderings with middle settings not shown in the figure). Our method outputs images almost identical to those produced with the maximum-quality settings. The plots on the right show power usage and error during 50 seconds of free camera navigation: Our framework keeps power consumption almost as low as the minimum settings (top), while ensuring that errors stay below the given threshold. Please refer to the supplementary video for the full demo.

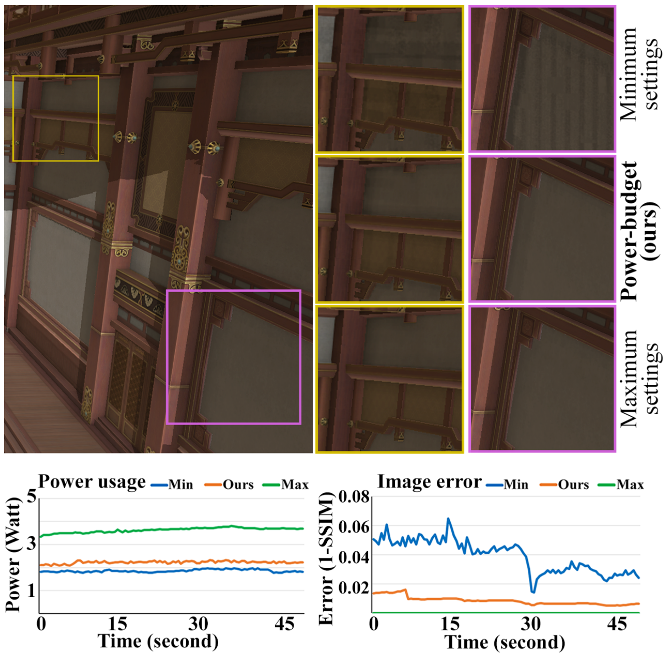


Figure 3.10: *Hall* real-time demo using our in-house render engine on a smartphone. Top-left image shows the maximum settings render. We compare insets (top-right) of minimum and maximum quality settings against our power-optimal rendering framework, selecting a power threshold of 2.2W, close to consumption at the minimum quality settings. Plots on the bottom show power usage and error during a 50-second camera path. Our optimized settings maintain a power usage below the given budget, while providing a quality close to the maximum settings. Please refer to the supplementary video for the full demo.

reader to the supplemental video[3] for the animations and the details of all the rendering settings.

Figure 3.6 shows the *Sun Temple* scene running on the Unreal Engine, under a power budget $p = 7W$. From the zoomed-in insets, it can be clearly seen how the quality produced with our framework is very close to the maximum quality, while the minimum rendering settings introduce visible artifacts such as wrong shadows, over-blurred areas, or missing reflections. The plots on the right show a lower power usage than maximum quality, with negligible error.

Figure 3.9 shows the *Valley* scene rendered on a desktop PC. During navigation under a visual error budget $e_{\text{bgt}} = 0.01$, our system automatically retrieves the optimal rendering settings to produce the image in

[3] http://www.cad.zju.edu.cn/home/rwang/projects/power-optimization/16power_video.mp4

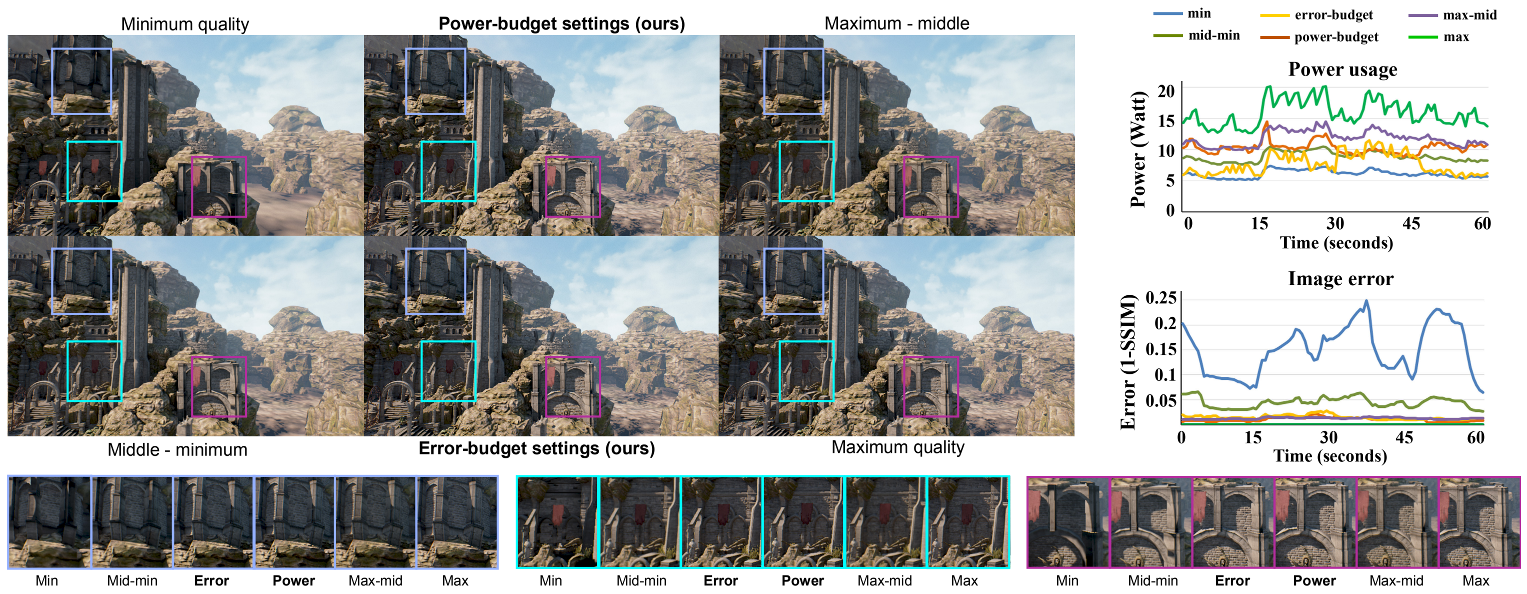


Figure 3.11: *Elven Ruins* real-time demo using the Unreal Engine on a desktop PC. We compare minimum, middle-minimum, maximum-middle and maximum quality settings against two modes of our power-optimal rendering framework: Selecting a power budget of 10W, and an error budget of 0.02. Plots on the right show power usage and error during a 60-second camera path. Our framework provides power-error optimized settings under the two different budget modes, while guaranteeing in both cases a visual quality very similar to the maximum settings, and power usage close to the minimum settings. Moreover, our optimized settings outperform manual settings. Please refer to the supplementary video for the full demo.

Elven Ruins					
	Ours	Max	Max-Mid	Mid-Min	Min
Avg. Power (W)	7.87	15.87	11.80	8.88	6.11
Avg. Error (e)	0.018	0	0.013	0.046	0.151

Sun Temple					
	Ours	Max	Max-Mid	Mid-Min	Min
Avg. Power	7.09	10.73	8.51	6.00	4.90
Avg. Error	0.0145	0	0.0158	0.068	0.236

Table 3.3: Average power consumption and error for the *Elven Ruins* and *Sun Temple* demos, using different rendering settings. Our power optimal framework (Ours), achieves the best tradeoff, producing images almost identical to the maximum quality settings while reducing power between 30% and 50% approximately.

real-time. We measure and compare, for the maximum and minimum rendering settings, the power usage and the error curves, plotted in Figure 3.9, right. Since this scene is relatively simple, the error between the maximum and minimum rendering settings is not extremely large. But even in this case, our method is able to find the optimal tradeoffs that keep the error within budget, while significantly reducing power usage.

Figure 3.10 shows a detail of the *Hall* demo running on the smartphone. We set a power budget $p = 2.2W$. Three power usage curves and error curves are plotted in Figure 3.10, bottom. Our method stays within the power budget, offering a good tradeoff between power and error.

Figure 3.11 shows the *Elven Ruins* demo, running on the Unreal Engine. Here we set two different budgets, a power budget $p = 10W$, and a visual error budget $e = 0.02$. Our system takes these two budgets into account during navigation, and selects power-optimal rendering settings accordingly. The plots on the right show how, if we use the power budget to guide the rendering, power consumption is more stable than using the error budget. In this case, our framework dynamically finds some rendering settings that dramatically reduce power consumption, bringing it close to the minimum consumption (at 25-32s and 48-60s), while maintaining a very low quality error. This is because our system automatically identifies which rendering parameters have a larger impact on quality for the current view, while still maintaining low power consumption.

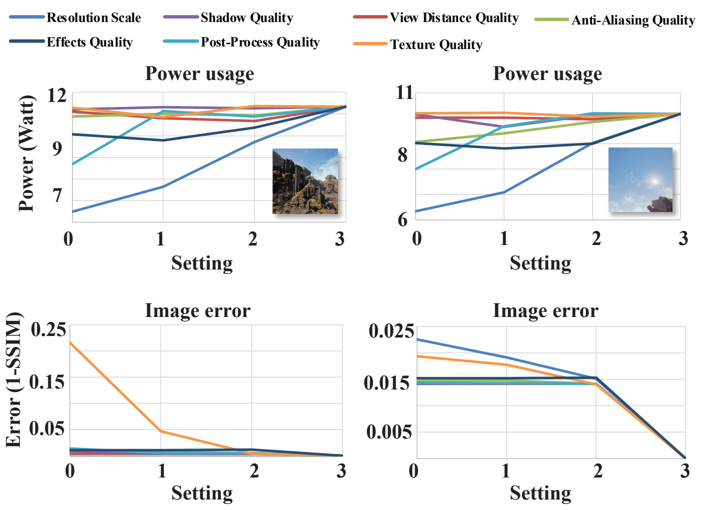


Figure 3.12: The influence of the different parameters is highly dependent on the particularities of the scene and view being rendered. Shown are two views of the *Elven Ruins* demo. For each view, we first set all parameters to produce the maximum rendering quality, and use it as the base setting. Then, we individually change only one parameter, from minimum to maximum value (shown as 0..3 in the figure), while keeping the others at maximum level. From the power consumption plots, it can be seen that in this case Resolution, Effects and Post-Process are the most dominant. However, the error is inversely correlated with Texture quality for the first view, while Resolution has an insignificant impact.

3.8.3 *Analysis of Different Settings*

A key advantage of our optimization framework is its flexibility, being agnostic to the particular choice of parameters and settings. This is a key feature, since we have not found a predictable correlation between the values for the different parameters and their effect on power saving and error. This impact is instead highly dependent on the particularities of the scene and view being rendered. For instance, Shadow Quality will only have a measurable effect when shadows are clearly visible in the frame (see for instance Figure 3.9). The only exceptions to this for our parameter space are Resolution Scale, which has a direct correlation with power consumption, and Texture Quality, which in our tests seemed to impact image quality the most. However, even these two parameters have a very different influence on power and error depending on the rendered view, as Figure 3.12 shows. Given the entire camera view-space of a scene, it would obviously be impractical to manually preset all optimal rendering settings. Our framework allows

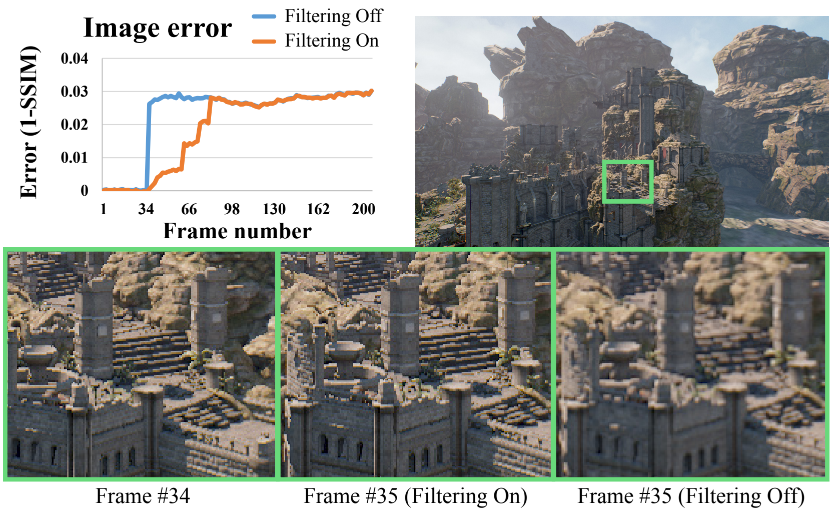


Figure 3.13: *Elven Ruins* scene with and without our temporal filtering, for a parameter change at frame #34. To illustrate the benefits, we propagate the settings at frame #34 to 200 frames, and compute the error of each frame with respect to the filtered (on) and non-filtered (off) versions. Notice how temporal filtering improves consistency, avoiding visible popping artifacts (sudden jump in the orange curve) by providing a smoother transition between settings. This is also shown in the insets comparing frames #34 and #35.

us to automatically select optimal power and error settings at runtime, without human intervention of prior knowledge about the scene.

We have also conducted a comparison with manually set tradeoffs between power and quality. In the Unreal Engine, some settings can be manually tweaked, allowing users to adjust the quality of various features. We set settings for four quality levels: maximum (all values set to maximum), maximum-middle, middle-minimum, and minimum (all values set to minimum). For this test we use the *Elven Ruins* and the *Sun Temple* scenes. Figure 3.11 compares one view under different settings and the corresponding plots for power and error in the *Elven Ruins* scene. The statistics of average power usage and errors are shown in Table 3.3. As can been seen, our method provides an excellent balance between visual error and power consumption: In the *Elven Ruins* demo, our method only consumes 7.87 W, which represents a saving of 50.4% of power compared to the maximum setting, and the visual quality is an order of magnitude better than the minimum setting. Similar conclusions can be inferred for the *Sun Temple* demo, with about 30% less power consumed. These results clearly demonstrate that our power-optimal framework is capable of automatically balancing optimal power consumption and quality, which would be very challenging to achieve by manually adjusting settings. Moreover, our framework provides *dynamic* optimal settings, while manually-set parameters in Unreal remain fixed throughout the demo.

3.8.4 *Temporal Filtering*

As described in Section 3.6, to reduce sudden changes in quality due to the runtime optimization of rendering settings, we apply a temporal filtering strategy. Despite this filtering being a discrete interpolation, our simple smoothing strategy improves the rendering quality in many cases. Figure 3.13 shows an example of 200 frames, including a runtime change of parameters with and without temporal filtering. We use the parameters before this change to render 200 frames, and regard them as reference to compute visual error. As shown in the plot, our temporal filtering provides a smoother transition, gradually modulating the visual error after a parameter change at frame #35, successfully reducing visible popping artifacts (refer also to the supplemental video[3]). The zoomed insets of frame #34 and #35 clearly demonstrate better consistence when applying the temporal filtering.

3.9 DISCUSSION AND FUTURE WORK

In some cases, the power or error curves may deviate slightly from the given budgets. This is due to the following reasons: First, at each view, the Pareto-optimal settings are discretely distributed in the power-error space. Therefore, the optimal setting computed under a budget may not exactly match the budget. Second, during the adaptive subdivision, the camera view-space is partitioned by thresholds until a fixed number of octree levels is reached. Therefore, in some local regions, using the optimal rendering setting at the closest sample camera may induce a slight deviation. In any case, as shown in Figure 3.11, the error and power curves remain very stable.

Since we focused on optimizing GPU consumption, we explicitly measured GPU power on desktop PC, and minimized CPU impact on mobile devices by deactivating as many external CPU sources as possible. While some rendering aspects may influence CPU power usage, in practice we found this variation negligible for the parameters we used. Nevertheless, it remains an interesting topic of future work to analyze the influence of a wider set of parameters on CPU power usage.

Our framework is not free of limitations and potential avenues of future work. First, it does not take into account dynamic changes in geometry or lighting. However, predicting the full space of all possible situations that may arise when dynamic changes area allowed is obviously intractable. One possibility to incorporate such changes in our optimization would be to precompute some of them, for instance the same view under different illuminations, and smoothly interpolate between settings at runtime. Our framework allows for such extensions of the power-error cost space, at the cost of longer preprocessing times. Nevertheless, this would only need to be done once; at run-time, the system would still be able to optimize in real-time, given our strategy of reducing the search for optimal settings to a one-dimensional Pareto curve.

Second, the capability to explore the full space of all possible combinations of rendering settings is limited by our GP optimization. Different strategies may yield slightly different Pareto frontiers, although we do not expect the final results to vary much in terms of power consumption or visual quality during navigation. Similarly, we have set our thresholds for the adaptive subdivision heuristically: although they provide a good balance between complexity of the subdivision and performance, we did not thoroughly explore the possibilities of other subdivision thresholds or schemes.

While the results in this work are strictly valid for the specific hardware configuration we used, our proposed framework is equally applicable to any other configurations. Moreover, we believe that the resulting optimization for a particular hardware setup will allow for a certain degree of transferability across similar configurations, by abstracting some of the dependencies. Finally, although the required precomputation time is not significant for large-scale productions, it would be interesting to find novel ways to reduce it, for instance by learning relationships among scene properties, rendering parameters and power usage, or acquiring higher-level knowledge about parameters.

To summarize, we hope that our power-saving rendering framework inspires future work in this direction. Our current implementation satisfies four key ideal characteristics: it produces optimal results between energy consumption and quality; it allows the user to fix either a power or a quality target; it is real-time; and it generalizes across different platforms. We have shown results on four different scenes, running on two different platforms, including a commercial one. Additionally, we have validated that our framework outperforms manually-set parameters, available in the Unreal Engine environment.

Part III

TRANSIENT LIGHT TRANSPORT

4 TRANSIENT LIGHT TRANSPORT: BASIC PRINCIPLES AND EXISTING APPROACHES

In the following chapter we first give an overview of light transport in transient state from a simulation perspective, and discuss existing reconstruction approaches and simulation algorithms. We then describe Time-of-Flight imaging, one of the paradigms for capturing transient light transport, and the different works that have addressed it. We finally summarize the existing approaches to tackle multipath interference, one of the main problems in transient imaging, which is specially aggravated in Time-of-Flight based applications.

This chapter is an adaptation of selected excerpts from the article *Recent Advances in Transient Imaging: A Computer Graphics and Vision Perspective*, which was published at *Visual Informatics*. As a third author of that work, my main role was covering the literature on Time-of-Flight imaging, and the different works on resolution of multipath interference in range imaging.

A. Jarabo, B. Masia, J. Marco & D. Gutierrez
Recent Advances in Transient Imaging:
A Computer Graphics and Vision Perspective
Visual Informatics, Vol.1(1), 2017

4.1 TRANSIENT LIGHT TRANSPORT SIMULATION

Light transport, described using either Maxwell's equations [22], or the more practical radiative approximation [28], is defined in a time-resolved manner. However, since the final goal is usually to compute light transport in steady-state, the practical assumption that the speed of light is infinite becomes a reasonable approximation from a simulation (rendering) perspective. See e.g. [81, 156, 205, 292] for an overview on steady-state rendering.

With the establishment of transient imaging in graphics and vision, the simulation of time-resolved light transport is becoming an increasingly important tool. Smith et al. [239] developed the first framework in the context of the traditional rendering equation [133]. This was later formalized by Jarabo et al. [111], extending the path integral [254] to include time-resolved effects such as propagation and scattering delays.

Transient rendering has been used to synthesize videos of light in motion [111], but is also a key tool to provide ground truth information to develop novel light transport models [1, 206], or benchmarking [198, 216]. It can also be used as a forward model for solving inverse problems [57, 58, 102, 123, 124, 142, 143, 148].

The key **differences** with respect to steady-state simulation are:

- The speed of light can no longer be assumed to be infinite, so propagation delays need to be taken into account. Note that some works in steady-state rendering also need to account for propagation delays (e.g. rendering based on wave-optics [189, 192], or solving the Eikonal equation for non-linear media [80, 104]), although their final goal is to obtain a steady-state image integrated in time.

- Scattering causes an additional delay, due to the electromagnetic and quantum mechanisms involved in the light-matter interaction. These give rise to effects such as fluorescence, or Fresnel phase delays. In the following Chapter 5, Figure 5.11 we show a few of these effects in a time-resolved manner.

- The temporal domain must be reconstructed; however, naive reconstruction strategies (i.e. frame-by-frame) are extremely inefficient.

- Motion in the scene (e.g. camera movements) brings about the need to include relativistic effects.

As we discuss in Chapter 5, rendering each transient frame independently is highly impractical, given the extremely short exposure times: sampling paths with a given temporal delay is almost impossible, while randomly sampling paths would be extremely inefficient. The most straightforward way to solve this issue and render effectively transient light transport is to reuse the samples for all frames, binning them in the temporal domain [1, 5, 108, 178, 206, 216]. This is equivalent to a histogram density estimation; although easy to implement, it has a slow convergence of $O(N^{-\frac{1}{3}})$, with N being the number of samples. In Chapter 5 [111] we present a better alternative, proposing a reconstruction method based on kernel density estimation [237], which leads to faster convergence ($O(N^{-\frac{4}{5}})$). Interestingly, rendering each frame independently, and using the histogram in the temporal domain, are equivalent to gate imaging [25, 165, 166, 170] and streak imaging techniques [62, 96, 259].

If the goal is not to generate the full transient profile, but just the modulated response at the sensor as if it were captured by a correlation-based sensor (see following Section 4.2.1), the problem is reduced to generating a single image modulating each sample according to its delay and the sensor response. Thus, while we still need to keep track of the path propagation delays, it can be done within the framework of the traditional path integral, where the sensor response is a function of time. For depth recovery, Keller and colleagues [142, 143] proposed a GPU-accelerated rendering system modeling such response. The system is limited to single-bounce scattering, so it assumes no MPI. The sensor response needs accurate sensor modulation models, including temporal behavior and noise. Gupta et al. [77] introduced a noise model for AMCW imaging devices, while Lambers et al. [160] presented other physically-based models of the sensor and the illumination, including high-quality noise and energy performance.

Depending on the application domain, existing algorithms to simulate transient light transport trade off accuracy for speed. As a forward model for efficient reconstruction of the geometry of occluded objects, Hullin [102] and Klein et al. [148] extended Smith et al.'s [239] transient version of the radiosity method [73] on the GPU. This method is limited to Lambertian surface reflections, and second-bounce interactions.

On the other hand, most works aiming at generating *ground truth* data have used transient versions of Monte Carlo (bidirectional) path tracing (BDPT) [1, 108, 111, 216]. These are unbiased methods, and support arbitrary scattering functions, including participating media. However, they are in general slow, requiring thousands of samples to converge. To accelerate convergence, in Chapter 5 [111] we introduce three techniques for uniform sampling in the temporal domain targeted to bidirectional methods. Lima et al. [171] and Periyasamy and Pramanik [213] proposed importance sampling strategies in the context of Optical Coherence Tomography. These techniques are designed to work in the presence of participating media; this is a particularly interesting case for transient imaging, since one of its key applications is seeing through such media (fog, murky water, etc). Camera movements at this temporal resolution bring about the need to simulate relativistic effects in transient light transport. These were simulated by Jarabo and colleagues [110, 112], including time dilation, light aberration, frequency shift, radiance accumulation and distortions on the camera's field of view. The system considered linear motion, as well as acceleration and rotation of the camera.

Other algorithms aiming to produce ground truth data robustly rely on a photon tracing and gathering approach [86, 121]. Meister and colleagues [186, 187] used a transient version of photon mapping, resulting into a robust estimation of light transport, and allowing to render caustics in the transient domain. Ament et al. [5] also used transient photon mapping to solve the refractive RTE. However, these techniques are intrinsically biased, due to the density estimation step at the core of the photon mapping algorithm. Our method proposed in Chapter 5 [111] reduces this bias by applying progressive density estimations along the temporal domain. Later, in Chapter 6 we introduce a transient version of the photon beams algorithm, providing optimal convergence rates for progressive reduction of bias and variance.

4.2 TRANSIENT IMAGING

From a capture perspective, transient imaging is usually interested in computing a transient image $\mathbf{i}(t)$, defined by the transient form of the light transport equation [206] as

$$\mathbf{i}(t) = \int_{-\infty}^{\infty} \mathbf{T}(\tau)\mathbf{p}(t-\tau)\mathrm{d}\tau \tag{4.1}$$

where $\mathbf{i}(t)$ stores the light arriving at time t, $\mathbf{p}(t)$ is the time-resolved illumination function at instant t, and $\mathbf{T}(t)$ is the transport matrix describing the light transport with a time-of-flight of exactly t. In practice, the transient image cannot be captured at instant t directly, given physical limitations of the sensor. Instead, the signal is also convolved by the temporal response of the sensor $\mathbf{s}(t)$ centered at t as:

$$\mathbf{i}(t) = \int_{-\infty}^{\infty} \mathbf{s}(t-\tau)(\mathbf{T} * \mathbf{p})(\tau)\mathrm{d}\tau \tag{4.2}$$

In order to capture the impulse response $\mathbf{T}$, there are several approaches depending on the type of illumination and sensor response used. If we focus only on illumination, the main lines of work have used either impulse illumination, or coded illumination. In the following we focus on the latter, where the coded illumination has been usually correlated with the coded sensor response, and the time-resolved response is computed by means of post-capture computation, as we show in the following.

4.2.1 *Time-of-Flight Imaging*

Phase-based time-of-flight (P-ToF) imaging, also called correlation-based time-of-flight (C-ToF) imaging or simply ToF imaging, cross-correlates emitted modulated light with frequency g_{ω_T}, and the impulse response of a pixel α_p, modulated and integrated at the sensor with frequency f_{ω_R} (see Figure 4.1). In its most typical continuous form (also known as *amplitude modulated continuous wave (AMCW)* systems[1]), the camera computes the cross-correlation as:

$$c(t) = \mathbf{s}(t) * \mathbf{p}(t), \tag{4.3}$$

with $\mathbf{s}(t)$ the radiance received at the sensor, and $\mathbf{p}(t)$ the emitted signal. These are in general modeled as:

$$\mathbf{s}(t) = \alpha_p \cos(f_{\omega_R} t + \phi) + \beta, \tag{4.4}$$

$$\mathbf{p}(t) = \cos(g_{\omega_T} t), \tag{4.5}$$

[1]Note that we use the term *AMCW* when referring to these specific sensors, whereas we use *ToF* for general phase-based time-of-flight sensors.

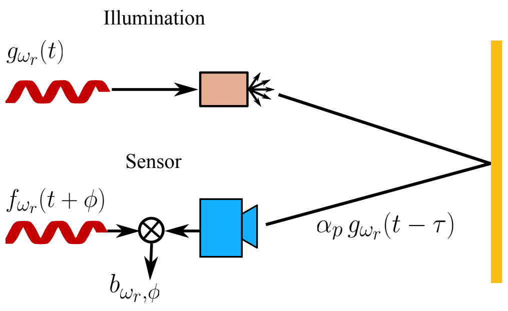


Figure 4.1: Basic operation principle of a time-of-flight emitter-sensor setup. Light is amplitude-modulated at the source, constantly emitted towards the scene, and each pixel modulates the impulse response of the observed scene point. By performing cross correlation (integration) of both modulated emitted and received signals, phase differences can be estimated to reconstruct light travel time (image from [93]).

where ϕ is the phase shift at the sensor, and β the ambient illumination. Capturing a set of different phase shifts ϕ allows to retrieve phase differences between the emitted and the received signals. These per-pixel phase differences correspond to light travel time, thus encoding distance (depth), and other possible sources of delay.

Early works demonstrated the applicability and performance limitations of this principle for range imaging in robotic environments [2, 90]. Due to hardware characteristics, these approaches were limited to a single range detection per shot, requiring systematic and time-consuming scanning of the scene to obtain a full depth map. The first prototype that allowed simultaneous scene capture with modulated array sensors was introduced by Schwarte et al. [229], coined under the denomination of *photonic mixer device (PMD)*. Lange and colleagues [161, 162] independently introduced a new type of ToF devices based on demodulation "lock-in" pixels, operating on CCD technology with modulation frequencies of a few tens of MHz, and allowing real-time range measurements. These technologies opened new avenues of research on applications and challenges imposed by hardware characteristics.

An important operational aspect of ToF setups resides in how the emitter and sensor frequencies are paired. Homodyne configurations use the same frequency at both emitter and sensor ($f_{\omega_R} = g_{\omega_T}$), while heterodyne ones use slightly different frequency pairs. While being more complicated computationally, heterodyne setups have been demonstrated to provide better ranging precision [36], allowing up to sub-millimeter resolution [42]. Additionally, proper calibration of ToF cameras was demonstrated to play a significant role when mitigating systematic errors on range estimation [58, 174].

Beyond traditional range imaging, Heide and colleagues [92] demonstrated that by correlating a set of sensor measurements with different modulation frequencies and phase shifts, a discrete set of per-pixel light travel times and intensities could be reconstructed through optimization, leading to an inferred transient image of the scene. However, the number of frequencies and phases required for this reconstruction is significantly higher than the default set provided by ToF devices (a few default frequencies and phases vs. hundreds of them). They work around this issue by substituting the built-in light source, signal generator and phase triggering by external elements. This ToF-based setup is much cheaper than femto-photography [259]; however, it only reaches nanosecond resolution (compared to picoseconds in femto-photography), the signal is reconstructed as opposed to directly captured, and tweaking the off-the-shelf devices requires a significant amount of skilled work[2].

Successive works aimed to overcome different ToF devices limitations that affect the viability of subsequent reconstruction methods. Kadambi et al. [130] reconfigured the emitter modulation with custom-coded illumination, which improved conditioning on the optimization by supporting sparsity constraints. This allowed them to recover per-pixel transient responses using a single frequency, instead of hundreds. Recent work by Peters and colleagues [215] introduced a way to generate robust sinusoidal light signals, which allowed them to obtain up to 18.6 transient responses per second using a closed-form reconstruction method.

ToF sensor noise, together with limited emitted light intensity due to safety and energy issues, make sensor exposure time and lens aperture the two main factors to achieve an acceptable SNR. To support real-time applications, exposure times must be kept short, so the aperture is usually large to capture as much available light as possible. This introduces a shallow depth of field that blurs scenarios with significant depth changes. Additionally, the low resolution of these sensors (e.g. 200x200 for PMDs) affects the spatial precision of the captured data. Godbaz and colleagues [69] provided a solution to the shallow depth of field by using coded apertures and explicit range data available in the ToF camera in order to perform defocus, effectively extending the depth of field. Xiao et al. [281] leveraged the amplitude and range information provided by the ToF devices to recover the defocus blur kernel and regularized the optimization in those amplitude and range spaces, allowing for defocus and increased resolution.

Regardless of wide apertures, exposure times need to be much longer than a single modulation period $T_{\omega_R} = 1/f_{\omega_R}$, in order to mitigate sensor noise. This causes a pathological problem known as *phase wrapping*. Since light travel time is encoded in the phase shift between

[2]http://www.pulsr.info/

emitted and received light, the modulation period T_{ω_R} determines the maximum light path length $c\,T_{\omega_R}$ that can be disambiguated, with c the speed of light. Any light path starting at the emitter that takes longer than this distance to reach a pixel in the sensor will *phase-wrap* T_{ω_R}, falling into the same phase shift than shorter paths within subsequent modulation periods. These phase-wrapped light paths produce interference in the measured data, leading to errors in the reconstruction. A straightforward way to solve this is to lower the modulation frequency, thus increasing the maximum unambiguous path length. However, this decreases the accuracy obtained for the reconstructed path lengths, leading to less precise depth measurements. Jongenelen et al. [129] demonstrated how to extend unambiguous maximum range while mitigating precision degradation, by exploring different dual combinations of simultaneous high and low modulation frequencies. Recently, the work by Gupta and colleagues [77] generalized the use of multiple high frequencies sequentially for this purpose in what they denominate micro-ToF imaging. Phase-wrapping is closely related to the widely-studied problem of MPI, where light from multiple light paths is integrated in the sensor resulting in signal interference and thus reconstruction errors. However, this is related to how some physical phenomena (e.g. interreflections, scattering) affect certain applications —actually affecting other capture methods too—, rather than to operational limitations of the ToF devices themselves. We provide a more detailed discussion of this problem in Section 4.2.2.

Recent works explore novel hardware modifications: Tadano and colleagues [247] increased temporal resolution beyond the limit of current ToF devices (around 100 picoseconds), by using arrays of LED emitters spatially separated by 3mm. This effectively corresponds to time shifts of 10 picoseconds. Shrestha and colleagues [234] explored imaging applications synchronizing up to three multi-view ToF cameras. To achieve this, they addressed interference problems between the light sources of the cameras, showing how they can be mitigated by using different sinusoidal frequencies for each sensor/light pair. The authors demonstrated applications such as improved range imaging for dynamic scenes by measuring phase images in parallel with two cameras, doubling single-camera frame rate, and mitigating motion artifacts.

4.2.2 *The multipath interference problem*

The multipath interference (MPI) problem is common for most transient imaging devices, specially in those with long exposure times: for example, in the context of gated-based LIDAR systems, where a modulated sensor response [165, 166] is used to robustly acquire depth. However, it is in ToF cameras where the problem is more noticeable.

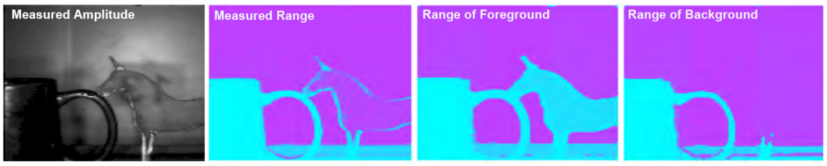


Figure 4.2: Given a scene with transparent objects (a), regular time of flight cameras fail at reconstructing their depth (b). Kadambi and colleagues' method, assuming that light transport is modeled as a sparse set of Dirac deltas, can correctly recover the depth of the unicorn (c and d). Figure from [130].

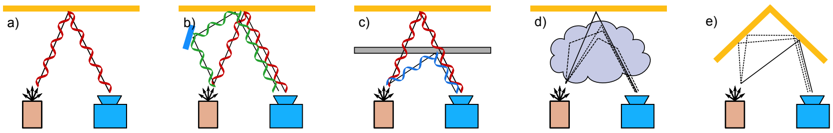


Figure 4.3: a) Range imaging using a ToF sensor, where the phase delay of the emitted modulated radiance encodes light time of flight. This formulation works for direct reflection, but presents problems when light from multiple paths is integrated in the sensor, as happens in the presence of specular reflections (b) or transmission (c). This problem gets even more complicated in the presence of diffuse transport, due to scattering in media (d) or Lambertian diffuse interreflections (e). Image after [18].

Some early approaches to solving the MPI problem in ToF cameras targeted in-camera light scattering [139, 227]; others targeted also indirect illumination but require placing tags in the scene [52], or made severe assumptions on scene characteristics [107]. For an in-depth discussion about the MPI problem from a signal processing perspective, we refer the reader to a recent article by Bhandari and Raskar [17].

The work of Fuchs [57] provided a model of MPI for the case in which all distracting surfaces are Lambertian, based on explicitly computing indirect illumination on the estimated depth map and iteratively correcting it. Follow-up works aimed at a more general solution targeting the source of the problem: the separation of the individual components when multiple returns are present [67, 68], also called Mixed Pixel Restoration. These techniques, however, cannot be used with off-the-shelf cameras, since they require measuring multiple phase steps per range measurement (as opposed to the usual four). Of large relevance is the work of Dorrington et al. [43], in which the authors proposed a numerical solution that can be employed in off-the-shelf ToF cameras. Shortly after, Godbaz et al. [70] proposed two closed-form solutions to the problem. These two works assume, however, that there are two return components per pixel, and work with two or up to four modulation frequencies. This two-component, dual-frequency approach was generalized by Bhandari et al. [18]. Kirmani et al. [147] targeted simultaneously phase unwrapping and multipath interference cancellation, using a higher number of frequencies (five or more), but at a lower computational cost than previous approaches, thanks to a closed form solution. Still, they assumed sparsity in the recovered signal, and again restricted their model to two-bounce situations.

The use of multiple modulation frequencies was also leveraged by Heide and colleagues [92]. In their case, they used hundreds of modulation frequencies, and proposed a model that includes global illumination. Freedman et al. [56] also required multiple frequencies, and proposed a model (not limited to two bounces) which assumes compressibility of the time profile; they solved the problem iteratively via $\mathcal{L}_1$ optimization. Kadambi et al. [130] reduced the number of frequencies required to recover a time profile (and thus depth information) to one, by using custom codes in the emission in combination with sparse deconvolution techniques, to recover the time profiles as a sparse set of Dirac deltas. This technique allowed to recover depth in the presence of interreflections, including transparent objects (Figure 4.2).

All these works assumed a K-sparse transport model[3]. It is worth noting, however, that in the case of scattering media being present, a sparse formulation of the time profile is no longer possible. A slightly different approach was taken by Jiménez et al. [124], who proposed an optimization framework to minimize the difference between the measured depth, and the depth obtained by their radiometric model. Convergence to a global minimum was not guaranteed, but a number of examples including real scenes were shown. Hardware modifications are not required.

A different means of eliminating or separating global light transport in a scene was presented by O'Toole et al. [206], who made the key observation that transient light transport is separable in the temporal frequency domain. This allowed them to acquire and process only the direct time-of-flight component, by using a projector with light modulated in space and time (note that they do not use correlation-based ToF). Gupta et al. [77] built on this idea, and proposed a framework termed *phasor imaging*. A key observation is that global effects vanish for frequencies higher than a certain, scene-dependent, threshold; this allowed the authors to recover depth in the presence of MPI, as well as to perform direct/global separation, using correlation-based time-of-flight sensors. Neither Gupta et al.'s work, nor O'Toole et al.'s, imposed the restriction of sparsity of the multipath profile. Neither did Naik et al. [197], who also attempted direct/global separation to obtain correct depth in the presence of MPI. A similar approach was followed by Whyte et al. [272].

[3]Please refer to the full article [113] for a complete review on these models.

5 A FRAMEWORK FOR TRANSIENT RENDERING

Efficient simulation of time-resolved light transport is of key importance in many applications within transient imaging. Motivated by this, this chapter presents a formalized framework for simulating light transport in transient state, and identifies and addresses the main problems that arise when accounting for light propagation time in simulation. For that purpose we introduce the *transient path integral* framework. Under this framework, we address variance issues by proposing a progressive technique based on density estimation to reconstruct temporal profiles of radiance. Additionally, we introduce several sampling methods for participating media to achieve uniform distributions of samples in the temporal domain.

This work was published in ACM Transactions on Graphics and presented at SIGGRAPH Asia 2014. Many of the final formulas provided in this chapter are fully derived in the supplemental document associated to this publication. Reference to this document and its corresponding sections are placed throughout the chapter when appropriate. My role as a second author of this work was the investigation and implementation of time-resolved density estimation algorithms in participating media, and the analysis of time-resolved light transport under different media configurations.

A. Jarabo, J. Marco, A. Munoz, R. Buisan, W. Jarosz & D. Gutierrez
A Framework for Transient Rendering
ACM Transactions on Graphics, Vol.33(6)
(Presented at SIGGRAPH Asia 2014)

5.1 INTRODUCTION

One of the most general assumptions in computer graphics is to consider the speed of light to be infinite, leading to the simulation of light transport in steady state. This is a reasonable assumption, since most of the existing imaging hardware is very slow compared to the speed of light. Light transport in steady state has been extensively investigated in computer graphics (e.g. Dutré et al. [47], Gutierrez et al. [81], Křivánek et al. [156]), including for instance the gradient [120, 220] or frequency [46] domains. In contrast, work in the temporal domain has been mainly limited to simulating motion blur [125] or time-of-flight imaging [151].

In this work we introduce a formal framework for *transient rendering*, where we lift the assumption of an infinite speed of light. While different works have looked into transient rendering [5, 108, 239], they have approached the problem by proposing straight forward extensions of traditional steady-state algorithms, which are not adequate for *efficient* transient rendering for a variety of reasons. Firstly, the addition of the extra sampling domain given by the temporal dimension dramatically increases the convergence time of steady state rendering algorithms. Moreover, by extending the well-accepted path integral formulation [254], we observe that paths contributing to each frame form a near-delta manifold *in time*, which makes sampling almost impossible. We solve this issue by devising new sampling strategies that improve the distribution of samples *along the temporal domain*, and a new density estimation technique that allows reconstructing the signal along time from such samples.

Our work presents valuable insight apart from rendering applications. Recent advances in time-resolved imaging are starting to provide novel solutions to open problems, such as reconstructing hidden geometry [257] or BRDFs [195], recovering depth of transparent objects [130], or even visualizing the propagation of light [259]. Despite these breakthroughs in technology, there is currently a lack of tools to efficiently simulate and analyze transient light transport. This would not only be beneficial for the graphics and vision communities, but it could open up a novel analysis-by-synthesis approach for applications in fields like optical imaging, material engineering or biomedicine as well. In addition, our framework can become instrumental in teaching the complexities of light transport [120], as well as visualizing in detail some of its most cumbersome aspects, such as the formation of caustics, birefringence, or the temporal evolution of chromatic dispersion.

In particular, in this work we make the following contributions:

- Establishing a theoretical framework for rendering in transient state, based on the path integral formulation and including propagation in free space as well as scattering on both surfaces and in media. This allows us to analyze the main challenges in transient rendering.
- Developing a progressive kernel-based density estimation technique for path reuse that significantly improves the reconstruction of time-resolved radiance.
- Devising new sampling techniques for participating media to uniformly sample in the temporal domain, that complement traditional radiance-based sampling.
- Providing time-resolved simulations of several light transport phenomena which are impossible to see in steady state.

5.2 RELATED WORK

TRANSIENT RADIATIVE TRANSFER. With advances in laser technology, capable of producing pulses of light in the order of a few femtoseconds, *transient* radiative transfer gained relevance in fields like optical imaging, material engineering or biomedicine. Many numerical strategies have been proposed, including Monte Carlo simulations, discrete ordinate methods, integral equation models or finite volume methods [188, 289, 291]. Often, these methods are applied on simplified scenarios with a particular application in mind, but a generalized framework has not yet been adopted.

ULTRA-FAST IMAGING. Several recent advances in ultra-fast imaging have found direct applications in computer graphics and vision. Raskar and Davis [222] introduce the basic theoretical framework in light transport analysis that would later lead to a number of practical applications, such as reconstruction of hidden geometry [146, 257] or reflectance acquisition [195]. Velten et al. [258, 259] have recently presented *femto-photography*, a technique that allows capturing time-resolved videos with an effective exposure time of one picosecond per frame, using a streak camera. Heide et al. [92] later propose a cheaper setup using Photonic Mixing Devices (PMDs), while sacrificing temporal and spatial resolution. Kadambi and colleagues [130] address multi path interference in time-of-flight sensors by recovering time profiles as a sequence of impulses, allowing them to recover depth from transparent objects.

ANALYSIS OF TIME-RESOLVED LIGHT TRANSPORT. Wu et al. [277] analyze the propagation of light in the frequency domain, and show how the cross-dimensional transfer of information between the temporal and frequency domains can be applied to bare-sensor imaging. Later, Wu et al. [276] used time-of-flight imaging to approximately decompose light transport into its different components of direct, indirect and subsurface illumination, by observing the temporal profiles at each pixel. Lin and colleagues [173] perform a frequency-domain analysis of multifrequency time-of-flight cameras. Recently, O'Toole and colleagues [206] derived transient light transport as a linear operator, as opposed to our formulation in ray space, and showed how to combine the generation and acquisition of transient light transport for scene analysis. In this regard, our work can be seen as complementary: we provide a simulation (rendering) framework, suitable for an analysis-by-synthesis approach to exploring novel ideas and applications, and to help better understand the mechanisms of light transport.

TRANSIENT RENDERING. The term *transient rendering* was first coined by Smith et al. [239]. In their work, the authors generalize the rendering equation as a recursive operator including propagation of light at finite speed. The model provides a solid theoretical background for time-of-flight, computer vision applications, but does not provide a practical framework for transient rendering of global illumination. Keller et al. [142] develop a time-of-flight sensor simulation, modeling the behavior of PMDs. These works are again geared towards time-of-flight applications; moreover, they are limited to surfaces, not taking into account the presence of participating media. Simulation of relativistic effects [110, 269] could also potentially benefit from our transient rendering framework.

Some recent works in computer graphics make use of transient state information: d'Eon and Irving [38] quantize light propagation into a set of states, and model the transient state at each instant using Gaussians with variance proportional to time. These Gaussians are then integrated into the final image. The wave-based approach by Musbach et al. [192] uses the Finite Difference Time Domain (FDTD) method to obtain a solution for Maxwell's equations, rendering complex effects like diffraction. In all these cases, however, the main goal is to render steady state images, not to analyze the propagation of light itself. Jarabo [108] showed transient rendering results based on photon mapping and time-dependent density estimation, but limited to surfaces in the absence of participating media. Last, Ament et al. [5] include time into the Radiative Transfer Equation in order to account for a continuously-varying indices of refraction in participating media, though they do not introduce efficient techniques for transient rendering.

ACOUSTIC RENDERING. Our work is somewhat related to the field of acoustic rendering [60]. Traditional light rendering techniques have been adapted to sound rendering, such as photon (*phonon*) mapping [16] or precomputed *acoustic* radiance transfer [6]. Closest to our approach, the work by Siltanen et al. [236] extends the radiosity method to include propagation delays due to the finite, though much slower, speed of sound. As opposed to us, they use finite elements methods to compute sound transport, do not handle participating media and do not propose sampling techniques for uniform temporal sample distribution.

5.3 TRANSIENT PATH INTEGRAL FRAMEWORK

We first extend the standard path integral formulation to transient state. This will allow us to formalize the notion of transient rendering, understand how to elevate steady state rendering to transient state, and, most importantly, identify the unique challenges of solving this more difficult light transport problem.

In the path integral formulation [254], the image pixel intensity I is computed as an integral over the space of light transport paths Ω. For transient rendering, in addition to integrating over spatial coordinates, we must also integrate over the space of temporal delays ΔT of all paths:

$$I = \int_{\Omega} \int_{\Delta T} f(\overline{\mathbf{x}}, \overline{\Delta \mathbf{t}}) \, \mathrm{d}\mu(\overline{\Delta \mathbf{t}}) \, \mathrm{d}\mu(\overline{\mathbf{x}}), \tag{5.1}$$

where $\overline{\mathbf{x}} = \mathbf{x}_0 \ldots \mathbf{x}_k$ represents the spatial coordinates of the $k+1$ vertices of a length-k path with $k \geq 1$ segments. Vertex $\mathbf{x}_0$ lies on a light source, $\mathbf{x}_k$ lies on the camera sensor, and $\mathbf{x}_1 \ldots \mathbf{x}_{k-1}$ are intermediate scattering vertices. The differential measure $\mathrm{d}\mu(\overline{\mathbf{x}})$ denotes area integration for surfaces vertices and volume integration for media vertices. $\overline{\Delta \mathbf{t}} = \Delta t_0 \ldots \Delta t_k$ defines a sequence of time delays and $\mathrm{d}\mu(\overline{\Delta \mathbf{t}})$ denotes temporal integration at each path vertex.

We define the path contribution function $f(\overline{\mathbf{x}}, \overline{\Delta \mathbf{t}})$ as the original, but with the emission L_e, path throughput $\mathfrak{T}$, and sensor importance W_e additionally depending on time:

$$f(\overline{\mathbf{x}}, \overline{\Delta \mathbf{t}}) = L_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1, \Delta t_0) \, \mathfrak{T}(\overline{\mathbf{x}}, \overline{\Delta \mathbf{t}}) \, W_\mathrm{e}(\mathbf{x}_{k-1} \to \mathbf{x}_k, \Delta t_k). \tag{5.2}$$

The temporal sensor importance W_e now defines not only the spatial and angular sensitivity, but also the region of time we are interested in evaluating. This could specify a delta function at a desired time, or more commonly, a finite interval of interest in the temporal domain (analogous to the shutter interval in steady state rendering, though at much smaller time scales). Likewise, the time parameter of the emission function L_e can define temporal variation in emission (e.g. pulses). The transient path throughput is now defined as:

$$\mathfrak{T}(\overline{\mathbf{x}}, \overline{\Delta \mathbf{t}}) = \left[\prod_{i=1}^{k-1} \rho(\mathbf{x}_i, \Delta t_i) \right] \left[\prod_{i=0}^{k-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}) V(\mathbf{x}_i, \mathbf{x}_{i+1}) \right]. \tag{5.3}$$

Since we assume that the geometry is stationary (relative to the speed of light), the geometry and visibility terms depend only on the spatial coordinates of the path, as in steady state rendering. However, we extend the scattering kernel ρ with a temporal delay parameter Δt_i to account for potential time delays at each scattering vertex $\mathbf{x}_i$. Such delays can occur due to e.g. multiple internal reflections within micro-geometry [270], electromagnetic phase shifts in the Fresnel equations [71, 226], or inelastic scattering effects such as fluorescence [81, 273].

TIME DELAYS. A transient light path is defined in terms of spatial and temporal coordinates. The temporal coordinates at each path vertex $\mathbf{x}_i$ are t_i^-, the time immediately before the scattering event, and t_i, the

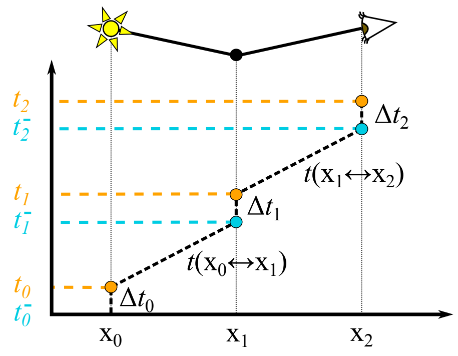


Figure 5.1: Spatio-temporal diagram of light propagation for a path with $k = 2$. Light is emitted at time t_0, and reaches $\mathbf{x}_1$ at $t_0 + t(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)$. After a microscopic temporal delay Δt_1, light emerges from $\mathbf{x}_1$ at t_1 and takes $t(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$ time to reach $\mathbf{x}_2$. The sensor may include a further temporal delay Δt_2.

time immediately after (see Figure 5.1). Both time coordinates can be obtained by accounting for all *propagation* delays between vertices $t(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$ and *scattering* delays Δt_i at vertices along the path:

$$t_i^- = \sum_{j=0}^{i-1} \left(t(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) + \Delta t_j \right), \quad t_i = t_i^- + \Delta t_i, \tag{5.4}$$

where t_0 and t_k denote the emission and detection times of a light path. The transient simulation is assumed to start at $t_0^- = 0$. In the general case of non-linear media [5, 80, 104], propagation time along a path segment is:

$$t(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) = \int_{s_j}^{s_{j+1}} \frac{\eta(\mathbf{x}_r)}{c} \, \mathrm{d}r, \tag{5.5}$$

where r parametrizes the path of light between the two points, s_j and s_{j+1} are the parameters of the path at $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$, respectively, c is the speed of light in vacuum and $\eta(\mathbf{x}_r)$ represents the index of refraction of the medium at $\mathbf{x}_r$. In the typical scenario where η is constant along a path segment, Equation (5.5) reduces to a simple multiplication: $t(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) = \|\mathbf{x}_j - \mathbf{x}_{j+1}\| \eta / c$. Figure 5.1 illustrates both the spatial and temporal dimensions of a path for the case of $k = 2$.

NUMERICAL INTEGRATION. Similar to its steady state counterpart, the the transient path integral (5.1) can be numerically approximated using a Monte Carlo estimator:

$$\langle I \rangle = \frac{1}{n} \sum_{j=1}^{n} \frac{f(\overline{\mathbf{x}}_j, \overline{\Delta \mathbf{t}}_j)}{p(\overline{\mathbf{x}}_j, \overline{\Delta \mathbf{t}}_j)}, \tag{5.6}$$

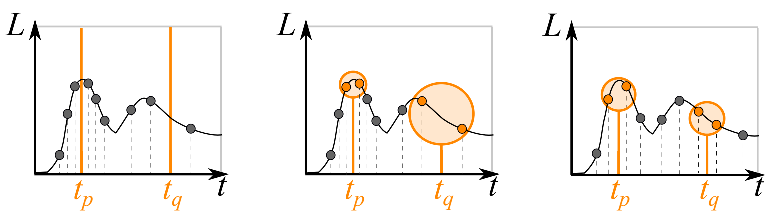


Figure 5.2: **Left:** The probability of finding a sample at a specific time instant (t_p or t_q) is nearly zero (Section 5.3). **Middle:** Density estimation on the temporal domain (Section 5.4) allows us to reconstruct radiance at any instant, although with varying bias and variance in time. **Right:** A more uniform distribution of samples in the temporal domain leads to more uniform bias and better reconstructions (Section 5.5).

which averages n random paths $\overline{\mathbf{x}}_j, \overline{\Delta\mathbf{t}}_j$ drawn from a spatio-temporal probability distribution (pdf) $p(\overline{\mathbf{x}}_i, \overline{\Delta\mathbf{t}}_i)$ defined by the chosen path and time delay sampling strategy. In steady state, the pdf only needs to deal with the location of path vertices $\overline{\mathbf{x}}_i$.

5.3.1 *Challenges of sampling in transient state*

Equation (5.1) shows a new domain of scattering delays ΔT that must be sampled. Most existing path sampling techniques generate random paths incrementally, vertex-by-vertex, by locally importance sampling the scattering function ρ at each bounce, and optionally making deterministic shadow connections between light and camera subpaths. We could in principle elevate any such algorithm to transient state by simply sampling the transient scattering function $\rho(\mathbf{x}_i, \Delta t_i)$, instead of the steady state scattering function $\rho(\mathbf{x}_i)$.

Unfortunately, transient rendering poses hidden challenges, since *propagation* delays between vertices $t(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$ are fundamentally different than *scattering* delays Δt_i defined at the light, sensor, and interior vertices. While scattering delays reside on a separate sampling domain ΔT, propagation delays are a direct consequence of the spatial positions of path vertices sampled from Ω. Hence, if spatial positions are determined by a steady state sampling routine ignorant of propagation delays, control of the propagation time in a path's total duration t_k is lost, leaving only the scattering delays Δt_i to control t_k.

Other factors resulting from the temporal structure of light transport make any naïve extension to transient rendering extremely inefficient: to visualize transient effects, the time window of both the sensor and the light source needs to be small (≈ 10 *pico*seconds); moreover, scattering events result in *femto*second temporal delays. The temporal domain of the path contribution thus becomes a near delta manifold (i.e. a *caustic* in time), which is virtually impossible to sample by random chance. Since the total path duration t_k cannot be directly controlled, deterministic shadow connections are rendered useless, having little chance of finding a non-zero contribution in both the light L_e and the sensor W_e. In general, the probability of randomly finding non-zero contribution for a specific time decreases as either Δt_i, L_e or W_e get closer to delta functions in the temporal domain, which are precisely the cases of interest in transient light transport.

When several distinct measurements of the path integral have to be computed, a common optimization strategy is to share randomly sampled paths to estimate all measurements simultaneously. This technique (path reuse) is utilized in the spatial domain in light tracing and bidirectional path tracing to estimate all pixels in the image plane at once. A similar situation occurs in the transient domain, where each frame f defines a specific sensor importance function $W_\mathrm{e}^f(\mathbf{x}_{k-1} \to \mathbf{x}_k, t_k)$ and the time window covered by all frames is significantly larger than the per-frame time window. We could therefore leverage temporal path reuse to improve the efficiency of steady state path sampling methods when applied to rendering transient light transport. In practice, for every generated random path in Equation (5.6), we could evaluate the contribution functions for every frame f, which differ only in the temporal window of the sensor importance function W_e^f.

This path reuse technique is equivalent to histogram density estimation [237] in the *temporal domain* of the sensor, where each bin of the histogram represents one frame, and the bin's width h is the frame duration. Unfortunately, this type of density estimation produces very noisy results, especially for bins with very small width (i.e. exposure time). This results in a low convergence rate of $O(n^{-1/3})$ [231], where n is the number of samples. This is illustrated in Figure 5.3: although obviously better than not reusing paths, results are still extremely noise even with a large amount of samples. Still, this suggests that more elaborated density estimation techniques may lead to better convergence rates and/or less noisy reconstructions.

In the following, we first show how kernel-based density estimation techniques in the temporal domain allow us to reconstruct radiance along time from a sparse set of samples (see Section 5.4 and Figure 5.2, middle). Then, we show how a skewed temporal sample distribution affects radiance reconstruction, and develop a set of sampling strategies for participating media that enable some control over propagation delays, leading to a more uniform distribution of samples in time and therefore more accuracy (see Section 5.5 and Figure 5.2, right).

5.4 KERNEL-BASED TEMPORAL DENSITY ESTIMATION

Kernel-based density estimation is a widely known statistical tool to reconstruct a signal from randomly sampled values. These techniques

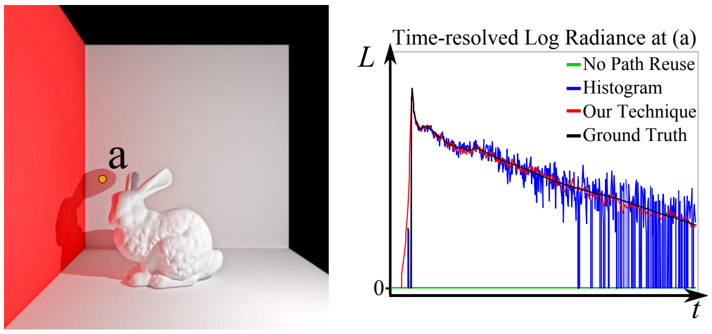


Figure 5.3: Time-resolved irradiance computed at pixel (a) in the scene on the left using, no path reuse (green), histogram-based path reuse (red), and kernel-based path reuse (blue), for the same number of samples. Without path reuse it is extremely difficult to reconstruct the radiance, since the probability of finding a path arriving at the specific frame is close to zero. This is solved using path reuse, although with different levels of improvement: while histogram-based density estimation shows a very noisy result, our proposed progressive kernel-based estimation shows a solution with significantly lower variance, while preserving high-frequency features due to the progressive approach.

significantly outperform histogram-based techniques (like the path reuse described above), especially for noisy data [237]. A kernel with finite bandwidth is used to obtain an estimate of the value of a signal at a given point by computing a weighted average of the set of random samples around such point. We thus introduce a temporal kernel $K_{\mathcal{T}}$ with bandwidth $\mathcal{T}$ to estimate incoming radiance I at the sensor at time t as a function of n samples of I:

$$\langle I_n \rangle = \frac{1}{n} \sum_{j=1}^{n} K_{\mathcal{T}}(\|t - t_{k,j}\|)\widehat{I_j}, \tag{5.7}$$

where $\widehat{I_j} = f(\overline{\mathbf{x}}_j, \overline{\Delta\mathbf{t}}_j)/p(\overline{\mathbf{x}}_j, \overline{\Delta\mathbf{t}}_j)$ is the contribution of path $\overline{\mathbf{x}}_j$ in the measured pixel, and $t_{k,j}$ is the total time of the path (5.4). Using this temporal density estimation kernel reduces variance, but at the cost of introducing bias (see Figure 5.2, middle). This can be solved by using consistent progressive approximations [83, 149], which converge to the correct solution in the limit.

Inspired by these works, we model our progressive density estimation along the temporal domain, for which we rely on the probabilistic approach for progressive photon mapping used by Knaus and Zwicker [149]. We compute the estimate $\langle I_n \rangle$ in n steps, progressively reducing bias while allowing variance to increase; this is done by reducing the kernel bandwidth $\mathcal{T}$ in each iteration as $\mathcal{T}_{j+1}/\mathcal{T}_j = (j+\alpha)/(j+1)$. The variance of our temporal progressive estimator vanishes with

$O(n^{-\alpha})$ as expected, since the shrinking ratio is inversely proportional to the variance increase factor. Bias, on the other hand, vanishes with $O(n^{-2(1-\alpha)})$. Note that the parameter α defines the convergence of *both* sources of error (bias and variance). To find the optimal value that minimizes both, we use the *asymptotic mean square error* (AMSE), defined as:

$$\mathrm{AMSE}(\langle I_n \rangle) = \mathrm{Var}[I_n] + \mathrm{E}[\epsilon_n]^2. \tag{5.8}$$

Using the convergence rate for both bias and variance, we find that the optimal α that minimizes the AMSE is $\alpha = 4/5$, which leads to a convergence of $O(n^{-4/5})$. This is significantly faster than using the histogram method, $O(n^{-1/3})$, which we illustrate in Figure 5.3. The detailed derivation of the behavior of the algorithm can be found in the Section B of the supplemental document[1].

5.4.1 *Transient progressive photon mapping*

Our approach above is agnostic to the algorithm used to obtain the samples (e.g. samples in Figure 5.3 have been computed using path tracing). This means that it can be combined with biased density estimation-based algorithms such as (progressive) photon mapping [82, 83, 121], which is well suited for complex light paths such as spatial caustics. However, although using progressive photon mapping as the source of samples for our temporal density estimation is consistent in the limit, it results in suboptimal convergence due to the coupling of the bias and variance between the spatial and temporal kernels. Instead, we introduce the temporal domain into the photon mapping framework, by adding the temporal smoothing kernel $K_{\mathcal{T}}$ in the radiance estimation [26]. Radiance $\widehat{L_o}(\mathbf{x}, t)$ is estimated using M photons with contribution γ_i as

$$\widehat{L_o}(\mathbf{x}, t) = \frac{1}{M} \sum_{i=1}^{M} K(\|\mathbf{x} - \mathbf{x}_i\|, \|t - t_i^-\|)\gamma_i. \tag{5.9}$$

Combining both kernels into a single multivariate kernel allows controlling the variance increment in each step as a function of a single α, so that it increments at a rate of $(j+1)/(j+\alpha)$, while reducing bias by progressively shrinking both the spatial and temporal kernel bandwidths (R and $\mathcal{T}$ respectively). These are reduced at each iteration j following:

$$\frac{\mathcal{T}_{j+1}}{\mathcal{T}_j} = \left(\frac{j+\alpha}{j+1}\right)^{\beta_{\mathcal{T}}}, \qquad \frac{R_{j+1}^2}{R_j^2} = \left(\frac{j+\alpha}{j+1}\right)^{\beta_R}, \tag{5.10}$$

[1] http://giga.cps.unizar.es/~ajarabo/pubs/transientSIGA14/downloads/Jarabo_siga14_Supplementary_Material.pdf

where $\beta_{\mathcal{T}}$ and β_R are scalars in the range $[0, 1]$ controling how much each term is to be scaled separately, with $\beta_{\mathcal{T}} + \beta_R = 1$. Please refer to Section C of the supplemental document [1] for the complete derivation. The convergence rate of the *combined* spatio-temporal density estimation is $O(n^{-4/7})$[2]. Using this formulation allows us to handle complex light paths in transient state, while still progressively reducing bias and variance introduced by both progressive photon mapping and our temporal density estimation, in the spatial and temporal domains respectively. In Section C of the supplementary material[1] can be found a detailed description of the algorithm, including the full derivation of the error and convergence rate.

5.5 TIME SAMPLING IN PARTICIPATING MEDIA

As we mentioned earlier (Section 5.3.1), the performance of our transient density estimation techniques can be further improved by a more uniform distribution of samples in time. This makes the relative error uniform in time and optimizes convergence (see Figure 5.2, right). Steady state sampling strategies aim to approximate radiance (path contribution). Since more radiant samples happen at earlier times (due to light attenuation), these sampling techniques skew the number of samples towards earlier times. As a consequence, there is a increase of error along time (see Figures 5.7 and 5.8). New sampling strategies are therefore needed for transient rendering.

Sampling strategies over scattering delays Δt_i have a negligible influence over the total path duration t_k (Figure 5.1). For surface rendering, scattering delays are the only control that sampling strategies can have on the temporal distribution of samples, and there is therefore little control over the total path duration. In participating media, however, sample points can be potentially located anywhere along the path of light, providing direct control also over the propagation times $t(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$. In this section we develop new sampling strategies for participating media that target a uniform sample distribution in the time domain, by customizing:

- The pdf for each segment of the camera or light subpath (Section 5.5.1).
- The pdf for a shadow connection (connecting a vertex of the camera path to a vertex of the light path) via an additional vertex (Section 5.5.2).
- The pdf in the angular domain to obtain the direction towards the next interaction (Section 5.5.3).

[2]Note that a naïve combination of the temporal (1D) and the spatial (2D) kernels would yield a slower convergence than the combined 3D kernel convergence $O(n^{-4/7})$ when using the optimal parameters $\alpha = 4/7$ and $\beta_{\mathcal{T}} = 1/3$ reported in previous work [137] (for volumetric density estimation) or in the statistics literature [231].

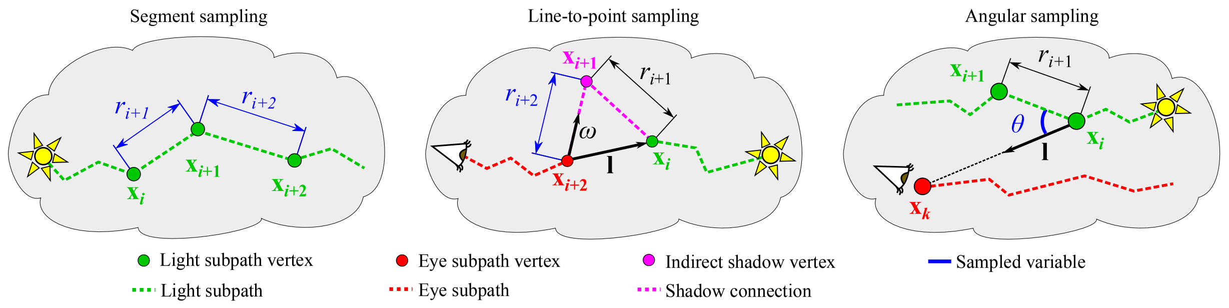


Figure 5.4: Sampling strategies for participating media with a uniform distribution in the time domain. **Left:** Sampling scattering distance for a light subpath. This strategy can also be applied to eye subpaths. **Middle**: Sampling shadow connections through a new indirect vertex: line-to-point sampling of shadow connections. **Right:** Sampling the angular scattering function (phase function).

Each of these sampling strategies ensures a uniform distribution of samples in time for each particular domain of the full path. Although this does not statistically ensure uniformity for the whole path, in practice the resulting distribution of total path duration t_k samples in time is close to uniform and therefore noise is reduced (the improvement over steady state strategies is discussed in Section 5.6). Note that these strategies are also agnostic of the properties of the media (except for the index of refraction), and can therefore be used in arbitrary participating media. Additionally, they can be combined with steady-state radiance sampling via multiple importance sampling (MIS) [255].

5.5.1 *Sampling scattering distance in eye/light subpaths*

Each of the segments of a subpath in participating media often shares the same steady-state sampling strategy, such as mean-free-path sampling, which does not necessarily ensure a uniform distribution of temporal location of vertices. We aim to find a pdf $p(r)$ (where r is the scattering distance along one of the subpath segments) so that the probability distribution $p\left(\cup_{i=1}^{\infty} t_i\right)$ of temporal subpath vertex locations is uniform (see Figure 5.4, left). We first define $p\left(\cup_{i=1}^{\infty} t_i\right)$ based on the combined probability distribution $p(t_i)$ (temporal location of vertex $\mathbf{x}_i$ in the light subpath) for all subpath vertices:

$$p\left(\cup_{i=1}^{\infty} t_i\right) = \sum_{i=1}^{\infty} p(t_i), \tag{5.11}$$

where $p(t_i)$ is recursively defined based on $p(t_{i-1})$. Given that $t_i = t(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) + t_{i-1}$, as shown in Equation (5.4), we have

$$p(t_i) = \int_0^{t_i} p\big(t(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)\big) p(t_{i-1}) dt_{i-1}, \tag{5.12}$$

$$p(t_1) = p\big(t(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\big), \tag{5.13}$$

since the probability of the addition of two random variables is the convolution of their probability distributions. $p\left(t(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1})\right)$ is the probability distribution of the propagation time, which is related to the scattering distance pdf $p(r)$ by a simple change of variable $r = \frac{c}{\eta} t(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)$. Note that, in this notation, we are assuming (as previously discussed) that scattering delays Δt_i are negligible compared to propagation time. This definition is analogous for the eye subpath.

We show (see supplementary material[1], Section D.1) that the exponential distribution $p(r) = \lambda e^{-\lambda r}$ ensures that $p\left(\cup_{i=1}^{\infty} t_i\right)$ follows a uniform distribution for any λ parameter. Figure 5.5 (left) experimentally shows that this exponential distribution leads to this uniform probability for the whole subpath, while a uniform pdf leads to a non-uniform temporal sample distribution. In practice, λ modulates the average number of segments of the subpath: for a path ending at time t_e, the average number of segments with path duration $t_k \leq t_e$ is $\lambda \frac{c}{\eta} t_e$. Our results show that an average of three or four vertices per subpath gives a good compromise between path length, efficiency and lack of correlation. Note that mean-free-path sampling is also an exponential distribution whose rate equals the extinction coefficient of the medium ($\lambda = \sigma_t$). Directly using mean-free-path sampling is thus optimal for time sampling when σ_t is close to the optimal λ.

SUBPATH TERMINATION. Russian roulette is a common strategy in steady state rendering algorithms. It probabilistically terminates subpaths at each scattering interaction, reducing longer paths with a small radiance contribution. In transient state, this unfortunately translates into fewer samples as time advances, reducing the signal-to-noise ratio (SNR) at higher frames. Instead, we simply terminate paths with a total duration greater than the established time frame.

While the temporal locations of subpath vertices are uniform, there is still little control over the spatial locations $\mathbf{x}_i$. These depend not only on scattering distances but also on scattering angles. As shadow rays are deterministic and depend on such spatial locations, uniformity cannot be ensured. To address this, we develop a new strategy that deals with such shadow connections (Section 5.5.2) and an angular sampling strategy (Section 5.5.3) that leads to an improved distribution in the temporal domain of the location-dependent propagation delays.

5.5.2 *Sampling line-to-point shadow connections*

Shadow rays are deterministic segments connecting a vertex in the eye subpath to another vertex in the light subpath, so their duration cannot be controlled. We introduce a new *indirect* shadow vertex whose position can be stochastically set to ensure a uniform sample distribution along the duration of the (extended) shadow connection. The geometry of this indirect connection is similar to equiangular sampling [135, 158, 224] (see Figure 5.4, middle).

Given a vertex $\mathbf{x}_i$ of a light subpath, a vertex $\mathbf{x}_{i+2}$ and a direction ω (importance sampled from the scattering function) on an eye subpath, our technique connects the two vertices via an indirect bounce at an importance-sampled location $\mathbf{x}_{i+1}$. If r_{i+1} and r_{i+2} are the distances from $\mathbf{x}_{i+1}$ to $\mathbf{x}_i$ and $\mathbf{x}_{i+2}$ respectively, we importance sample r_{i+2} to enforce a uniform propagation time between the connected vertices $\{\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}\}$. This connection could also be done in reverse order (from $\mathbf{x}_{i+2}$ to $\mathbf{x}_i$).

Given $\mathbf{l} = \mathbf{x}_i - \mathbf{x}_{i+2}$ and a connection time range (t_a, t_b) (in which we aim to get uniformly distributed samples), the pdf is:

$$p(r_{i+2}) = \frac{\eta}{c(t_b - t_a)}\left[1 + \frac{r_{i+2} - (\mathbf{l}\cdot\omega)}{\sqrt{r_{i+2}^2 - 2r_{i+2}(\mathbf{l}\cdot\omega) + (\mathbf{l}\cdot\mathbf{l})}}\right], \qquad (5.14)$$

which leads to the following inverse cumulative distribution function (cdf):

$$r_{i+2}(\xi) = \frac{(\xi(t_b - t_a) + t_a - t_i - \Delta t_{i+1})^2 - \left(\frac{\eta}{c}\right)^2(\mathbf{l}\cdot\mathbf{l})}{2\frac{\eta}{c}(\xi(t_b - t_a) + t_a - t_i - \Delta t_{i+1}) - 2\left(\frac{\eta}{c}\right)^2(\mathbf{l}\cdot\omega)}. \qquad (5.15)$$

where $\xi \in [0, 1)$ is a random number. Assuming a rendered temporal range of $(0, t_e)$, we set the shadow connection limits to $t_a = t_i + t(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+2})$ and $t_b = t_e - \Delta t_k - \left(\sum_{j=i+2}^{k-1} t(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}) + \Delta t_j\right)$. The derivation of this pdf can be found in the supplementary material[1], Section D.2. Figure 5.5 (middle) compares our line-to-point sampling strategy with other common strategies in terms of sample distribution along the temporal domain, leading to a uniform distribution of samples. Note that we discard all paths with a total duration larger than t_e (when $t_b < t_a$).

5.5.3 *Angular sampling*

Importance sampling the phase function generally leads again to a suboptimal distribution of samples in time. We propose a new angular pdf $p(\theta)$ to be applied at each interaction of the light subpath, which targets the temporal distribution of samples assuming that the next vertex $\mathbf{x}_{i+1}$ casts a deterministic shadow ray towards the sensor. Given the sensor vertex $\mathbf{x}_k$ and a sampled distance r_{i+1} between two consecutive vertices $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ (see Figure 5.4, right), this strategy ensures a uniform distribution of the total propagation time in $\{\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_k\}$. The direction from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ is $\omega = (\theta, \phi)$ (in spherical coordinates) where θ is the sampled angle and ϕ is uniformly sampled in $[0..2\pi)$. Note that the sampled angle θ is related to the direction towards the sensor ($\mathbf{l} = \mathbf{x}_k - \mathbf{x}_i$) and not to the incoming direction (which is often the system of reference for phase function importance sampling). This pdf is:

$$p(\theta) = \frac{r_{i+1}\sin\theta}{2\sqrt{r_{i+1}^2 + |\mathbf{l}|^2 - 2r_{i+1}|\mathbf{l}|\cos\theta}}, \qquad (5.16)$$

with the following inverse cdf:

$$\theta(\xi) = \arccos\left(\frac{|\mathbf{l}| - 2r_{i+1}^2\xi^2 - 2\xi r_{i+1}(|\mathbf{l}| - 1)}{r_{i+1}|\mathbf{l}|}\right). \qquad (5.17)$$

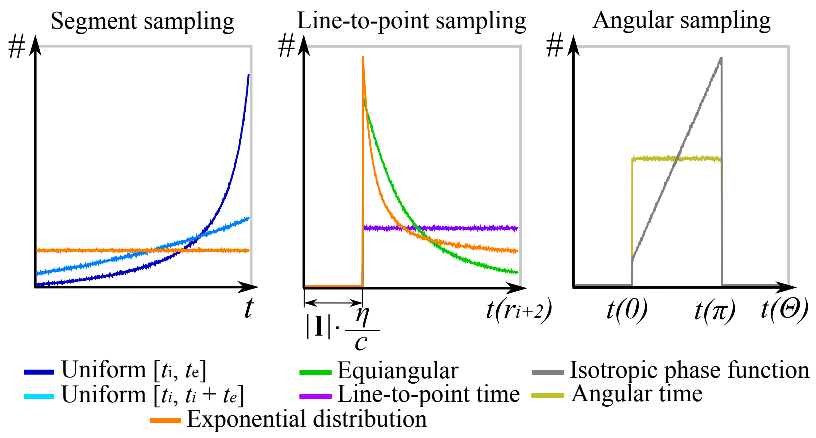


Figure 5.5: Histogram of the number of samples along the temporal dimension for different sampling strategies. **Left:** Sample distribution for the whole light subpath, according to the importance sampling of subpath segments. **Middle:** Importance sampling of a line-to-point shadow connection **Right:** Angular importance sampling. Notice how our developed sampling strategies (exponential for segment sampling and the corresponding time sampling strategies in the other two cases) lead to a uniform distribution of samples along the temporal domain on each case. Both the line-to-point and the angular sampling are defined over a certain range.

The supplementary material[1], Section D.3 contains the full derivation. This pdf prioritizes segments towards the target vertex $\mathbf{x}_t$, which helps in practice since backward directions often lead to paths that become too long for the rendered time frame. Figure 5.5 (right) shows how our angular sampling strategy leads to a uniform distribution of samples in time, as opposed to other alternatives. The shadow ray from vertex $\mathbf{x}_{i+1}$ to the sensor in $\mathbf{x}_k$ (and to every vertex in the eye subpath in bidirectional path tracing) is then cast by applying the sampling technique described in Section 5.5.2. Alternatively, the shadow ray could be cast from $\mathbf{x}_i$ by applying MIS between this angular sampling and line-to-point time sampling (Section 5.5.2). We also apply the same angular sampling strategy for each interaction of the eye subpath, targeting the light source.

5.6 RESULTS

Here we show and discuss our rendered scenes. For visualization we use selected frames of the animations; we refer the reader to the supplementary material[1] for more rendered examples, and to the video[3] for

[3] http://giga.cps.unizar.es/~ajarabo/pubs/transientSIGA14/videos/Jarabo2014_main_video.mp4

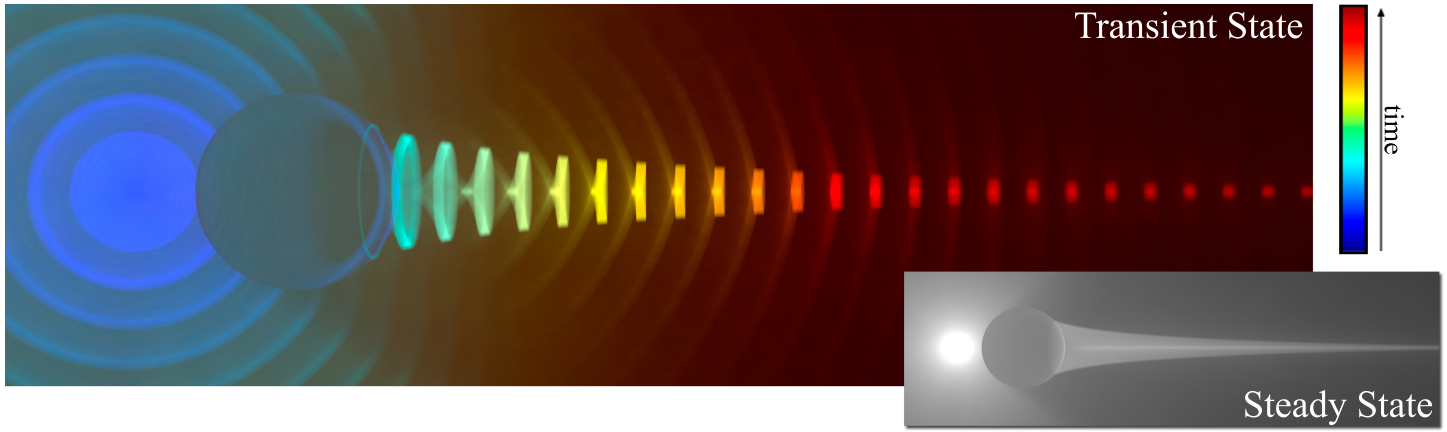


Figure 5.6: Our transient rendering framework allows time-resolved visualizations of light propagation. The caustic wavefront produced by the spherical lens above is distorted by the lens and also delayed due to the longer optical path traversed within the sphere. Note that in this scene we omit the propagation time of the last segment (from the scene to the camera).

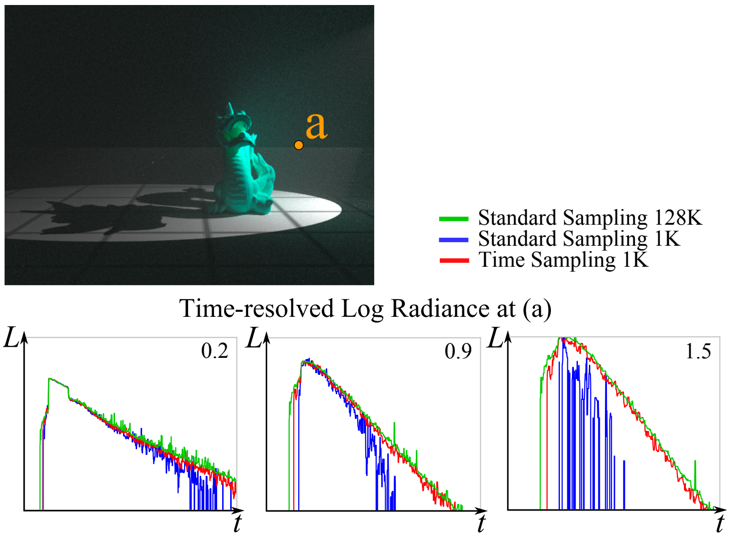


Figure 5.7: Comparison of our three time sampling strategies combined, against the standard techniques used in steady state, in the *dragon* scene accounting for multiple scattering (top). Each graph shows the time-resolved radiance (bottom) at pixel (a), for three different scattering coefficients $\sigma_s = \{0.2, 0.9, 1.5\}$, and absorption $\sigma_a = 0.1$. For 1K samples per pixel and frame, our combined techniques (red) feature a similar quality as standard steady state techniques with 128 times more samples (green), while with the same number of samples, our techniques significantly outperform standard sampling (blue), especially in highly scattering media. To emphasize the differences between sampling techniques, here we use the histogram path reuse (see Section 5.4). Additional results for other types of media can be found in the supplementary material.

the complete animations. In all the scenes light emission occurs at $t = 0$ with a delta pulse[4]. Unless otherwise stated, we use transient path tracing and kernel-based density estimation (Section 5.4) for sampling and reconstruction, respectively. For the latter, we use a Perlin [214] smoothing kernel, following previous work [84, 137], with forty nearest neighbors to determine the initial kernel bandwidth. Unless noted otherwise, all results are shown in camera time [259] (i.e. including the propagation time of the last segment).

Figure 5.7 compares transient rendering using our three time-based sampling strategies (Section 5.5) against common radiance-based steady-

[4]We could use a Gaussian pulse, although this would introduce a number of downsides: 1) an ideal delta pulse does not introduce any additional temporal blur; 2) in reality, the scale of physical Gaussian pulses is 2-3 orders of magnitude smaller than the shutter open interval, in effect constituting a delta pulse; and 3) a delta pulse allows us to distinguish between effects caused by the actual behavior of light and effects due to limitations of current hardware.

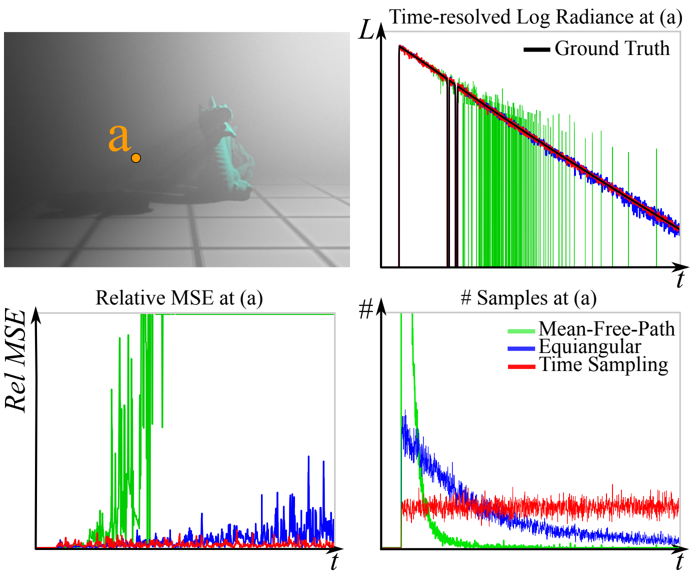


Figure 5.8: Comparison of different sampling techniques for computing single scattering, in a scene consisting of a dragon illuminated by point light source within participating media (left). As opposed to simple *mean-free-path sampling* and the state-of-the-art *equiangular sampling* [158], that distributes samples based on radiance, our point-to-line sampling (Section 5.5.2) distributes samples so that the are uniformly distributed in time (bottom, right). This allows performing better in terms of relative error (bottom, left) when rendering time-resolved radiance, avoiding the radiance signal degradation at longer times. Here we use the histogram (Section 5.4) to emphasize the performance of the algorithms.

state sampling techniques (mean-free-path and phase-function sampling, and deterministic shadow connection). Our approach distributes samples more uniformly in time, which reduces variance along the whole animation, while significantly lowering noise in later frames. We obtain similar quality to standard sampling using two orders of magnitude less samples. These advantages are even more explicit when using our line-to-point sampling strategy to render single scattering, as shown in Figure 5.8, where we compare against equiangular sampling [158]. Figure 5.9 shows how the *combination* of our kernel-based density estimation and our time sampling strategies produces better results than using either technique in isolation.

Figures 5.6 demonstrates the macroscopic delays due to traversing media with different order of refraction, which leads to a temporal delay of the wavefront, especially visible in the caustics. In this example, we use a transient version of the *photon beams* algorithm [118] to obtain the radiance samples due to scattering in the media. To illustrate this

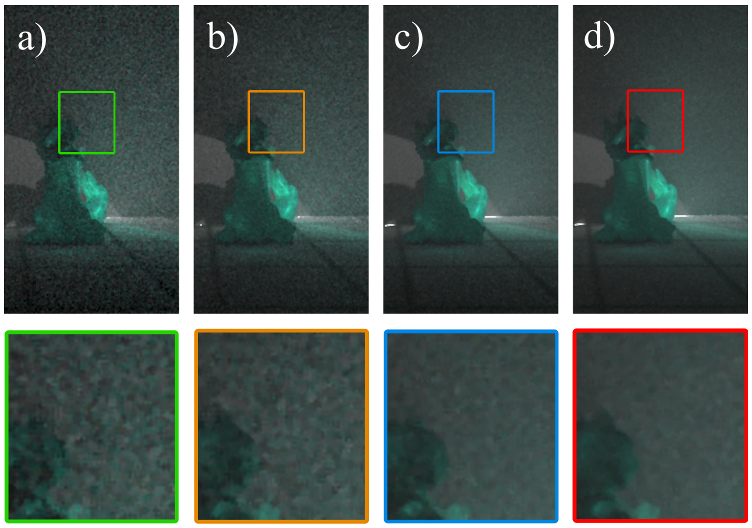


Figure 5.9: Selected frame of the *dragon* scene with $\sigma_s = 0.2$, rendered with a) standard sampling and histogram, b) our time sampling and histogram, c) standard sampling and our kernel-based density estimation, and d) time sampling and kernel-based density estimation. We can see how using our techniques combined lead to frames with significantly lower noise.

in a single image we use the *peak-time* visualization proposed by Velten et al. [259].

Figure 5.10 compares our simulation against a real scene captured with the femtophotography technique of Velten et al. [259]. We can see that our simulation faithfully reproduces the different orders of scattering events occurring during light propagation. Finally, Figure 5.11 shows different examples of non-trivial phenomena visible in transient state, including temporal chromatic dispersion due to wavelength-dependent index of refraction, refraction delays for ordinary and extraordinary rays in birefringent crystals [164, 268] and fluorescence due to energy re-emission after absorption [81]. We refer to the supplementary video [3] for full visualization of the different phenomena.

5.7 DISCUSSION

In summary, we have extended the classical path space integral to include the temporal domain, and shown how the high frequency nature of transient light transport leads to severe sampling problems. We have proposed novel sampling strategies and density estimation techniques, which allow us to distribute samples uniformly in time, resulting in reduced variance and a constant distribution of bias. Our

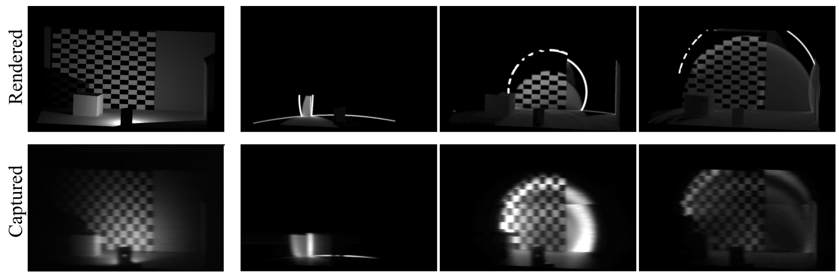


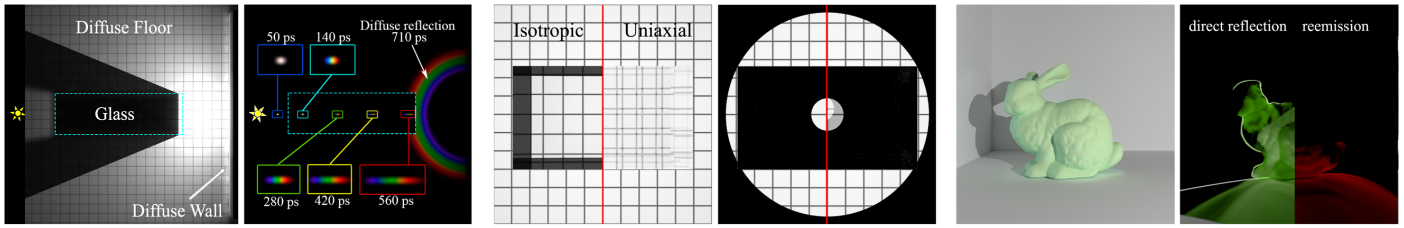
Figure 5.10: Comparison between the *Cube* scene from [259] and our rendered simulation of the same scene. Visible differences are due to approximate materials and camera properties.

supplementary material contains a rigorous mathematical analysis of all our technical contributions. Last, we have presented simulations of interesting transient light transport effects using modified versions of a representative cross section of common rendering algorithms.

Apart from educational benefits, our work could be used to help design prototypes of novel ultra-fast imaging systems, or as a forward model for inverse problems such as recovering hidden geometry or material estimation. Our temporal progressive density estimation (Section 5.4) could also be used to accelerate radiance reconstruction in time-resolved imaging techniques, reducing the need for taking repeated measurements to improve the SNR. Moreover, synthetic ground truth data may become a very valuable tool for designing and benchmarking future ultra-fast imaging devices.

Our time-resolved simulations can help analyze the complex phenomena involved in light transport, and gain new insights. For instance, Figure 5.12 shows how during the early stages of light propagation, the first orders of scattering determine the shape of the light distribution (a spherical wavefront), but over time this shape becomes a Gaussian of increasing variance. This observation is consistent with previous work [286], where it is shown that light in a medium exhibits diffusion after traveling about ten times the mean-free-path, and might explain some of the errors near the light source reported in the quantized diffusion model [38]. This effect is more accentuated in the presence of anisotropic media, where the wavefront behavior is even more dominant.

FUTURE WORK. There are many compelling avenues of future work: First, it would be interesting to extend a unified path sampling framework [157] to transient state. We have shown how the photon beams algorithm [118] can be used in transient rendering, combined with our temporal density estimation; however, a spatio-temporal progressive photon beams framework would be needed to achieve optimal convergence in transient state. Additionally, by building a joint sam-



a) Temporal chromatic dispersion b) Birefringence c) Fluorescence

Figure 5.11: Examples of different phenomena observed in transient state: from left to right, temporal chromatic dispersion due to wavelength dependent index of refraction; ordinary and extraordinary image formation in a birefringent crystal; and energy re-emission in a fluorescent bunny. See the supplementary video for the full animations.

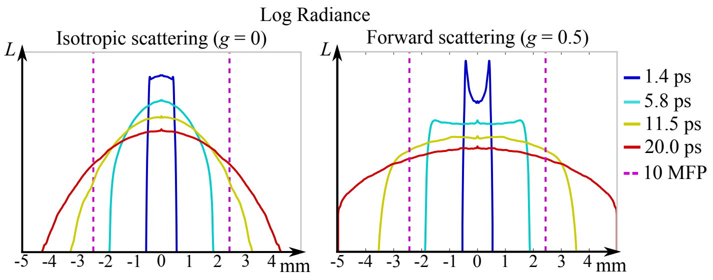


Figure 5.12: Time-resolved light transport from a point light source placed in the middle of an isotropic (left) and forward (right) scattering medium, emitting at time $t = 0$. Both media have a mean free path of 0.244 mm ($\sigma_t = 4.1$ mm^{-1}). In the initial phase the light distribution is dominated by the wavefront shape of the low-order scattering events. In isotropic scattering, light distribution becomes Gaussian after traveling ten times the mean free path. In forward scattering, this distance is increased.

pling strategy in both angle and distance, as in recent advanced steady state sampling techniques [64, 204], we could leverage the benefits of both to ensure better uniformity in the temporal distribution of samples. Furthermore, the three proposed time sampling strategies are limited to participating media; extending this to surface transport results in a much narrower sampling space. Metropolis Light Transport techniques [256] represent promising candidates in this regard, where temporal mutation strategies would be needed.

We hope that our research will inspire future work on our understanding of light transport, the design of ultra-fast imaging and the development of novel rendering techniques. For instance, several geometric approaches to acoustic rendering are also based on ray tracing: a more extensive analysis of similarities between acoustic and transient rendering might prove fruitful to both domains. Our code and datasets (scenes and movies) are publicly available at `http://giga.cps.unizar.es/~ajarabo/pubs/transientSIGA14/`.

6 PROGRESSIVE TRANSIENT PHOTON BEAMS

In this chapter we introduce a novel density estimation algorithm for transient rendering of participating media based on the framework proposed in previous Chapter 5. The photon beams algorithm [118] supposed a significant advance in steady-state rendering in participating media. By extending radiance records from points to beams, the density of information within a participating medium is dramatically increased, significantly improving convergence. In this work we make the observation that this aspect presents key benefits also in transient state, where variance is usually aggravated over time due to uneven distributions of radiance samples. The continuity of photon beams along the direction of propagation allows to densely fill not only space but also time. Motivated by this observations, we adapt the progressive photon beams algorithm [119] by accounting for temporal delays in the radiative transfer equation, and provide a progressive version in both time and space. We derive optimal convergence rates accounting for spatial and temporal kernels, and demonstrate the potential of the method in a wide variety of scenes including caustics and multiple scattering.

This work has been conditionally accepted after minor revisions to Computer Graphics Forum (CGF). This work is an extension from an article accepted at Spanish Conference on Computer Graphics (CEIG), where it was granted one of the two best papers awards. Additionally a poster version of the CEIG article was accepted at SIGGRAPH 2017, where it ended as one of the semifinalists on the graduate category of the Student Research Competition (SRC).

J. Marco, I. Guillén, W. Jarosz, D. Gutierrez & A. Jarabo
PROGRESSIVE TRANSIENT PHOTON BEAMS
Conditionally accepted after minor revisions at Computer Graphics Forum

J. Marco, W. Jarosz, D. Gutierrez & A. Jarabo
TRANSIENT PHOTON BEAMS
Spanish Conference on Computer Graphics (CEIG), *Best paper (1 in 2)*

J. Marco, W. Jarosz, D. Gutierrez & A. Jarabo
TRANSIENT PHOTON BEAMS
SIGGRAPH 2017 Posters, *Semifinalist in the SRC, graduate category*

6.1 INTRODUCTION

The emergence of transient imaging has led to a vast number of applications in graphics and vision [113], where the ability of sensing the world at extreme high temporal resolution allows new applications such as imaging light in motion [259], appearance capture [195], geometry reconstruction [25, 179], or vision through media [24, 279] and around the corner [8, 257]. Sensing through media is one of the key applications: The ability of demultiplexing light interactions in the temporal domain is a very promising approach for important practical domains such as non-invasive medical imaging, underwater vision, or autonomous driving through fog. Accurately simulating light transport could help enormously in these applications, potentially serving as a benchmark, a forward model in optimization, or as a training set for machine learning.

Transient rendering in media is, however, still challenging: The increased dimensionality (time) increases variance dramatically in Monte Carlo algorithms, potentially leading to impractical rendering times. This variance is especially harmful in media, where the signal tends to be smooth due to the low-pass filtering behavior of scattering, in both the spatial and temporal domains. One of the major drawbacks of transient rendering is that it requires much higher sampling rates to fill up the extended temporal domain, specially when using $0D$ (photon) point samples, which are sparsely distributed across both time and space. We make the observation that $1D$ photon trajectories populate both space and time much more densely; hence, a technique based on photon beams [118] should significantly reduce the rendering time when computing a noise-free time-resolved render, and, given its density estimation nature, it could naturally combine with the temporal domain density estimation proposed by Jarabo et al. [111].

We present a new method for transient-state rendering of participating media, that leverages the good properties of density estimation for reconstructing smooth signals. Our work improves Jarabo et al. [111] by extending *progressive photon beams* (PPB) [119] to the transient domain, and combining it with temporal density estimation for improved reconstruction in both the spatial and temporal domains. Our technique is biased but consistent, converging to the ground truth using finite memory by taking advantage on the progressive [83, 149] nature of density estimation. We analyze the asymptotic convergence of our proposed space-time density estimation, computing the optimal kernel reduction ratios for both domains. Finally, we demonstrate our method on a variety of scenes with complex volumetric light transport, featuring high-frequency occlusions, caustics, or glossy reflections, and show its improved performance over naively extending PPB to the transient domain.

This work is an extension of our previous work on rendering transient volumetric light transport [180], where we proposed a naive extension of photon beams to transient state. Here we increase the applicability of the method, by proposing a progressive version of the space-time density estimation, and rigorously analyze its convergence.

6.2 RELATED WORK

Rendering participating media is a long-standing problem in computer graphics, with a vast literature on the topic. Here we focus on works related directly with the scope of the paper. For a wider overview on the field, we refer to the recent survey by Novák et al. [205].

PHOTON-BASED LIGHT TRANSPORT. Photon mapping [121] is one of the most versatile and robust methods for rendering complex global illumination, with several extensions for making it compatible with motion blur [26], adapting the distribution of photons [75, 240], carefully selecting the radiance estimation kernel [106, 137, 240], combining it with unbiased techniques [63, 85], or making it progressive for ensuring consistency within a limited memory budget [83, 149]. Hachisuka et al.'s [86] recent SIGGRAPH course provides an in-depth overview.

Jensen and Christensen [122] were the first to extend photon mapping to media, and Jarosz and colleagues [117] significantly improved eits efficiency with the beam radiance estimate, which replaces repeated point queries with one "beam" query finding all photons along the entire camera ray. Jarosz et al. [118] later applied this idea to the photon tracing process by storing full photon trajectories (photon beams), leading to a dramatic increase in photon density for the same photon tracing step. Their progressive and hybrid counterparts [118, 157] leveraged the benefits of photon beams while providing consistent solutions using finite memory. Recently, Bitterli and Jarosz [21] generalized $0D$ photon points and $1D$ photon beams to even higher dimensions, proposing the use of photon planes ($2D$), volumes ($3D$) and, in theory, higher-dimensional geometries, leading to unbiased density estimation. All these works are, however, restricted to steady-state renders; we instead focus on simulating light transport in transient state.

TRANSIENT RENDERING. Though the transport equations [28, 66] are time-resolved, most rendering algorithms focus on steady-state light transport. Still, several works have been proposed to deal with light transport in a time-resolved manner. In particular, most previous work on transient rendering has focused on simulating surfaces transport: Klein et al.[148] extended Smiths' transient radiosity [239] for second bounce diffuse illumination, while other work has used more general methods based on transient extensions of Monte Carlo (bidirectional) path tracing [108, 109, 111, 216] and photon mapping [186, 206]. Several works have also dealt with time-resolved transport on the field of neutron transport [14, 27, 45, 274]. Closer to our work, Ament and colleagues [5] rendered transient light transport in refractive media using volumetric photon mapping, but they do not provide an efficient approach that guarantees consistency. Jarabo et al. [111] proposed a transient extension of the path integral, and introduced an efficient technique for reconstructing the temporal signal based on density estimation. They also proposed a set of techniques for sampling media interactions uniformly in time. Their method is however limited to bidirectional path tracing and photon mapping, often failing to densely populate media in the temporal domain. Finally, Bitterli [20] and Marco et al. [178, 180] proposed a transient extension of the photon beams algorithm, but these approaches are not progressive, therefore not converging to the correct solution in the limit. Our work extends the latter, proposing a progressive, consistent, and robust method for rendering transient light transport. We leverage beams continuity and spatio-temporal density estimation to mitigate variance in the temporal domain, and derive the parameters for optimal convergence of the method.

6.3 TRANSIENT RADIATIVE TRANSFER

The *radiative transfer equation* (RTE) [28] models the behavior of light traveling through a medium. While the original formulation is time-resolved, its integral form used in traditional rendering ignores this temporal dependence, and computes the radiance L reaching any point $\mathbf{x}$ from direction $\vec{\omega}$ as

$$L(\mathbf{x},\vec{\omega}) = T_r(\mathbf{x},\mathbf{x}_s)\, L_s(\mathbf{x}_s,\vec{\omega}) + \int_0^s \mu_s(\mathbf{x}_q)\, T_r(\mathbf{x},\mathbf{x}_q)\, L_o(\mathbf{x}_q,\vec{\omega})\mathrm{d}q, \quad (6.1)$$

where $\mathbf{x}_d = \mathbf{x} - d \cdot \vec{\omega}$ is a point at distance d, μ_s is the scattering coefficient, and $T_r(\mathbf{x},\mathbf{x}_d) = \exp(-\int_0^d \mu_t(\mathbf{x}_{d'})\mathrm{d}d')$ is the *transmittance* describing the fraction of photons that make it between $\mathbf{x}$ and $\mathbf{x}_d$ without undergoing extinction at any point $\mathbf{x}_{d'}$, determined by the *extinction coefficient* $\mu_t(\mathbf{x}_{d'})$. The outgoing radiance L_o in direction $\vec{\omega}$ from a medium point $\mathbf{x}_q$ at distance q is defined by the scattering integral:

$$L_o(\mathbf{x}_q,\vec{\omega}) = L_e(\mathbf{x}_q,\vec{\omega}) + \int_{\mathcal{S}} f_s(\mathbf{x}_q,\vec{\omega}_i,\vec{\omega})\, L(\mathbf{x}_q,\vec{\omega}_i)\mathrm{d}\vec{\omega}_i, \quad (6.2)$$

where $\mathcal{S}$ is the spherical domain, and f_s is the phase function. L_s is defined analogously via the *rendering equation* [133], but integrated over the hemispherical domain, and using the cosine-weighted BSDF in place of the phase function.

TRANSIENT RTE Equations (6.1) and (6.2) assume that the speed of light is infinite. However, if we want to solve the RTE at time scales comparable to the speed of light we need to incorporate the different delays affecting light. In the following we review the main practical considerations for accounting time into the integral form of the RTE for its application in transient rendering. Light takes a certain amount of time to propagate through space, and therefore light transport from a point $\mathbf{x}_0$ towards a point $\mathbf{x}_1$ does not occur immediately. In the absence of scattering effects, transport between two points $\mathbf{x}_0$ and $\mathbf{x}_1$ occurs as

$$L(\mathbf{x}_1, \vec{\omega}, t) = L(\mathbf{x}_0, -\vec{\omega}, t - \Delta t), \tag{6.3}$$

where Δt is the time it takes the light to go from $\mathbf{x}_0$ to $\mathbf{x}_1$. In turn, Δt is defined by

$$\Delta t(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) = \int_{\mathbf{x}_0}^{\mathbf{x}_1} \frac{\eta(\mathbf{x})}{c} \mathrm{d}\mathbf{x}, \tag{6.4}$$

where $\eta(\mathbf{x})$ is the index of refraction at a medium point $\mathbf{x}$ and c is the speed of light in vacuum. Note that in this case light does not travel in a straight line, but by following the Eikonal equation [5, 80]. In a medium with a constant index of refraction $\eta(\mathbf{x}) = \eta_m$, then $\Delta t(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)$ can be expressed as

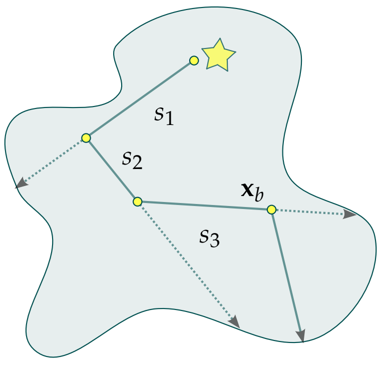
$$\Delta t(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) = \frac{\eta_m}{c} ||\mathbf{x}_1 - \mathbf{x}_0||. \tag{6.5}$$

The second form of delay occurs in the scattering events, and might occur from different sources, including electromagnetic phase shift, fluorescence and phosphorescence, or multiple scattering within the surface (or particle) microgeometry. To account for these sources of scattering delays, we introduce a temporal variable in the phase function as $f_s(\mathbf{x}, \vec{\omega}_i, \vec{\omega}, t)$, where t is the instant of light interacting with the particle before it is scattered. With those delays in place, we reformulate the RTE (Equations (6.1) and (6.2)) introducing the temporal dependence as [66]

$$\begin{aligned} L(\mathbf{x}, \vec{\omega}, t) = {} & T_r(\mathbf{x}, \mathbf{x}_p)\, L_s(\mathbf{x}_p, \vec{\omega}, t - \Delta t_p) \\ & + \int_0^p \mu_s(\mathbf{x}_q)\, T_r(\mathbf{x}, \mathbf{x}_q)\, L_o(\mathbf{x}_q, \vec{\omega}, t - \Delta t_q) \mathrm{d}q, \end{aligned} \tag{6.6}$$

$$\begin{aligned} L_o(\mathbf{x}_q, \vec{\omega}, t) = {} & \int_{-\infty}^{t} L_e(\mathbf{x}_q, \vec{\omega}, t) \mathrm{d}t' \\ & + \int_{\mathcal{S}} \int_{-\infty}^{t} f_s(\mathbf{x}_q, \vec{\omega}_i, \vec{\omega}, t - t')\, L(\mathbf{x}_q, \vec{\omega}_i, t) \mathrm{d}t' \mathrm{d}\vec{\omega}_i, \end{aligned} \tag{6.7}$$

with $\Delta t_p = \Delta t(\mathbf{x} \leftrightarrow \mathbf{x}_p)$ and $\Delta t_q = \Delta t(\mathbf{x} \leftrightarrow \mathbf{x}_q)$ (Equation (6.4)). L_s changes analogously. Note that we assume that the matter does not change at time-scales comparable to the speed of light, and therefore



(a)

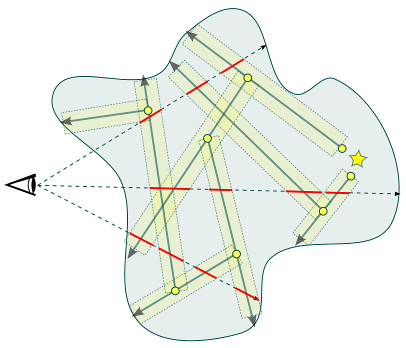

(b)

Figure 6.1: (a) A photon emitted from the light source will take a time $t_{b_0} = \frac{\eta_m}{c}(s_1 + s_2 + s_3)$ to get to $\mathbf{x}_b$. (b) Radiance estimation in the medium is done by intersecting every ray against the photon beam map, and performing density estimations at the ray-beam intersections (red).

avoid any temporal dependence on μ_s and μ_t. Introducing temporal variation at such speeds would produce visible relativistic effects [112, 269].

6.4 TRANSIENT PHOTON BEAMS

Photon beams [118] provide a two-pass numerical solution for rendering participating media in steady state: In the first pass (Figure 6.1a), a series of random walk paths are traced from the light sources. These paths represent packages of light (photons) traveling through the medium. Every interaction of a photon within the medium is stored on a map as a *beam* with a direction $\vec{\omega}_b$, position $\mathbf{x}_b$ and power Φ_b. In the second pass (Figure 6.1b), rays are traced from the camera against the scene, and Equation (6.1) is approximated by summing up the contribution of all near photon beams R_b of the eye ray defined by $r = (\mathbf{x}_r, -\vec{\omega}_r)$

$$L(\mathbf{x}_r, \vec{\omega}_r) \approx \sum_{b \in R_b} L_b(\mathbf{x}_r, \vec{\omega}_r), \tag{6.8}$$

where $L_b(\mathbf{x}_r, \vec{\omega}_r)$ is the contribution of photon beam b. Every photon beam b is considered to have certain radius R_b, and radiance seen by a camera ray is computed by performing a density estimation on every ray-beam intersection. For 1D and 2D kernels, this radiance is computed as

$$L_b^{1\mathrm{D}}(\mathbf{x}_r, \vec{\omega}_c) = K_{1\mathrm{D}}(R_b)\Phi_b f_s(\theta_b)\mu_s \frac{e^{-\mu_t s_b} e^{-\mu_t s_r}}{\sin\theta_b}, \tag{6.9}$$

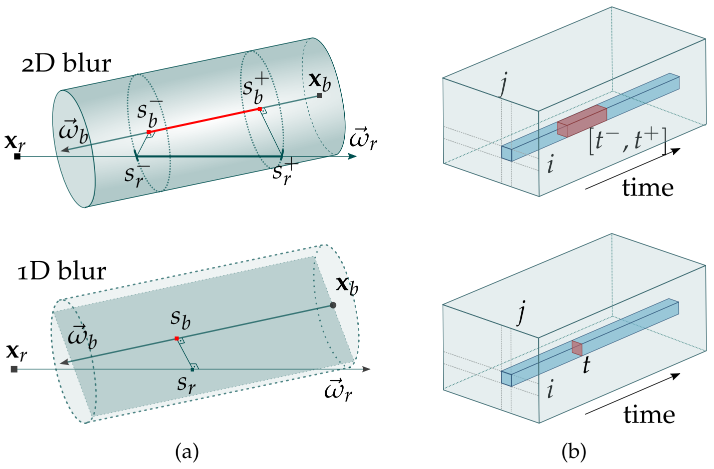


Figure 6.2: (a) Ray-beam intersection for density estimation using a 2D kernel (top) and 1D kernel (bottom). Time delays t_b, t_r within these spatial density estimations will depend on the ray-beam orientation the blur region intersections s_b, s_r, the speed of light, and the index of refraction of the media. (b) Radiance estimate of a single beam at pixel ij using a 2D blur generates a temporal footprint over a time interval $[t^-, t^+]$ (top) while radiance estimate using a 1D blur occurs at a single time instant t (bottom).

$$\begin{aligned} L_b^{2\mathrm{D}}(\mathbf{x}_r, \vec{\omega}_r) = & K_{2\mathrm{D}}(R_b)\Phi_b f_s(\theta_b)\mu_s \\ & \frac{e^{-\mu_t(s_c^- - s_c^+)(|\cos\theta_b| - 1)} - 1}{e^{\mu_t(s_r^- + s_b^-)}\mu_t(|\cos\theta_b| - 1)}, \end{aligned} \tag{6.10}$$

where the beam is defined by $\mathbf{x}_b + s_b\vec{\omega}_b$ and the ray is defined by $\mathbf{x}_r + s_r\vec{\omega}_r$ (see setups in Figure 6.2a).

6.4.1 *Our algorithm*

To generalize photon beams to the transient domain, we need to account for the duration of light paths. This requires considering propagation and scattering delays along the camera and light subpaths, but also the effect of time in the density estimation connecting these two subpaths.

CREATING THE PHOTON MAP We compute the photon propagation as a standard random walk through the scene, which can be modeled using the subpath formulation defined by Jarabo et al. [111]. Let us define a light subpath $\bar{\mathbf{x}}_l = \mathbf{x}_0 ... \mathbf{x}_{k-1}$, with k vertices, where $\mathbf{x}_0$ is the light source. This light path defines $k-1$ photon beams, in which a beam b_j is defined by its origin at $\mathbf{x}_{b_j} = \mathbf{x}_j$ and direction $\vec{\omega}_{b_j} = \frac{\mathbf{x}_{j+1} - \mathbf{x}_j}{\|\mathbf{x}_{j+1} - \mathbf{x}_j\|}$.

Using Jarabo's definition of the path integral (and therefore of the contribution of the subpaths), we compute the flux of each photon as:

$$\Phi_{b_j} = \frac{f(\bar{\mathbf{x}}_j, \bar{\tau}_j)}{Mp(\bar{\mathbf{x}}_j, \bar{\tau}_j)} = \frac{L_e(\mathbf{x}_0 \to \mathbf{x}_1, \tau_0) T(\bar{\mathbf{x}}_j, \bar{\tau}_j)}{M \prod_{i=0}^{j} p(\mathbf{x}_i, \tau_i)}, \tag{6.11}$$

with $\bar{\mathbf{x}}_j$ the subpath of $\bar{\mathbf{x}}_l$ up the vertex j, f the subpath contribution function, $\bar{\tau}_j = \tau_0 ... \tau_j$ the sequence of time delays up to vertex j, M the number of photon random walks sampled, $L_e(\mathbf{x}_0 \to \mathbf{x}_1, \tau_0)$ the emission function, $p(\mathbf{x}_i, \tau_i)$ the probability density of sampling vertex $\mathbf{x}_i$ with time delay τ_i. The throughput, $T(\bar{\mathbf{x}}_j, \bar{\tau}_j)$, of subpath $(\mathbf{x}_i, \tau_j)$ is defined as:

$$T(\bar{\mathbf{x}}_j, \bar{\tau}_j) = \left[\prod_{i=1}^{j-1} f_s(\mathbf{x}_i, \tau_j) \right] \left[\prod_{i=0}^{j-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}) \, V(\mathbf{x}_i, \mathbf{x}_{i+1}) \right], \tag{6.12}$$

with $f_s(\mathbf{x}_i, \tau_j)$ the scattering event at vertex $\mathbf{x}_i$ with delay τ_j, and $G(\mathbf{x}_i, \mathbf{x}_{i+1})$ and $V(\mathbf{x}_i, \mathbf{x}_{i+1})$ the geometry and visibility terms between vertices $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, respectively. Finally, for transient state we need to know the instant t_{b_j} at which the photon beam is created (through emission or scattering), defined as:

$$t_{b_j} = \sum_{i=0}^{j-1} \tau_j + \sum_{i=0}^{j-1} \Delta t(\mathbf{x}_i, \mathbf{x}_{i+1}). \tag{6.13}$$

RENDERING For rendering, we adapt Equation (6.8) to account for the temporal domain, as

$$L(\mathbf{x}_r, \vec{\omega}_r, t) \approx \sum_{b \in R_b} L_b(\mathbf{x}_r, \vec{\omega}_r, t), \tag{6.14}$$

with $L_b(\mathbf{x}_r, \vec{\omega}_r, t)$ the radiance estimation for beam b to ray t at instant t. In essence, $L_b(\mathbf{x}_r, \vec{\omega}_r, t)$ will return zero radiance if t is out of the temporal footprint of the density estimation kernel. Depending on the dimensionality of the density estimation, Jarosz and colleagues [118] proposed three different estimators based on 3D, 2D and 1D kernels. Since the 3D kernel results impractical due to costly 3D convolutions, we focus on 1D and 2D kernels (Equations (6.9) and (6.10)), and extend them to transient state, assuming homogeneous media.

KERNEL 2D We generalize Jarosz's et al.'s 2D estimate $L_b^{2\mathrm{D}}$ (Equation (6.10)) by introducing a temporal function $W(t)$ as

$$\begin{aligned} L_b^{2\mathrm{D}}(\mathbf{x}_r, \vec{\omega}_r, t) =& K_{2\mathrm{D}}(R_b) \Phi_b f_s(\theta_b, t) \mu_s \\ & \frac{e^{-\mu_t (s_r^- - s_r^+)(|\cos\theta_b| - 1)} - 1}{e^{\mu_t (s_r^- + s_b^-)} \mu_t (|\cos\theta_b| - 1)} W_{2D}(t), \end{aligned} \tag{6.15}$$

where $[s_r^-, s_r^+]$ are the limits of the ray-beam intersection (Figure 6.2a), θ_b is the angle between $\vec{\omega}_b$ and $\vec{\omega}_r$, and $K_{2D}(R_b)$ is a canonical 2D kernel with radius R_b. The temporal function $W_{2D}(t)$ models the temporal footprint of the 2D kernel as

$$W_{2D}(t) = \begin{cases} \frac{1}{t^+ - t^-} & \text{if } t \in (t^-, t^+) \\ 0 & \text{otherwise} \end{cases}, \tag{6.16}$$

where $t^- = t_b + t_r + \frac{\eta_m}{c}(s_r^- + s_b^-)$ and $t^+ = t_b + t_r + \frac{\eta_m}{c}(s_r^+ + s_b^+)$, and t_r and t_b are the initial times of the camera ray and beam, respectively. Note that due to transmittance, the photon energy varies as it travels across the blur region. Evenly distributing the integrated radiance L_b across this interval introduces temporal bias, in addition to the inherent spatial bias introduced by density estimation. However we observed this even distribution provides a good tradeoff between bias, variance, and computational overhead.

KERNEL 1D In the 1D kernel defined for density estimation by Jarosz et al. the spatial blur is performed over a line. Therefore, the energy of the beam is just spread on the ray on a single point at $r(s_r)$, from a single point of the beam $b(s_b)$ (see Figure 6.2a). In consequence, $s_r^\pm \to s_r$ and $s_b^\pm \to s_b$, which implies that $t^\pm \to t_{br}$, and the temporal function reduces to $W_{1D}(t - tb) = \delta(t)$, with $\delta(t)$ the Dirac delta function. With that in place, we transform Jarosz et al. 1D estimate to

$$L_b^{1D}(\mathbf{x}_r, \vec{\omega}_r, t) = K_{1D}(R_b)\Phi_b f_s(\theta_b, t)\mu_s \frac{e^{-\mu_t s_b} e^{-\mu_t s_r}}{\sin\theta_b}\delta(t - t_b), \tag{6.17}$$

with $K_{1D}(R_b)$ a 1D kernel with radius R_b.

IMPLEMENTATION Since photon beams correspond to full photon trajectories, they allow us to estimate radiance at any position $\mathbf{x}_b + s\vec{\omega}_b$ of the beam, and therefore at any arbitrary time $t(\mathbf{x}_b + s\vec{\omega}_b)$. As mentioned, one-dimensional radiance estimate corresponds to a single time across the beam. In a traditional rendering process where camera rays are traced through view-plane pixels against the beams map, the temporal definition *within* a pixel will be proportional to the amount of samples per pixel taken. Additionally, 2D blur requires distributing every radiance estimate along a time interval, which reduces variance in the time dimension of a pixel at the expense of introducing additional temporal bias.

Finally, note that the temporal footprint of the density estimation might be arbitrarily small, so the probability of finding a beam b at an specific time might be very low. We alleviate this issue using path reuse via density estimation [111]. In particular, for the non-progressive results we use histogram temporal density estimation. In this technique, the samples in the temporal domain are reused across all frames by evaluating their contribution functions, which correspond to the temporal window covered by each frame.. In Section 6.5 we introduce temporal kernel-based density estimation, and combine it with the spatial density estimation of the beam.

6.5 PROGRESSIVE TRANSIENT PHOTON BEAMS

By means of Equations (6.15) and (6.17) we have introduced temporal dependence on the spatial density estimations that use 2D and 1D kernels, respectively. These density estimations reduce variance at the expense of introducing bias in the results, which means both Equations (6.8) and (6.14) will not converge to the correct solution, even with an infinite number of photons M. To avoid this, progressive density estimation aims to provide a biased, yet consistent technique, that in the limit converges to the expected value (in other words, the bias vanishes in the limit). The key idea is to average several render passes with a finite number of photon random walks M, progressively reducing the bias in each iteration while allowing variance to slightly increase.

In order to fully leverage a progressive approach, we propose to combine our time-resolved spatial density estimations (Section 6.4) with additional *temporal* density estimations. While our time-resolved 2D spatial kernel implicitly performs a temporal blur over the interval $[t^-, t^+]$, it is coupled with the spatial blur. This does not allow to choose its own initial kernel size for the temporal density estimation, which is a desirable degree of freedom since the temporal resolution may not be proportional to the spatial one. In contrast, our time-resolved 1D spatial kernel does not perform a temporal blur, since the footprint is a single instant in time. As we show in the remainder of this section, this allows us to perform additional progressive temporal density estimations with an independent initial kernel size, while keeping the same two-dimensionality (1D spatial and 1D temporal). In the following, we introduce our spatio-temporal beam density estimation based on our time-resolved 1D kernel, and then present our progressive approach.

SPATIO-TEMPORAL BEAM ESTIMATION Jarabo et al. [111] showed that progressive density estimations in the temporal domain can in fact improve the convergence rate for transient rendering, in particular when compared with the histogram method used in Section 6.4 for rendering the temporal domain. To combine such approach with the (progressive) spatial density estimation in photon beams [119], we

Algorithm 1 Pseudo-code of our progressive spatio-temporal density estimation.

```
$L_n \leftarrow 0$
$R_b \leftarrow R_0$
$\mathcal{T} \leftarrow \mathcal{T}_0$
**for** $i \in [0..N)$ **do**
  $r \leftarrow$ traceRay()
  $B \leftarrow$ beamsMap()                         (Eqs. (6.6), (6.7), (6.11)-(6.13))
  $R_b \leftarrow R_b\sqrt{\frac{i+2/3}{i+1}}$                   (Eq. (6.20), left)
  $\mathcal{T} \leftarrow \mathcal{T}\sqrt{\frac{i+2/3}{i+1}}$                    (Eq. (6.20), right)
  $L_b \leftarrow 0$
  **for** $b \in B$ **do**
    $L_b \leftarrow L_b +$ radiance($r$, $b$, $R_b$, $\mathcal{T}$)      (Eq. (6.18))
  **end for**
  $L_n \leftarrow L_n + L_b$
**end for**
```

reformulate the 1D kernel in Equation (6.17), by convolving it with a 1D temporal kernel $K_{\mathcal{T}}(t)$ so that

$$L_b^{1\text{D}}(\mathbf{x}_r, \vec{\omega}_r, t) = K_{1\text{D}}(R_b)\Phi_b f_s(\theta_b, t)\mu_s \frac{e^{-\mu_t s_b} e^{-\mu_t s_r}}{\sin\theta_b} K_{\mathcal{T}}(t - t_b). \tag{6.18}$$

PROGRESSIVE TRANSIENT PHOTON BEAMS We generalize the computation of $L(\mathbf{x}_r, \vec{\omega}_r, t)$ (Equation (6.14)) using an iterative estimator, defined as

$$L(\mathbf{x}_r, \vec{\omega}_r, t) \approx \widehat{L}_n(\mathbf{x}_r, \vec{\omega}_r, t) = \frac{1}{n}\sum_{i=0}^{n}\sum_{b\in B_i} L_b(\mathbf{x}_r, \vec{\omega}_r, t) \tag{6.19}$$

with $\widehat{L}_n$ the estimate of L after n iterations, and B_i the set of photon beams in iteration i. Note that the previous equation assumes that the camera ray r is the same for all iterations. That is not necessarily true (and in fact it is not) but for simplicity we express this way.

The error of the estimate $\widehat{L}_n$ is defined by its bias and variance, which as shown in Appendix 6.B is dependent on the bandwidth of the spatial and temporal kernels. In particular, the variance of the error increases linearly with the bandwidth of the kernels, while bias is reduced at the same rate. Then, on each iteration we reduce the bias by allowing the variance to increase at a controlled rate of $(i+1)/(i+\alpha)$, with $\alpha \in [0,1]$ being a parameter that controls how much the variance is allowed to increase at each iteration. To achieve that reduction, on each iteration $i+1$ we reduce the footprint of kernels K_{1D} and $K_{\mathcal{T}}$ ($R_{b|j}$ and $\mathcal{T}_i$) by

$$\frac{R_{b|i+1}}{R_{b|i}} = \left(\frac{i+\alpha}{i+1}\right)^{\beta_R}, \qquad \frac{\mathcal{T}_{i+1}}{\mathcal{T}_i} = \left(\frac{i+\alpha}{i+1}\right)^{\beta_{\mathcal{T}}}, \tag{6.20}$$

where β_R and $\beta_{\mathcal{T}}$ control the individual reduction ratio of each kernel, with $\beta_{\mathcal{T}} = 1 - \beta_R$. A pseudo code of the main steps of our progressive approach can be found in Algorithm 1. In the following, we analyze the convergence rate of the method, and compute the optimal values for the parameters α, $\beta_{\mathcal{T}}$ and β_R.

CONVERGENCE ANALYSIS We analyze the convergence of the algorithm as a function of the *asymptotic mean squared error* (AMSE) defined as

$$\text{AMSE}(\widehat{L}_n) = \text{Var}[\widehat{L}_n] + \text{E}[\epsilon_n]^2, \tag{6.21}$$

where $\text{Var}[\widehat{L}_n]$ is the variance of the estimate and $\text{E}[\epsilon_n]$ is the bias at iteration n. As shown in Appendix 6.C, the variance converges with rate

$$\text{Var}[\widehat{L}_n] \approx O(n^{-1}) + O(n^{-\alpha}) = O(n^{-\alpha}), \tag{6.22}$$

while the bias converges with rate

$$\text{E}[\epsilon_n] = O(n^{1-\alpha})^{-2\beta_{\mathcal{T}}} + O(n^{1-\alpha})^{2\beta_{\mathcal{T}}-2}. \tag{6.23}$$

Plugging Equation (6.22) and (6.23) into Equation (6.21), we can model the AMSE as

$$\text{AMSE}(\widehat{L}_n) = O(n^{-\alpha}) + \left(O(n^{1-\alpha})^{-2\beta_{\mathcal{T}}} + O(n^{1-\alpha})^{2\beta_{\mathcal{T}}-2}\right)^2. \tag{6.24}$$

Finally, by minimizing Equation (6.24) (see Appendix 6.D) we obtain the values for optimal asymptotic convergence $\beta_{\mathcal{T}} = 1/2$ and $\alpha = 2/3$, which by substitution gives us the final asymptotic convergence rate of our progressive transient photon beams

$$\text{AMSE}(\widehat{L}_n) = O(n^{-\frac{2}{3}}). \tag{6.25}$$

6.6 RESULTS

In the following we illustrate the results of our proposed method in five scenes: CORNELL SPHERES, MIRRORS, PUMPKIN, SOCCER [245], PUMPKIN, and JUICE. See Figures 6.4, 6.3 (right), and 6.8 (left) for steady-state renders of the scenes. Results of Figures 6.5 and 6.6 were taken on a desktop PC with Intel i7 and 4GB RAM using a transient 2D kernel (Equation (6.15)). Figures 6.3, 6.7, and 6.8 were rendered on an

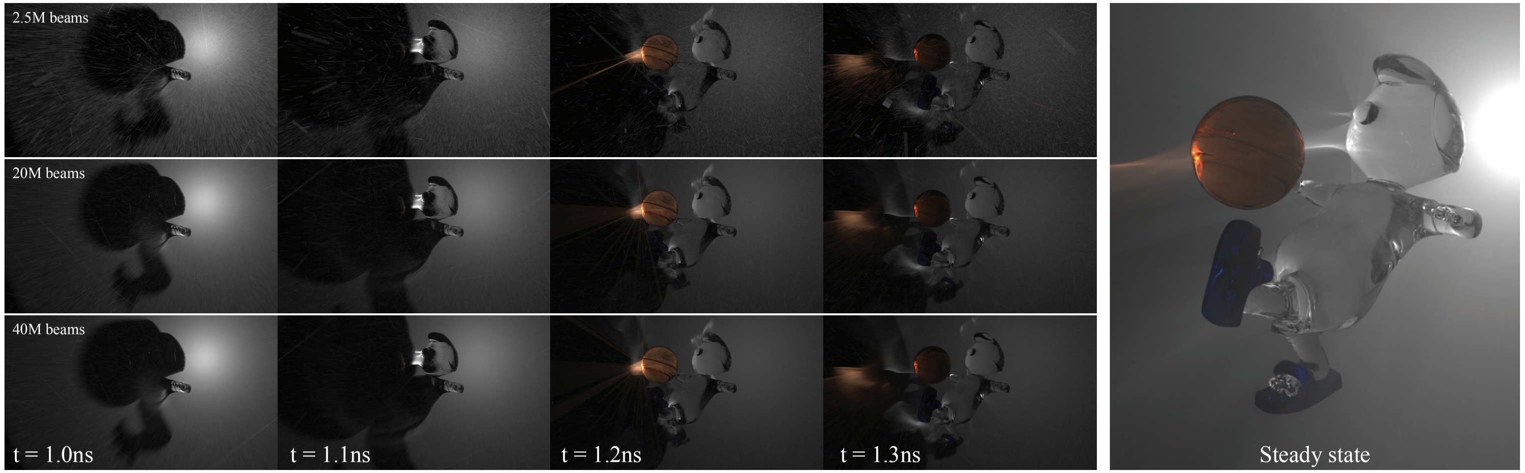


Figure 6.3: We present a robust method for transient rendering in participative media. We reformulate photon beams to support transient light propagation, and derive a progressive approach that performs spatio-temporal density estimations. Here we show different frames of transient light transport (left sequences) in the SOCCER scene (steady-state render on the right), at different levels of convergence. The scene features complex caustic light transport in the medium due to multiple reflections and refractions in smooth dielectrics inside the medium. The progressive nature of our algorithm is consistent under finite memory, and works by accumulating several independent iterations of transient photon beams that progressively reduce both bias and variance. Please refer to the supplemental video[1] for the full sequence.

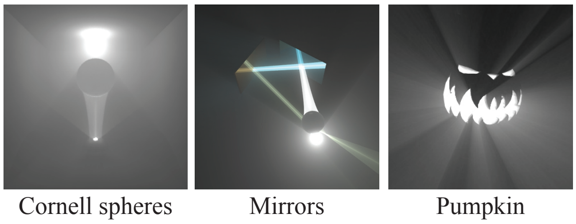


Figure 6.4: Steady-state renders for the scenes CORNELL SPHERES (Figure 6.5), MIRRORS (Figure 6.6), and PUMPKIN (Figure 6.7).

Intel Xeon E5 with 256GB RAM, using our progressive spatio-temporal kernel density estimations (Section 6.5) derived from the transient spatial 1D kernel (Equation (6.17)). All temporal density estimations are performed using radiance samples within fixed radius of the corresponding iteration (instead of using a nearest neighbor approach). Please refer to the supplemental video[1] for the full sequences of all the scenes.

Figure 6.5 shows a Cornell box filled with a scattering medium, and demonstrates the effect of *camera unwarping* [259] when rendering. Camera unwarping is an intuitive way of visualizing how light propagates *locally* on the scene without accounting for the time light takes to reach the camera. The scene consists of a diffuse Cornell box with a point light on the top, a glass refractive sphere (top, IOR = 1.5) and a mirror sphere (bottom). While Figure 6.5b shows the real propagation of light—including camera time—, Figure 6.5a depicts more intuitively how light comes out from the point light, travels through the refractive sphere, and the generated caustic bounces on the mirror sphere. Note how in the top sequence we can clearly see how light is slowed down through the glass sphere due to the higher index of refraction. We can also observe multiple scattered light (particularly noticeable in frames t=4ns and t=6ns) as a secondary wavefront.

Figure 6.6 compares visualizations of light propagation within the MIRRORS scene using Heaviside and Dirac delta light emission. The scene is composed by two colored mirrors and a glass sphere with IOR = 1.5, and was rendered using the previously mentioned camera unwarping. We can observe how delta emission generates wavefronts that go through the ball and bounce in the mirrors, creating wavefront holes where constant emission creates medium shadows. In the last frame of the top row Delta emission clearly depicts the slowed down caustic through the glass ball respect to the main wavefront.

Our progressive method combines time-resolved 1D spatial kernels of photon beams and temporal density estimations, reducing bias while providing consistent solutions in the limit with an optimal convergence

[1] http://webdiis.unizar.es/~juliom/pubs/2018CGF-PTPB/2018CGF_PTPB.mp4

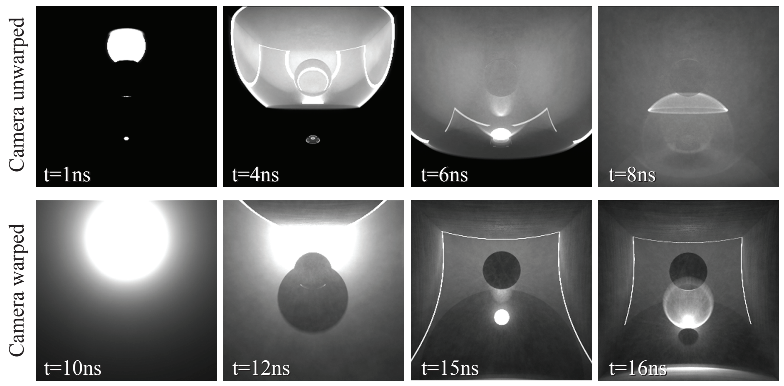


Figure 6.5: Comparison of CORNELL SPHERES scene using *camera-unwarping* (top), where we do not take into account the camera time, and real propagation of light (bottom). In the bottom row the shape of the wavefront is altered by the camera time, as if we were scanning the scene from the viewpoint towards the furthest parts of the scene. Camera unwarping on the other hand illustrates more intuitively how light propagates locally.

rate of $O(n^{-\frac{2}{3}})$. In Figure 6.7 we analyze its convergence with respect to progressive transient path tracing with temporal KDE [111] (PTPT). In the middle graph we show the temporal profile on a single pixel for both our algorithm and PTPT after 4096 equal-time iterations, where both algorithms converge to the reference solution taken with transient path tracing (no temporal KDE) with 64 million samples. While PTPT presents faster convergence (see Figure 6.7, right graph), our algorithm presents a better behavior over time where variance increases due to the lack of samples (center graph). Additionally, it requires much fewer iterations than PTPT to achieve a similar MSE (see log-log right graph).

In Figure 6.3 we show a more complex scenario, with different caustics rendered, with our progressive algorithm. It contains a smooth dielectric figurine with different transmission albedos placed within a participating medium with an isotropic phase function. Our method is capable of handling complex caustics transmitted from light sources through the player, and then through the ball. Our algorithm progressively reduces bias and variance to provide a consistent solution.

Finally in Figure 6.8 we illustrate a setup combining different media properties, and specular refractive and reflective materials. The liquid has a very forward phase function, making the light first travel through the direction of the stream (t = 4.6 ns), and then going through the liquid inside the glass (t = 5.1ns to t = 6.3ns). The mirror surface makes the light to bounce back to the surrounding medium as a caustic through the water spills and ice cubes at t = 5.1ns and t = 6.6ns. Note

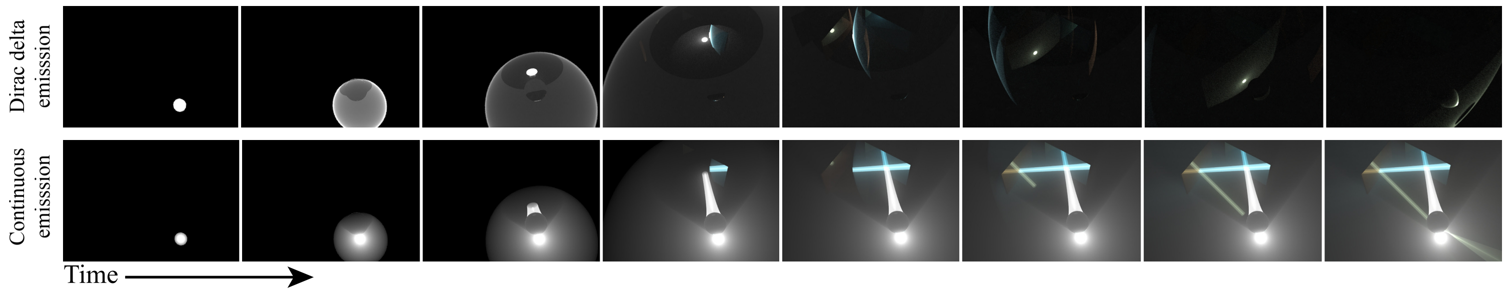
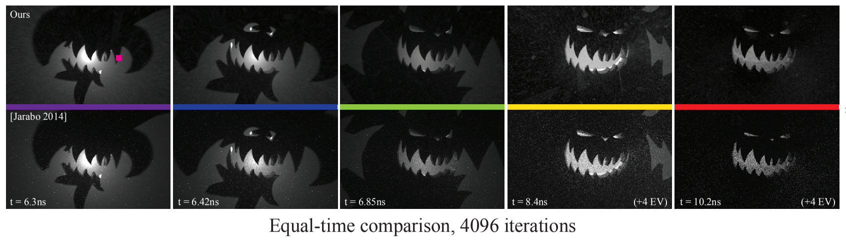
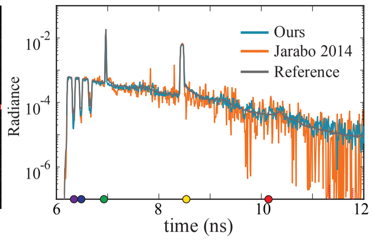


Figure 6.6: Comparison between Dirac delta (top) and continuous (Heaviside) emission (bottom). Dirac delta emission lets us see how a pulse of light travels and scatters across the scene, depicting the light wavefronts bouncing on the mirrors and going through the glass ball. Continuous emission shows how light is emitted until it reaches every point in the scene, as if we were taking a picture with a camera at very slow-motion.





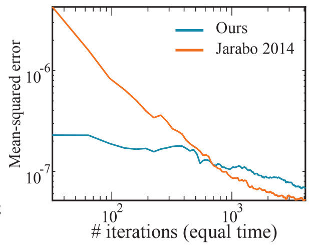


Figure 6.7: The PUMPKIN scene shows a jack o'lantern embedding a point light that creates hard shadows through the holes. The left frames show a sequence of the time-resolved renders after 4096 iterations of our algorithm (10k beams / iteration), and temporal KDE on a progressive transient path tracer (PTPT, 16spp / iteration) [111]. The middle plot compares the whole temporal footprint at the pink marker. Reference solution (dark grey) was obtained with a transient path tracer (no KDE) using 64M samples per pixel. Right plot shows MSE convergence with respect to the number of progressive iterations (in log-log scale), at 1 minute/iteration on each algorithm. As expected, the convergence of our method ($O(n^{-\frac{2}{3}})$) is slower than PTPT ($O(n^{-\frac{4}{5}})$) ; however, as shown in the equal-time comparison, our algorithm presents better temporal behavior with much less variance on later timings.

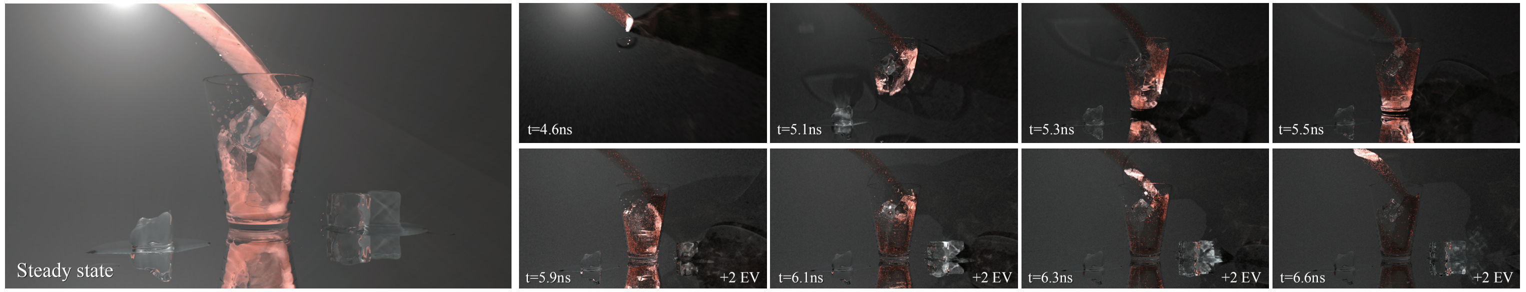


Figure 6.8: We illustrate the potential of our method in the Juice scene [19], which presents a scene very difficult to render for path tracing methods, but well-handled by photon-based methods. The scene is filled by a thin participating medium, while the glass contains ruby grapefruit juice as measured by Narasimhan et al. [200]. The highly forward phase function of the juice, as well as the delta interactions on the glass, ice cubes, and the mirror floor surface, generate complex caustic patterns which our method is able to simulate in transient state. Bottom row has increased exposure respect to top row to show the radiance at later timings.

that these are not fully observable in the steady-state render (left) due to the accumulated radiance from the surrounding medium and the adjusted exposure of the image.

6.7 CONCLUSIONS

In this work we have presented a robust progressive method for efficiently rendering transient light transport with consistent results. We derived our method based on progressive photon beams [119], extending its density estimators to account for light time-of-flight, and deriving a new progressive scheme. We then compute the convergence of the method, and derive the parameters for optimal asymptotic convergence. Our results demonstrate that combining continuous photon trajectories in transient state and our optimal spatio-temporal convergence rates allow to robustly compute a noise-free solutions to the time-resolved RTE for complex light paths. We believe that our work might be very useful for developing new techniques for transient imaging and reconstruction in media, as well as to obtain new insights on time-resolved light transport.

As future work it would be interesting to analyze more thoroughly the optimal performance and kernels for variance reduction and bias impact in transient state, under varying media characteristics. In addition, extending our method to leverage recent advances in media transport, such as transient-state adaptations of higher-dimensional photon estimators [21] as well as hybrid techniques [157], could improve performance of time-resolved rendering for a general set of geometries and media characteristics.

APPENDICES

6.A ERROR IN TRANSIENT PROGRESSIVE PHOTON BEAMS

Here we analyze the consistency of the transient progressive photon beams algorithm described in Section 6.5. For our analysis on the error of the estimate, we use the *asymptotic mean squared error* (AMSE) defined as

$$\mathrm{AMSE}(\widehat{L}_n) = \mathrm{Var}[\widehat{L}_n] + \mathrm{E}[\epsilon_n]^2, \tag{6.26}$$

where $\mathrm{Var}[\widehat{L}_n]$ is the variance of the estimate and $\mathrm{E}[\epsilon_n]$ is the bias at iteration n. We model $\mathrm{Var}[\widehat{L}_n]$ as [149]

$$\mathrm{Var}[\widehat{L}_n] = \frac{1}{n}\mathrm{Var}[\Psi\ L] + \frac{1}{n^2}\sum_{j=1}^{n}\mathrm{Var}[\Psi\ \epsilon_j], \tag{6.27}$$

where Ψ is the contribution of the eye ray, and ϵ_j is the bias for iteration j. The first term is the standard variance of the Monte Carlo estimate, which is unaffected by the kernel. The second term, on the other hand, is the variance of the error, and is dependent on density estimation. On the other hand, the estimated value of the error (bias) $\mathrm{E}[\widehat{L}_n]$ is defined as

$$\mathrm{E}[\widehat{L}_n] = L + \mathrm{E}[\Psi]\mathrm{E}[\epsilon_n], \tag{6.28}$$

where $\mathrm{E}[\epsilon_n]$ is the bias of the estimator after n steps:

$$\mathrm{E}[\epsilon_n] = \frac{1}{n}\sum_{j=1}^{n}\mathrm{E}[\epsilon_j], \tag{6.29}$$

with $\mathrm{E}[\epsilon_j]$ the expected error at iteration j. In the following, we first derive the variance and expected value of the error for a single iteration. Then, we analyze the asymptotic behavior of the these terms, and compute the values for optimal convergence for $\beta_{\mathcal{T}}$, β_R and α.

6.B VARIANCE AND EXPECTED VALUE OF THE ERROR OF THE TIME-RESOLVED BEAM RADIANCE ESTIMATE

We first analyze the variance and expected value of the error (bias) introduced by the radiance estimate at each iteration. Let us first define the error in each iteration as:

$$\begin{aligned}\epsilon &= \widehat{L}_n(\mathbf{x}_r, \vec{\omega}_r, t) - L(\mathbf{x}_r, \vec{\omega}_r, t)\\ &= \sum_{i=1}^{M} K_{1\mathrm{D}}(R_b)K_{\mathcal{T}}(t - t_i)\Phi_i - L(\mathbf{x}_r, \vec{\omega}_r, t).\end{aligned} \tag{6.30}$$

VARIANCE We first define the variance of the error $\mathrm{Var}[\epsilon]$ as (in the following, we omit dependences for clarity):

$$\begin{aligned}\mathrm{Var}[\epsilon] &= \mathrm{Var}[\sum_{i=1}^{M} K_{1\mathrm{D}} K_{\mathcal{T}} \Phi - L] \\ &= (\mathrm{Var}[K_{1\mathrm{D}}] + \mathrm{E}[K_{1\mathrm{D}}]^2)(\mathrm{Var}[K_{\mathcal{T}}] + \mathrm{E}[K_{\mathcal{T}}]^2) \\ &\quad (\mathrm{Var}[\Phi] + \mathrm{E}[\Phi]^2) - \mathrm{E}[K_{1\mathrm{D}}]^2 \mathrm{E}[K_{\mathcal{T}}]^2 \mathrm{E}[\Phi]^2,\end{aligned} \tag{6.31}$$

In order to compute the variance of the error $\mathrm{Var}[\epsilon]$ we need to make a set of assumptions: First, we assume that the beams' probability density is constant within the kernel $K_{1\mathrm{D}}$ in the spatial domain [119], and within $K_{\mathcal{T}}$ in the temporal domain [111]. We denote these probabilities as p_{R_b} and $p_{\mathcal{T}}$ respectively. We also assume that the distance between view ray and photon beam, time t_b and beams' energy Φ_i are independent samples of the random variables D, T and Φ, respectively, which are mutually independent. Finally, we assume that D and T have probability densities p_{R_b} and $p_{\mathcal{T}}$.

With these assumptions, and taking into account that $\mathrm{E}[K_{1\mathrm{D}}] = p_{R_b}$ and $\mathrm{E}[K_{\mathcal{T}}] = p_{\mathcal{T}}$, we can model the the variance introduced by the temporal kernel $\mathrm{Var}[K_{\mathcal{T}}]$ as [111]

$$\mathrm{Var}[K_{\mathcal{T}}] = \frac{p_{\mathcal{T}}}{\mathcal{T}} \int_{\mathbb{R}} k_{\mathcal{T}}(\psi)^2 \mathrm{d}\psi - p_{\mathcal{T}}^2, \tag{6.32}$$

where we express $K_{\mathcal{T}}$ as a canonical kernel $k_{\mathcal{T}}$ with unit integral such that $K_{\mathcal{T}}(\xi) = k_{\mathcal{T}}(\xi/\mathcal{T})\mathcal{T}^{-1}$. Analogously, $\mathrm{Var}[K_{1\mathrm{D}}]$ is [119]:

$$\mathrm{Var}[K_{1\mathrm{D}}] = \frac{p_{R_b}}{R_b} \int_{\mathbb{R}} k_{1D}(\psi)^2 \mathrm{d}\psi - p_{R_b}^2. \tag{6.33}$$

This allow us to express the variance of the error $\mathrm{Var}[\epsilon]$ as:

$$\mathrm{Var}[\epsilon] \approx \left(\mathrm{Var}[\Phi] + \mathrm{E}[\Phi]^2\right) \left(\frac{p_{R_b}}{R_b} \mathcal{C}_{1D}\right) \left(\frac{p_{\mathcal{T}}}{\mathcal{T}} \mathcal{C}_{\mathcal{T}}\right), \tag{6.34}$$

where $\mathcal{C}_{1D}$ and $\mathcal{C}_{\mathcal{T}}$ are kernel-dependent constants. The last term can be neglected by assuming that the kernels cover small areas in their respective domains, which effectively means that $\mathcal{C}_{1D} \gg p_{R_b}$ and $\mathcal{C}_{\mathcal{T}} \gg p_{\mathcal{T}}$. Equation (6.34) shows that for transient density estimation, the variance $\mathrm{Var}[\epsilon]$ is inversely proportional to $R_b \mathcal{T}$.

BIAS Bias at each iteration j is defined as the expected value of the error $\mathrm{E}[\epsilon_j]$ as

$$\begin{aligned}\mathrm{E}[\epsilon_j] &= \mathrm{E}[\sum_{i=1}^{M} K_{1\mathrm{D}}\, K_{\mathcal{T}}\, \Phi - L] \\ &= \mathrm{E}[K_{1\mathrm{D}}]\, \mathrm{E}[K_{\mathcal{T}}]\, \mathrm{E}[\Phi] - L.\end{aligned}$$

Using a second-order expansion of $p_{\mathcal{T}}$ and p_{R_b}, instead of the zeroth-order used when modeling variance, we can express the expected value of $K_{\mathcal{T}}$ as [111]

$$\mathrm{E}[K_{\mathcal{T}}] \approx p_{\mathcal{T}} + \mathcal{T}^2 \int_{\mathbb{R}} k_{\mathcal{T}}(\psi) O(\|\psi\|^2) \mathrm{d}\psi = p_{\mathcal{T}} + \mathcal{T}^2 \mathcal{C}^{ii}_{\mathcal{T}}, \tag{6.35}$$

while the expected value of $K_{1\mathrm{D}}$ is [119]

$$\mathrm{E}[K_{1\mathrm{D}}] \approx p_{R_b} + R_b \int_{\mathbb{R}^2} k_{1D}(\psi) O(\|\psi\|^2) \mathrm{d}\psi = p_{R_b} + R_b \mathcal{C}^{ii}_{1D}, \tag{6.36}$$

where $\mathcal{C}^{ii}_{\mathcal{T}}$ and $\mathcal{C}^{ii}_{1D}$ are constants dependent on the higher-order derivatives of the spatio-temporal light distribution. Using (6.35) and (6.36), and $L = p_{R_b} p_{\mathcal{T}} \mathrm{E}[\Phi]$ we finally compute $\mathrm{E}[\epsilon_j]$ for iteration j as

$$\begin{aligned} \mathrm{E}[\epsilon_j] &\approx (p_{R_b} + {R_b}^2 \mathcal{C}^{ii}_{1D})(p_{\mathcal{T}} + \mathcal{T}^2 \mathcal{C}^{ii}_{\mathcal{T}})\mathrm{E}[\Phi] - p_{R_b} p_{\mathcal{T}} \mathrm{E}[\Phi] \\ &= \mathrm{E}[\Phi](p_{R_b} \mathcal{T}^2 \mathcal{C}^{ii}_{\mathcal{T}} + p_{\mathcal{T}} {R_b}^2 \mathcal{C}^{ii}_{1D} + \mathcal{T}^2 \mathcal{C}^{ii}_{\mathcal{T}} {R_b}^2 \mathcal{C}^{ii}_{1D}). \end{aligned} \tag{6.37}$$

6.C CONVERGENCE ANALYSIS OF PROGRESSIVE TRANSIENT PHOTON BEAMS

Based on the expressions for $\mathrm{Var}[\epsilon]$ and $\mathrm{E}[\epsilon_j]$ defined above (Equations (6.34) and (6.37)), we can know derive the asymptotic behaviour of Equation (6.21). For that, we will compute the variance $\mathrm{Var}[\widehat{L}_n]$ and bias $\mathrm{E}[\epsilon_n]$ after n iterations.

VARIANCE Assuming that the random variables Ψ and ϵ_j are independent, we model the variance of the estimator $\mathrm{Var}[\widehat{L}_n]$ in Equation (6.27) as [149]:

$$\begin{aligned} \mathrm{Var}[\widehat{L}_n] &= \frac{1}{n}\mathrm{Var}[\Psi L] + \frac{1}{n^2}\sum_{j=1}^{n} \mathrm{Var}[\Psi \epsilon_j] \\ &= \frac{1}{n}\mathrm{Var}[\Psi L] + \mathrm{Var}[\Psi]\frac{1}{n^2}\sum_{j=1}^{n}\mathrm{Var}[\epsilon_j] + \\ &\quad \mathrm{E}[\Psi]^2 \frac{1}{n^2}\sum_{j=1}^{n}\mathrm{Var}[\epsilon_j] + \mathrm{Var}[\Psi]\frac{1}{n^2}\sum_{j=1}^{n}\mathrm{E}[\epsilon_j]^2. \end{aligned} \tag{6.38}$$

Following [137], we can approximate $\mathrm{Var}[\epsilon_n]$ as a function of the variance at the first iteration $\mathrm{Var}[\epsilon_1]$ as:

$$\mathrm{Var}[\epsilon_n] \approx \frac{\mathrm{Var}[\epsilon_1]}{(2-\alpha)n^{\alpha}} = O(n^{-\alpha}). \tag{6.39}$$

Finally, by applying $\mathrm{Var}[\epsilon_n]$ and asypmtotic simplifications, we can formulate $\mathrm{Var}[\widehat{L}_n]$ (6.39) as:

$$\begin{aligned}\mathrm{Var}[\widehat{L}_n] &\approx \frac{1}{n}\mathrm{Var}[\Psi L] + \mathrm{E}[\Psi]^2\mathrm{Var}[\epsilon_n] \\ &\approx \frac{1}{n}\mathrm{Var}[\Psi L] + \frac{\mathrm{Var}[\epsilon_1]}{(2-\alpha)n^\alpha} \\ &= O(n^{-1}) + O(n^{-\alpha}) = O(n^{-\alpha}). \qquad (6.40)\end{aligned}$$

BIAS The expected value of the error $\mathrm{E}[\epsilon_n]$ is modeled in Equation (6.28) as a function of the averaged bias introduced at each iteration $\mathrm{E}[\epsilon_j]$ (6.37). Computing the kernels' bandwidth $\mathcal{T}_j$ and R_{bj} at iteration j by expanding Equation (6.20) as a function of their initial value by we get

$$\begin{aligned}\mathcal{T}_j &= \mathcal{T}_1(j\ \alpha\ \mathrm{B}(\alpha,j))^{-\beta_\mathcal{T}}, \qquad (6.41)\\ R_{bj} &= R_{b1}(j\ \alpha\ \mathrm{B}(\alpha,j))^{-\beta_{Rb}}, \qquad (6.42)\end{aligned}$$

where $\mathrm{B}(x,y)$ is the Beta function. Using (6.41) and (6.42) in Equation (6.37) we can express $\mathrm{E}[\epsilon_j]$ as a function of the initial kernel bandwidths

$$\begin{aligned}\mathrm{E}[\epsilon_j] = &\ \mathrm{E}[\Phi]p_{R_b}\mathcal{C}^{ii}_{\mathcal{T}}\mathcal{T}_1^2\Theta(j^{1-\alpha})^{-2\beta_\mathcal{T}} \\ &+\mathrm{E}[\Phi]p_{\mathcal{T}}\mathcal{C}^{ii}_{1D}R_{b1}^{\ 2}\Theta(j^{1-\alpha})^{-2\beta_{Rb}} \\ &+\mathrm{E}[\Phi]\mathcal{C}^{ii}_{\mathcal{T}}\mathcal{C}^{ii}_{1D}\mathcal{T}_1^2R_{b1}^{\ 2}\Theta(j^{1-\alpha})^{-2(\beta_\mathcal{T}+\beta_{Rb})}. \qquad (6.43)\end{aligned}$$

Finally, we use $\sum_{j=1}^{n}\Theta(j^x) = n\ O(n^x)$ to plug Equation (6.43) into Equation (6.29) to get the asymptotic behavior of $\mathrm{E}[\epsilon_n]$ in transient progressive photon beams:

$$\mathrm{E}[\epsilon_n] = O(n^{1-\alpha})^{-2\beta_\mathcal{T}} + O(n^{1-\alpha})^{-2\beta_{Rb}} + O(n^{1-\alpha})^{-2(\beta_\mathcal{T}+\beta_{Rb})},$$

which, by using the equality $\beta_{Rb} = 1-\beta_\mathcal{T}$, becomes:

$$\begin{aligned}\mathrm{E}[\epsilon_n] &= O(n^{1-\alpha})^{-2\beta_\mathcal{T}} + O(n^{1-\alpha})^{2\beta_\mathcal{T}-2} + O(n^{1-\alpha})^{-2} \\ &= O(n^{1-\alpha})^{-2\beta_\mathcal{T}} + O(n^{1-\alpha})^{2\beta_\mathcal{T}-2}. \qquad (6.44)\end{aligned}$$

6.D MINIMIZING ASYMPTOTIC MEAN SQUARED ERROR

Using the asymptotic expression for variance and bias in Equations (6.40) and (6.44), we can express the AMSE (6.21) as

$$AMSE(\widehat{L}_n) = O(n^{-\alpha}) + \left(O(n^{1-\alpha})^{-2\beta_\mathcal{T}} + O(n^{1-\alpha})^{2\beta_\mathcal{T}-2}\right)^2. \qquad (6.45)$$

which is a function of the parameters α and $\beta_{\mathcal{T}}$. Given that the variance is independent of $\beta_{\mathcal{T}}$, we first obtain the optimal value for this parameter that yields the highest convergence rate of the bias $\mathrm{E}[\epsilon_n]$. We differenciate Equation (6.44), apply asymptotic simplifications and equating to zero, we obtain the optimal value $\beta_{\mathcal{T}} = 1/2$. By plugging this value in Equation (6.45), we obtain:

$$AMSE(\widehat{L}_n) = O(n^{-\alpha}) + O(n^{-2(1-\alpha)}). \tag{6.46}$$

Finally, by finding the minimum again with respect to α we get the optimal parameter $\alpha = 2/3$, which results in the optimal convergence rate of the AMSE for our transient progressive photon beams as

$$AMSE(\widehat{L}_n) = O(n^{-\frac{2}{3}}) + O(n^{-2(1-\frac{2}{3})}) = O(n^{-\frac{2}{3}}). \tag{6.47}$$

7 DEEPTOF: OFF-THE-SHELF REAL-TIME CORRECTION OF MULTIPATH INTERFERENCE IN TIME-OF-FLIGHT IMAGING

In this chapter we address one of the pathological problems in off-the-shelf Time-of-Flight range cameras: multipath interference (MPI). One of the main motivations that triggered this project was to demonstrate and exploit the potential of transient light transport simulation in transient imaging problems. The goal of this work is to correct MPI errors introduced by the assumption of single-bounce illumination. Our operational baseline is to avoid hardware modifications, and to keep real-time performance. To achieve this we propose a convolutional deep-learning approach to learn corrections of MPI. To circumvent the lack of labeled real data, we propose a hybrid approach real and synthetic data. Our hybrid scheme first learns to obtain lower-dimensional representations of real depth maps. Then, it is re-trained using synthetic labeled data to decode these representations to the *corrected* maps without MPI. We demonstrate that our method works in real time, and overcomes previous works in a wide variety of real and synthetic scenarios.

This project started during my two-month internship at Microsoft Research Asia in Beijing, China. It was partially developed there, and continued as a collaboration with them until its publication. The work was published in ACM Transactions on Graphics and presented at SIGGRAPH Asia 2017.

J. Marco, Q. Hernandez, A. Muñoz, Y. Dong,
A. Jarabo, M. H. Kim, X. Tong & D. Gutierrez
DeepToF: Off-the-Shelf Real-Time Correction of Multipath Interference in Time-of-Flight Imaging
ACM Transactions on Graphics, Vol.36(6)
(Presented at SIGGRAPH Asia 2017)

7.1 INTRODUCTION

Time-of-flight (ToF) imaging, and in particular continuous-wave ToF cameras, have become a standard technique for capturing depth maps. Such devices compute depth of visible geometry by emitting modulated infrared light towards the scene, and correlating different phase-shifted measurements at the sensor. However, ToF devices suffer from multi-path intereference (MPI): a single pixel records multiple light reflections, but it is assumed that all light reaching it has followed a direct path (see [113] for details). This introduces an error on the captured depth, which reduces the applicability of ToF cameras.

To compensate for MPI, most previous works leverage additional sources of information, such as coded illumination or multiple modulation frequencies that lead to different phase-shifts, from which indirect light might be disambiguated. This requires either hardware changes (e.g. modifying the built-in illumination, or using sensors that can handle multiple modulated frequencies), or multiple passes with a standard ToF camera. Other single-modulation approaches simulate an estimation of the ground-truth light transport, then compensate the captured MPI using information from such simulation. Such approaches, while working with any out-of-the-box ToF sensor, typically require several minutes for a single frame, and might lead to errors when the simulation is not accurate.

Our work aims to lift these limitations: We present a novel technique to correct MPI using an *unmodified, off-the-shelf camera*, with a *single frequency*, and in *real time*. A key observation is that, since both the camera and the infrared emitter are co-located and share almost the same visibility frustrum, most of the MPI information is actually present in image space. Furthermore, from the discretized geometry given by a depth map, light transport at each pixel can only be estimated as a linear combination (with unknown weights) of the contributions from the rest of the pixels. This linear process can be represented as a spatially-varying convolution in image space, with unknown convolution filters. This motivates our design of a convolutional neural network (CNN) to obtain such filters.

However, suitable ToF datasets that include depth with MPI and its corresponding ground-truth reference do not exist, and capturing such dataset is not possible with current devices. To overcome this, we synthesize such data using an existing physically-based transient light transport renderer (Section 7.5), extending it with a ToF camera model. Using this model, we introduce MPI in the simulated depth estimation, and compare it with the reference depth. Our network uses the synthetic data for training, takes depth with MPI as input, and returns a corrected depth map. In particular, since the input and output have the same resolution, we design an encoder-decoder network. While amplitude or phase-shifted images could provide additional information to the network, they are often uncalibrated and highly dependent on the device characteristics. We therefore use depth as the only reliable input to our network, providing a real-time solution that is robust even for single-frequency ToF devices.

This approach introduces two new challenges. First, generating synthetic data is time-consuming, and thus the simulated dataset is unlikely to be large and diverse enough to avoid overfitting. Second, although we carefully analyze our synthetic data to make sure that it is statistically similar to real-world data, it might still be too perfect, lacking for instance subtle differences due to imperfections in the sensor or the emitter. We address this by leveraging the fact that the input and output depths must be structurally similar, and devising a two-stage training. The first stage is a convolutional autoencoder (CAE) from real-world captured data, which requires no ground-truth reference. This tackles both challenges, since it allows us to use large datasets with real-world imperfect data for training. This first stage thus trains the encoding filters of the network as a feature dictionary learned from structural properties of ToF depth images (Section 7.6.1).

The second stage provides supervised learning for the regression with the synthetic dataset as reference, which accounts for the effect of MPI (input and output are now different), and feeds the decoder (while the encoder remains unmodified). We treat the effect of MPI as a residue, and therefore model this second stage as supervised residual learning (Section 7.6.2).

We analyze the performance of our approach in synthetic and captured ground-truth data, and compare against previous works, showing favorable results while being significantly faster. Finally, we demonstrate our technique in-the-wild, correcting MPI from depth maps captured by a ToF camera in real time. In summary, we make the following contributions:

- A two-stage training strategy, with a convolutional autoencoder plus a residual learning approach. It leverages statistical knowledge from real captured data with no ground truth, and then compensates the error from synthetic data (which includes ground truth).
- A synthetic ToF dataset of scenes sharing similar statistical properties as real-world scenes. For each scene, we provide both MPI-corrupted and ground-truth depth. We believe this data is much needed, and we hope our dataset can help future works.
- A trained network that compensates multipath interference from a single ToF depth image in real time, which outperforms previous algorithms even using such minimal input.

Our training dataset and trained network are publicly available online at the project page[1].

7.2 RELATED WORK

Convolutional neural networks (CNNs) have been widely used for many image-based reconstruction tasks, such as intrinsic images, nor-

[1] http://webdiis.unizar.es/ juliom/pubs/2017SIGA-DeepToF/

mal estimation, or depth recovery. Here we only focus on CNN-based depth reconstruction methods that are closely related to our work. We refer to Jarabo et al.'s recent survey [113] for a complete overview on transient and ToF imaging, and to Goodfellow et. al.'s book [72] for other deep learning techniques and their applications.

CNN-BASED DEPTH RECONSTRUCTION A set of methods derive depth from multi-view images. Žbontar and LeCun [287] trained a CNN for computing the matching cost of stereo image pairs. Kalantari et. al. [134] exploited a CNN to estimate the disparity between sparse light-field views, and fed the result to another CNN to interpolate the light-field views for novel view synthesis. Different from these multi-view methods, we reconstruct a depth image from a single snapshot captured by a monocular ToF camera.

Other methods estimate depth from a single RGB image. Eigen et. al. [48, 49] proposed an end-to-end CNN in which a coarse depth image is first recovered, then progressively refined. In each step, the coarse depth is upsampled and combined with fine scale image features. Based on this approach, several works formulate the generated depth as a conditional random field (CRF), and then refine it with the help of color image segmentation [169, 175, 261], or multi-resolution depth information generated by intermediate CNN layers [282]. Although these methods improve the accuracy of the result, the CRF optimization is expensive and slow. Most recently, Su et. al. [242] trained a CNN with a synthetic dataset rendered from a large dataset for reconstructing a low-resolution 3D shape from a single RGB or RGB-D image. Different from these methods, we do not rely on additional sources of information. Moreover, we have also developed an efficient scheme that combines both unlabeled real ToF images and labeled synthetic data for CNN training. This can efficiently generate a full-resolution depth map at a rate of up to 100 frames/second.

MULTIPATH INTERFERENCE Several works take advantage of inputs with several amplitudes and phase images through multiple modulation frequencies. This input can be translated into multipath interference correction through optimization [43, 56], closed-form solutions with inverse attenuation polynomials [70], spectral methods [54, 147], sparse regularization [18], or through modeling indirect lighting as phasor interactions in frequency space [78]. Although these techniques are efficient for some devices that can capture a few frequencies simultaneously, such as Kinect V2 [54], they require multiple passes for single-frequency ToF cameras.

Other approaches deal with multipath interference by adding or modifying hardware. Wu and colleagues [278] decompose global light transport into direct, subsurface scattering, and interreflection compo-

Table 7.1: Comparison between the different existing techniques that address the problem of multipath interference, including required input, tests shown in the paper (material types and variability of scenes) and execution time. Our approach performs in real time while requiring the most reduced input. Furthermore, we show a greater scene variability in terms of materials and geometries.

Work	Input	Tested materials	Scenes	Reported time
Fuchs [57]	Single frequency	Single albedo	Few planes	$\approx$ 10 minutes
Dorrington et al. [43]	Multiple frequencies	Multiple albedos	Few planes	Unreported
Godbaz et al. [70]	Multiple frequencies	Multiple albedos	Few planes	Unreported
Fuchs et al. [59]	Single frequency, user input	Multiple albedos	Few planes	$\approx$ 150 seconds
Heide et al. [92]	Multiple freqs. hardware mods.	Multiple albedos	Several	$\approx$ 90 seconds
Kadambi et al. [130]	Single freq., custom code	Multiple albedos	Several	$\approx$ 4 seconds
Kirmani et al. [147]	Multiple frequencies	Single albedo	Few planes	Real time
Freedman et al. [56]	Multiple frequencies	Mult. albedos, specular	Few scenes	31.2 ms
Bhandari et al. [18]	Multiple frequencies	Transparency	Few planes	Unreported
O'Toole et al. [206]	Multiple freqs., structured light	Multiple albedos	Several	Unreported
Jiménez et al. [124]	Single frequency	Multiple albedos	Many	Several minutes
Peters et al. [215]	Multiple frequencies	Mult. albedos, specular	Scene, video	18.6 fps
Gupta et al. [78]	Multiple frequencies	Multiple albedos	Few scenes	Unreported
Naik et al. [197]	Single freq. structured light	Multiple albedos	Few planes	"Not real time"
Feigin et al. [54]	Multiple frequencies	Transparency	Several	"Efficient"
Our work	Single depth	Multiple albedos	Many	10ms

nents, leveraging the extremely high temporal resolution of the femto-photography technique [259]. Modified ToF sensors allow to reconstruct a transient image from multiple frequencies [92, 215]. Other techniques include custom coding [130], or combining ToF sensors with structured light projection [197, 206]. In contrast, our method works with just an out-of-the-box phase ToF sensor, without any modifications.

Some techniques remove multipath interference from a single frequency (a single amplitude and depth image) by estimating light transport from the approximated depth. One of the approximations considers a single indirect diffuse bounce (assuming constant albedo) connecting all pairs of pixels on the scene [57]. This has been later extended to multiple diffuse bounces and multiple albedos, by adding some user input [59]. Last, an optimization algorithm over depth space with path tracing has also been presented [124]. While these approaches manage to compensate multipath interference from input obtained with any ToF device, they are very time-consuming; moreover, given the sparse input and the assumptions simplifying the underlying light transport model, they are unable to completely disambiguate indirect light in all cases. In contrast, given even less information (a single depth map, without amplitude) our work is able to compensate multipath interference for varied and complex geometries, with different albedos, and in real time.

Table 7.1 summarizes these approaches, and compares them to our method. We list the required input, the variability of the tested scenes (including albedo and geometry) and the execution time. Our method works on a large range of scenes, with a just a single depth map as input, and yields real time performance.

7.3 PROBLEM STATEMENT

TOF DEPTH ERRORS Using four phase-shifted measurements $\mathbf{c}_{1\ldots4}$, ToF devices compute the depth z at every pixel p as

$$z = \frac{c\,\phi}{4\pi f_{\omega_R}} \tag{7.1}$$

$$\phi = \arctan\left(\frac{\mathbf{c}_4 - \mathbf{c}_2}{\mathbf{c}_1 - \mathbf{c}_3}\right), \tag{7.2}$$

where f_{ω_R} is the device modulation frequency, c is the speed of light in a vacuum, and ϕ is the phase of the wave reaching a pixel p. This model works under the assumption of a single impulse response from the scene, therefore assuming

$$\mathbf{c}_i(p) = x(p)e^{2\pi j\phi(p)\frac{f_{\omega_R}}{c}+\theta_i}, \tag{7.3}$$

with $j = \sqrt{-1}$, $x(p)$ the amplitude of the wave, and θ_i the phase shift of measurement $\mathbf{c}_i$. However, given indirect illumination, the observed

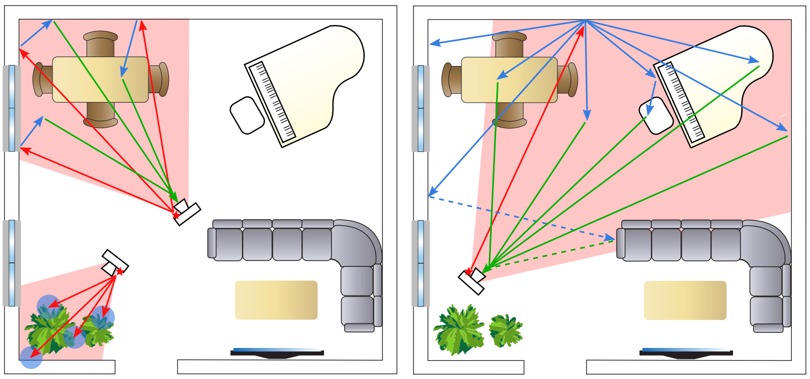


Figure 7.1: ToF MPI at different scales, showing IR emission (red), indirect bounces (blue) and observed radiance (green). Left: Observed second-bounce illumination occurs mostly from reflections on visible geometry due to shared light-camera visibility frustum (camera facing the table), while higher-order bounces usually have a significant impact in the locality of the observed points (e.g. camera facing the plants). Right: Large objects such as a wall may cast a significant indirect component over the whole scene when captured from afar, while longer paths from out-of-sight geometry (discontinuous) create negligible MPI due to attenuation.

pixel may receive light from paths other than single-bounce direct light, which leads to

$$\hat{\mathbf{c}}_i(p) = \mathbf{c}_i(p) + \int_P x(\overline{p})\, e^{2\pi j \phi(\overline{p}) \frac{f\omega_R}{c} + \theta_i} d\overline{p}, \tag{7.4}$$

where P is the space of all the light paths $\overline{p}$ reaching pixel p from more than one bounce; the amplitude $x(\overline{p})$ and phase delay due to light time-of-flight $\phi(\overline{p})$ are now functions of the path $\overline{p}$. The effect of multiple bounce paths if often ignored in ToF sensors obtaining approximate measures $\mathbf{c}_i(p) = \hat{\mathbf{c}}_i(p)$ and therefore leading to a depth estimation error, the multipath interference (MPI).

MPI OBSERVATIONS In ToF range devices, the light source is typically co-located with the camera, sharing a similar visibility frustum. As we illustrate in Figure 7.1 (camera facing the table), this implies that most of the MPI due to second-bounce indirect illumination comes from actual visible geometry. Previous works have leveraged this by taking into account only second-bounce illumination [43, 57], or by ignoring non-visible geometry [124]. Higher-order indirect illumination might come from non-visible geometry, but due to the exponential decay of scattering events and quadratic attenuation with distance, light paths of more than two bounces interfere mainly in the local neighborhood of a point (see Figure 7.1, camera facing the plants). These observations suggest that most of the information on multipath interference from a scene is available in *image space*, where Equation (7.4) is discretized into a summatory and can be modeled as a spatially-varying convolution. Please refer to Appendix 7.A for a more detailed derivation of such spatially-varying convolution model. Last, since the effect of multipath interference does not eliminate major structural features of a depth map, the incorrect depth and the reference depth are structurally similar.

7.4 OUR APPROACH

Given that MPI can be expressed as a spatially-varying convolution, MPI compensation could be modeled as a set of convolutions and deconvolutions in depth space. This in turn suggests that MPI errors could in principle be solved designing a convolutional neural network (CNN). Specifically, since incorrect and correct depths are only slightly different (but structurally similar), using a convolutional autoencoder (CAE) would be a tempting solution. A convolutional autoencoder is a powerful tool which takes the same input and output to learn hidden representations of lower-dimensional feature vectors by *unsupervised* learning, resulting in two symmetric networks: an encoder and a decoder. This allows to build a deeper network architecture, and preserves spatial locality when building these representations [182]. The lower-dimensional feature vectors retain the relevant structural information on the input and eliminate existing errors, effectively returning the restored (reference) image. Recently, CAEs have been successfully used in many vision and imaging tasks (e.g. [32, 44]).

A straight-forward CAE, nevertheless, cannot be applied to our particular problem: as the errors introduced by MPI are highly correlated with the reference depth to be recovered, we need such ground-truth reference for training. However, a large enough labeled dataset (i.e., pairs of MPI-corrupted depth and its corresponding ground-truth depth), needed for training, does not exist. Although real-world, ToF depth images are widely available, measuring their ground-truth depth maps is a non-trivial task. On the other hand, rendering time-resolved images from synthetic scenes is extremely time-consuming, and the results would only cover a small portion of real-world scene variations.

We propose a two-step training scheme to infer our convolutional neural network from both *unlabeled* real depth images, and labeled synthetic depth image pairs (with and without MPI). Our method is inspired by super-resolution methods based on overcomplete dictionaries and sparse coding [285]. Figure 7.5 shows an overview of our network: We first learn an encoder network as a depth prior through traditional unsupervised CAE training, in which unlabeled real depth images (with unknown errors) are used as both input and output. The

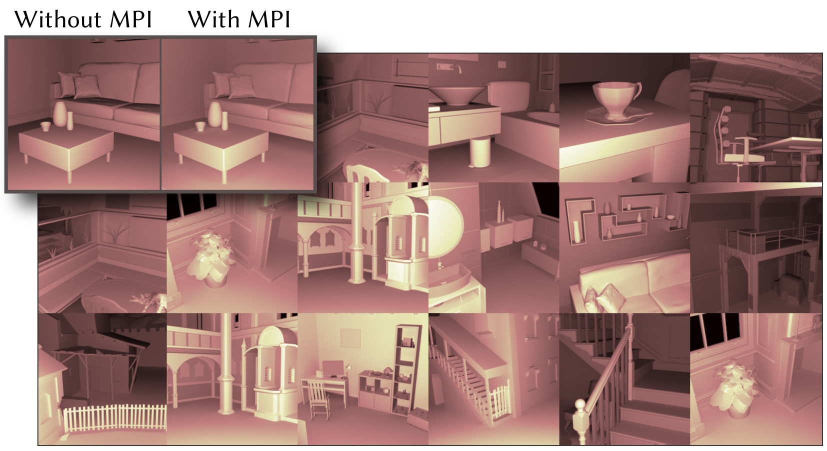


Figure 7.2: Representative sample of the scenes (amplitude) rendered with the ToF model to generate depth images with MPI. The top-left image shows the same scene with and without MPI.

resulting encoder allows us to obtain lower-dimensional feature vectors of incorrect depth images. In our second step, different from sparse coding where original signals are reconstructed as a linear product of dictionary atoms, we train a decoder that can reconstruct the reference depth map from such feature vectors. To this end, we keep the encoder network unchanged and cascade it with a residual decoder network. The weights of the decoder network are learned from the synthetic depth pairs via supervised CNN training.

Our network therefore only takes the depth image with MPI as input, and outputs a depth map without the effects of MPI. We do not feed our network with any other ToF information, such as pixel amplitudes or phase images, because these properties highly depend on specific ToF camera settings, and are unstable. By using only depth as input, our solution is robust using off-the-shelf ToF devices that operate with a single frequency.

In the following two sections, we first introduce our dataset for training (Section 7.5), then describe in detail our network and our two-step training scheme (Section 7.6).

7.5 TRAINING DATA

To obtain accurate pairs of incorrect (due to MPI) and ground-truth reference depth maps, we simulate the response of the ToF lighting model using the publicly-available, time-resolved bidirectional path tracer of Jarabo et al. [111]. This allows us to obtain four phase images

[Equation (7.2)], and from those the MPI-distorted depth estimation of a ToF camera [Equation (7.1)].

Existing ToF devices generally use square modulation functions instead of perfect sinusoidal ones; this introduces additional errors on the depth estimation (*wiggling*), although ToF cameras do account for these errors and compensate them in the final depth image. The wiggling effect and its correction are specific for each camera, and in general information is not provided by manufacturers. We therefore use ideal sinusoidal functions to avoid introducing non-MPI-related error sources. Ground-truth depths are straightforward to obtain from simulation.

DATASET We simulated 25 different scenes with varying materials, using six different albedo combinations between 0.3 and 0.8, and rendered from seven different viewpoints, at 256×256 resolution—similar to what ToF cameras yield—, computed with up to 20 bounces of indirect lighting. From these we obtained 1050 depth images with MPI (which we flip and rotate to generate a total of 8400) and their respective reference depths. The geometric models of the scenes were obtained from three different free repositories[2]. Some examples can be seen in Figure 7.2. In the remaining of this chapter we refer to this as the **synthetic dataset**.

Training a network from scratch requires a sufficiently large labeled dataset. However, generating it is very time-consuming, so using only synthetic data for our purposes is highly unrealistic. We therefore gather an additional dataset of 6000 *unlabeled*, real depth images (48000 with flips and rotations) from public repositories [138, 235, 280], and use them to pre-initialize our network; we refer to this as the **real dataset**. Learning representations of unlabeled real depths will later improve depth corrections from our smaller synthetic labeled dataset (Section 7.6).

Figure 7.3 compares a real scene captured with a ToF camera (thus including MPI) with our ToF simulation, showing a good match. Additionally, in Appendix 7.B we perform a statistical analysis on both datasets, to assess their similarity.

QUANTITATIVE ANALYSIS OF MPI ERRORS Figure 7.4 shows the error distributions across our entire synthetic dataset. Note that while previous works have addressed local errors of just a few centimeters in small scenes, our data indicates that the global component can introduce much larger errors, with an average error of 26 cm (red line in the top-left histogram) for scenes up to 7.5 m. On average, 12% of

[2]https://benedikt-bitterli.me/resources/
http://www.blendswap.com/
https://free3d.com/

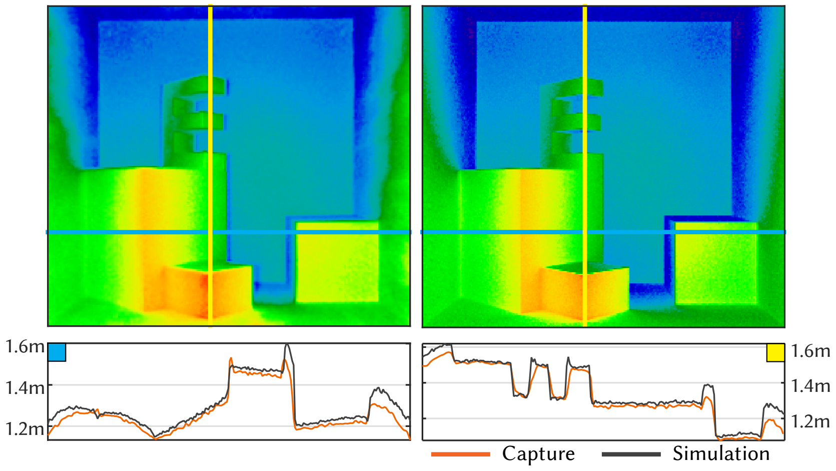


Figure 7.3: Depth map with MPI of a real scene (top left) captured by a ToF camera, and its corresponding synthetic model (top right). The bottom row shows the depth profiles for the horizontal (left) and vertical (right) lines, showing a good agreement.

the measured ToF depth corresponds to MPI (bottom-left histogram). The bivariate histogram relating relative error and observed depth (bottom-right) additionally shows that the average remains constant at around 10% for most measured depths, being larger for smaller depths.

7.6 NETWORK ARCHITECTURE

We now describe how to train our network, following our two-stage scheme with the real dataset (during autoencoding) and our synthetic dataset (during supervised decoding).

7.6.1 *Stage One: Autoencoder*

We train our convolutional autoencoder using the real dataset, containing 48000 depth images with unknown errors. We use this incorrect depth as input and output for this unsupervised training, and use the synthetic dataset (with MPI) as its validation set. With this stage we pre-initialize the network so the encoder (Figure 7.5, top, gray blocks) is able to generate lower-dimensional feature vectors (Figure 7.5, top, blue block) for both real and synthetic depth maps. Training and validation curves of this stage are shown in Figure 7.6. Once we train the parameters of the encoder, we freeze its convolutional layers and update only the decoder layers in the second stage.

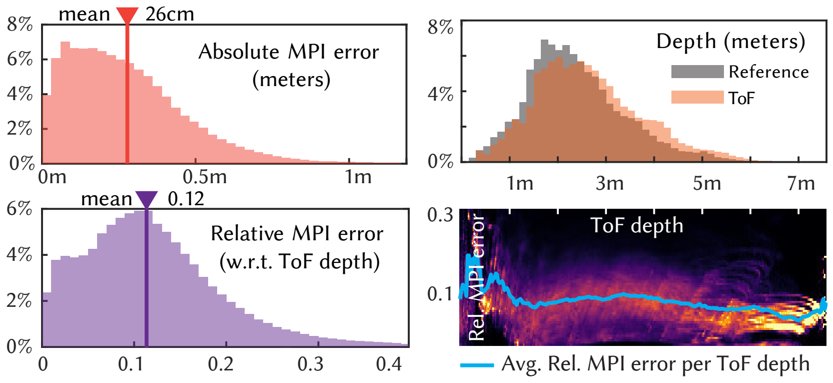


Figure 7.4: In reading order: Absolute MPI error; depth distribution for reference and ToF depths, at a modulation of 20MHz (i.e. maximum unambiguous distance of 7.5m); relative error with respect to measured ToF depth; and bivariate histogram showing relative MPI error density per measured ToF depth. Measured ToF depths contain an average error of 12%. The blue line in the bivariate histogram shows that the average relative MPI error per ToF depth remains around 10% for most measured depths.

NETWORK PARAMETERS In the encoding stage, we apply sets of two 5×5 convolutions with a two-pixel padding, and a stride of two pixels to progressively reduce the size of the convolutional inputs to each layer. This helps to effectively combine and find features at different scales. We perform this operation at six scales, applying pairs of convolutions over features of 256×256 pixels (input), down to 8×8 (innermost convolution pair, last encoding layer). In the decoding stage, we perform upsampling and 5×5 convolutions with two-pixel padding, starting from the encoder output (see Figure 7.5 top, red arrows) until we reach the output resolution 256×256.

We have tested our network without this pre-initialization step, feeding it directly with synthetic labeled data. The results show that the network loses the ability to generalize, arbitrarily decreasing the accuracy in the validation dataset, as presented in Section 7.7.

7.6.2 *Stage Two: Supervised Decoder*

In the second stage, we freeze the encoder layers, and train the decoder through supervised learning using our synthetic dataset. We introduce incorrect depth (with MPI) as input, and target reference depth (without MPI) as output. We use 80% of our synthetic dataset for training, and the remaining 20% for validation. Given that the encoder performs downsampling operations to detect features at multiple scale levels, full resolution outputs (256×256) are significantly blurred. We thus add

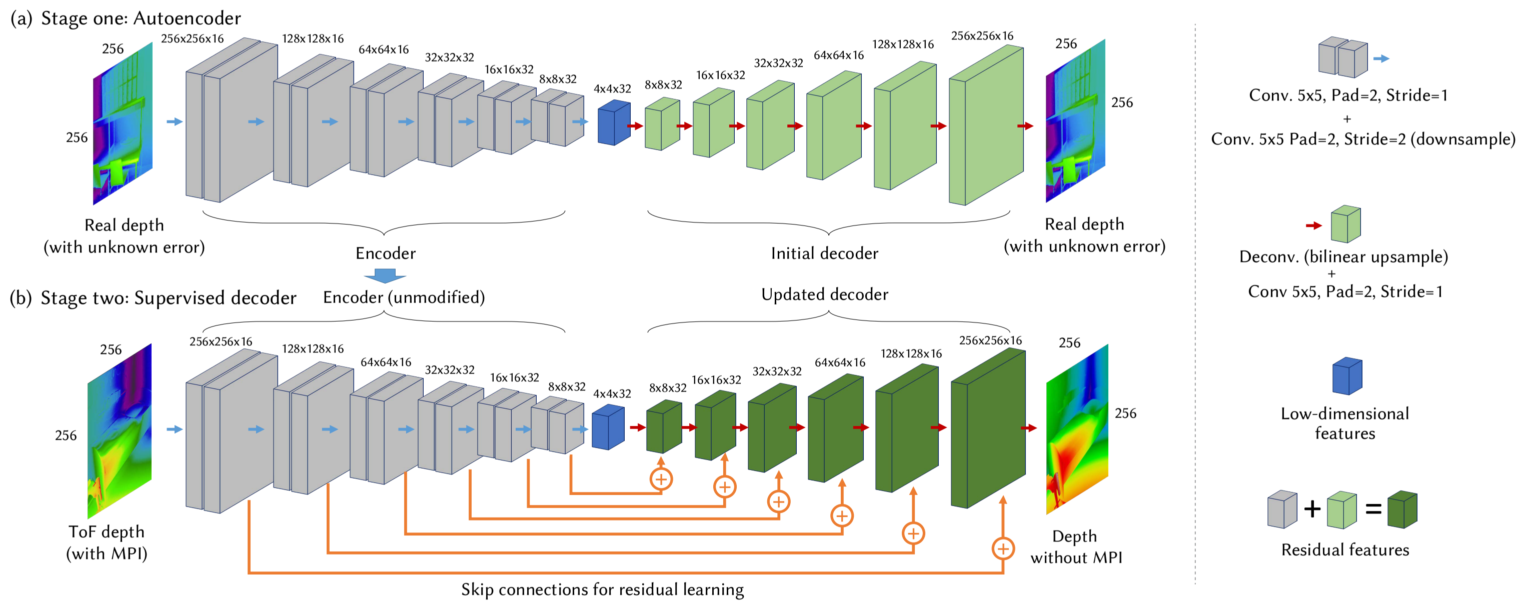

(a) Stage one: Autoencoder
256
256
256x256x16
128x128x16
64x64x16
32x32x32
16x16x32
8x8x32
4x4x32
8x8x32
16x16x32
32x32x32
64x64x16
128x128x16
256x256x16
Real depth (with unknown error)
Encoder
Initial decoder
Real depth (with unknown error)
(b) Stage two: Supervised decoder
Encoder (unmodified)
Updated decoder
ToF depth (with MPI)
Depth without MPI
Skip connections for residual learning
Conv. 5x5, Pad=2, Stride=1 + Conv. 5x5 Pad=2, Stride=2 (downsample)
Deconv. (bilinear upsample) + Conv 5x5, Pad=2, Stride=1
Low-dimensional features
Residual features

Figure 7.5: Our two-stage training process for our regression network architecture. The first training stage learns a convolutional autocoder (an encoder-decoder pair) to learn lower-dimensional representations of depth. We use incorrect (with MPI) depth as both input and output in this unsupervised training. In the second, supervised training stage we input both incorrect depth with MPI and depth without MPI, and add skip connections for residual learning while fixing the weights of the encoder. This supervised training allows us to update the weights of a reconstruction decoder. Our final network is the pair of the original encoder and the updated decoder, working as a regression network. Please refer to the text for a more complete description.

symmetric skip connections to mix detailed features of the encoding convolutions (which stay unchanged) to their symmetric outputs in the decoder (Figure 7.5, bottom). Since we observed that the difference between depth with MPI (input) and reference depth (output) is on average 12%, we treat MPI as a residue [88] by performing element-wise additions between the upsampled features and the skipped ones. Training and validation curves of this stage are shown in Figure 7.6.

In principle, concatenation of skipped features (instead of simpler element-wise additions) could create more complex combinations with upsampled features using additional learned filters. However, in our results we observed that our residual approach performs equally well (even slightly better, see Section 7.7.1) while yielding a 30% smaller network model, reducing also execution time.

7.6.3 *Implementation Details*

We have implemented our network in Caffe, and trained it on an NVIDIA GTX 1080. Our network takes the input depth without applying any normalization. Following previous works using CNNs, all convolutional layers are followed by a batch normalization layer, a scale and bias layers, and a ReLU activation layer, in that order. For training, we use the Adam solver [145] for gradient propagation. The learning rate was set to $1 \cdot 10^{-4}$, and adjusted in a stepped fashion in steps between $1 \cdot 10^{-3}$ and $1 \cdot 10^{-5}$, to avoid getting stuck in a plateau, while our batch size is set to 16 to maximize memory usage. Our resulting network performs MPI corrections for a single frame in 10 milliseconds. Additional details on the definition of both the network and training, including the input sources, can be found in the project webpage.

7.7 RESULTS AND VALIDATION

In this section, we first analyze other alternative, simpler networks, showing how they yield inferior results. We then compare our results against existing methods using off-the-shelf cameras, and thoroughly validate the performance of our approach in both synthetic and real scenes, including video in real time. Real scenes were captured with a PMD CamCube 3.0, which provides depth images at 200×200 resolution, and operates at 20MHz with 7.5 meters of maximum unambiguous depth.

7.7.1 *Alternative Networks*

We test three alternatives to our CNN: (1) suppressing the pre-initialization autoencoder stage by directly training an encoder-decoder with syn-

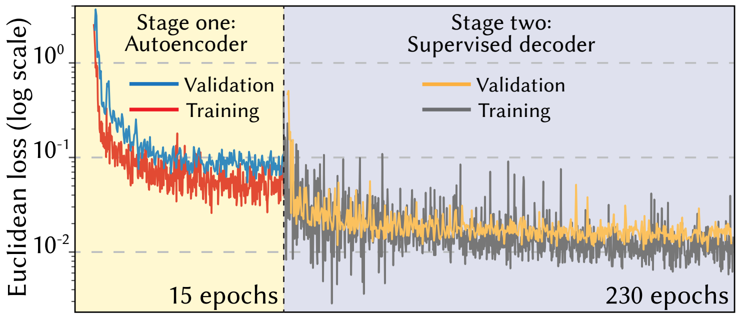


Figure 7.6: Learning curves for our two-stage scheme. The first stage (left) learns lower-dimensional representations of real depths using an autoencoder. The second stage (right) learns MPI corrections in the decoder by training with our labeled synthetic data. Note that the first stage converges quickly and provides a good starting point for the second stage, which uses a different training set.

thetic labeled data; (2) removing the residual skip connections; (3) substituting residual connections by concatenated connections. Figure 7.8 shows how our autoencoder with the residual learning approach yields better results with respect to these other alternatives, with better generalization and smaller network size. By computing R^2 scores between the depth predicted by each network and the target depths across the whole set of images, we observe that, without the autoencoding stage, the images present a lower average score than our results (see Figure 7.8, top-right table). Also, the variance of the per-image mean absolute error without pre-initialization triplicates the variance of our residual network errors, leading to more unstable accuracy. Concatenating skip connections worsens results slightly, while additionally making the network about 30% larger, due to the need to learn more parameters to combine the additional features. Last, removing residual skip connections avoids enriching upsampled features with high resolution features from the encoder layers, producing blurry outputs and therefore a much higher error.

7.7.2 *Comparison with Previous Work*

In Figure 7.9 we compare our solution to previous works requiring no hardware modifications, and using a single frequency [57, 124]. We use both a synthetic and a real scene. Fuchs' approach [57] results in a noisy estimation due to the discretization of light transport, taking around 10 minutes to compute. Jimenez et. al's technique [124] is hindered by the geometrical complexity of the scenes, taking around one hour for a lower resolution image of 100×100; we were unable

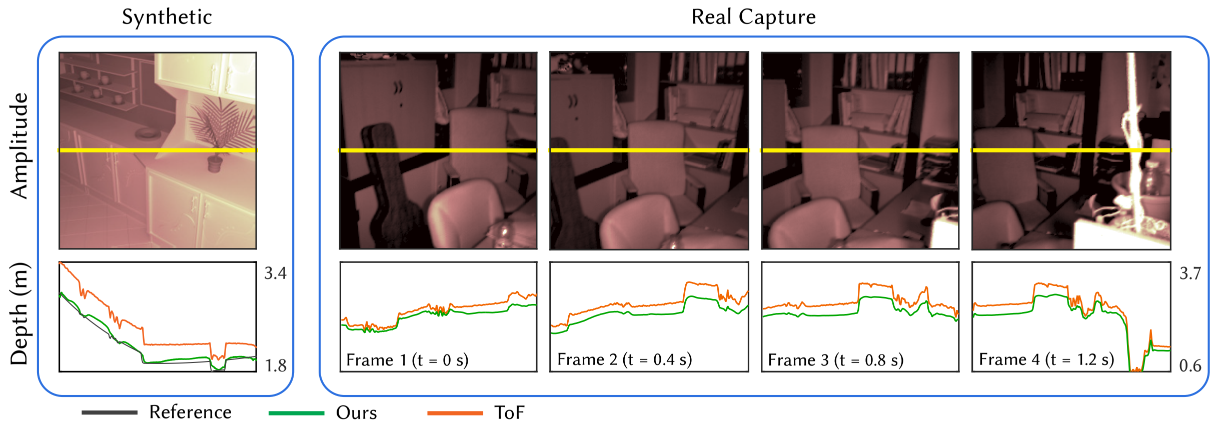
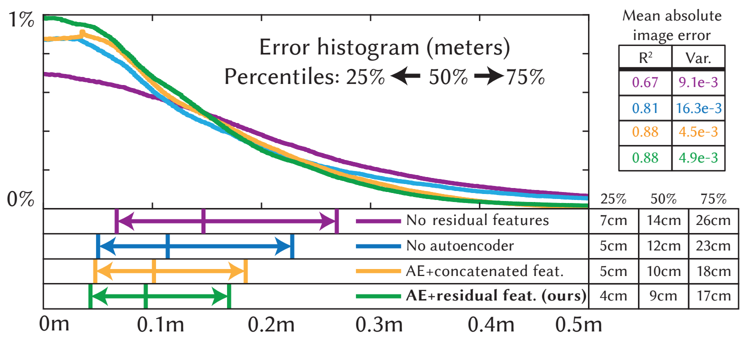


Figure 7.7: The left image shows how our system corrects one of our synthetic scenes from the validation set. Our method (green) provides a better fit to ground truth data (black) than ToF depth from the camera (orange). In the sequence to the right, we illustrate how our system allows for real-time MPI compensation (10 milliseconds per frame) in real scenes captured with ToF devices, preserving temporal consistency. Please refer to the supplemental video[3] for more results.



Mean absolute image error

R^2	Var.
0.67	9.1e-3
0.81	16.3e-3
0.88	4.5e-3
0.88	4.9e-3

	25%	50%	75%
No residual features	7cm	14cm	26cm
No autoencoder	5cm	12cm	23cm
AE+concatenated feat.	5cm	10cm	18cm
AE+residual feat. (ours)	4cm	9cm	17cm

Figure 7.8: Error distributions for different network alternatives: without residual connections (purple), without the autoencoding pre-initialization (blue), with concatenated skip connections (yellow), and our residual approach (green). Average R^2 across all predicted depth images shows that our residual learning with autoencoding pre-initialization reaches the best error distributions in the results, in terms of accuracy and low variance. The percentiles show that our approach presents also the best error distribution.

to compute larger images due to high memory consumption (about 60GB). Moreover, the results are very close in general to the input ToF captures, including MPI errors and several outlier pixels. Our results are significantly closer to the reference, eliminating MPI errors, while being orders of magnitude faster.

7.7.3 *Synthetic Scenes*

From the synthetic dataset, a total of 213 scenes were used for the validation set (augmented to 1704 with flips and rotations). As Figure 7.10 (left) shows, our method yields a much better error distribution. Figure 7.11 shows a comparison of simulated ToF depth, our MPI-corrected depth, and the reference depth images. Our CNN preserves details while significantly mitigating MPI errors.

Additionally, in Figure 7.12 we compare the errors for five albedo combinations of three different scenes by randomly varying each object's reflectance between 0.3 and 0.8. It can be observed how our network is robust to these variations, consistently correcting depth errors due to MPI. Note how even under strong albedo changes on large flat objects (e.g., the cabinet in the first row, or the tabletop in the second) our network successfully recovers the correct depth.

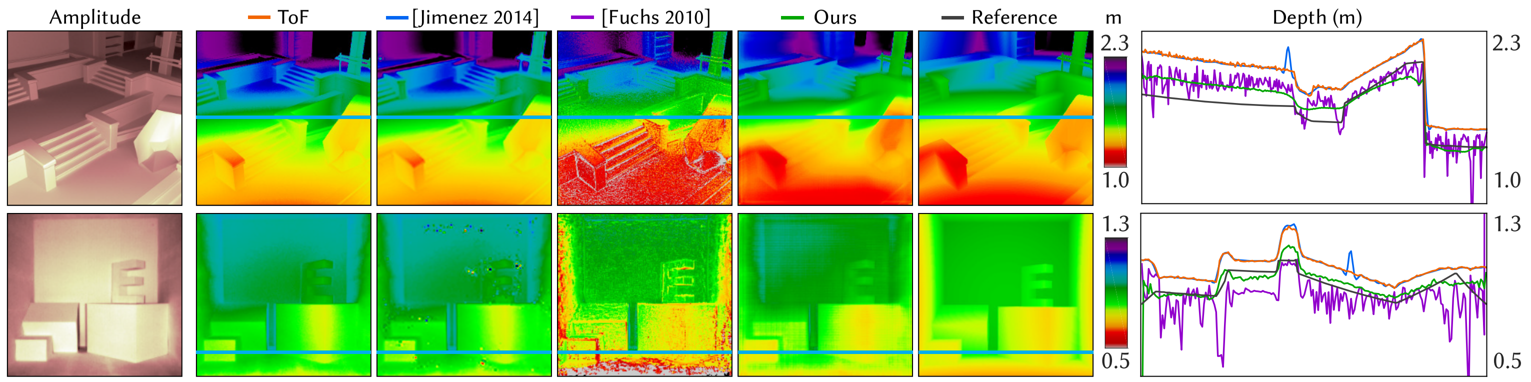


Figure 7.9: Comparison with previous works. Top row: synthetic scene. Bottom row: real scene. Jimenez et. al' approach [124] is hindered by the geometrical complexity of the scenes, and fails to correct MPI since the optimization converges to a local minimum. Fuchs' technique [57] is closer to the reference, but is very noisy and greatly diverges in some regions. Our approach is the closest to the reference, as shown both in the depth maps and the graphs plotting the blue scanlines.

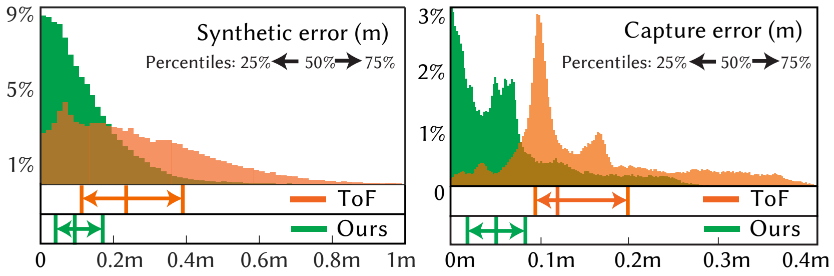


Figure 7.10: Per-pixel distributions of absolute error for the synthetic validation dataset (left), and real dataset with measured ground truth (right). For each distribution, three percentiles (25%, 50% and 75%) are marked below. Our results clearly present a better error distribution.

7.7.4 *Real Scenes*

We now analyze the performance of our method in real scenes, captured with a PMD camera. We first show results on controlled scenarios with combinations of V-shapes, panels and a Cornell box, and then more challenging captures in the wild. The lens distortion of all the captures was corrected using a standard calibration of the intrinsic camera parameters, using a checkerboard pattern and captures at different distances [23, 174].

CORNELL BOX AND V-SHAPES We created different setups combining a Cornell box structure and V-shapes with flat panels (see Figure 7.13, left column). We accurately measured the geometry of these scenes, to create corresponding synthetic reference images for a quantitative analysis. The Cornell box dimensions were 600×500×640 mm, with additional panels from 400 mm to 1200 mm. The PMD camera was placed at multiple distances from 0.5 to 2.4 meters. We added several geometric elements to the scenes: three prisms with different dimensions, and a cardboard letter E, in order to add extra sources of MPI error. The surfaces of the Cornell box, two panels, and the smaller shapes were painted twice with a 50-50 mixture of barium sulfate and white matte paint, providing a good trade-off between durability and high-reflectance diffuse surfaces [150, 209]. Note that this mixture has an albedo of approximately 0.85, leading to large MPI and thus ensuring very challenging scenarios. Figure 7.13 (middle and right) shows the results of the captured depth and our corrected result. We compensate most of the MPI errors in both scenes, approximating depth much closer to the reference than the ToF camera. We can observe on the error distributions (right histogram in Figure 7.10) how our CNN

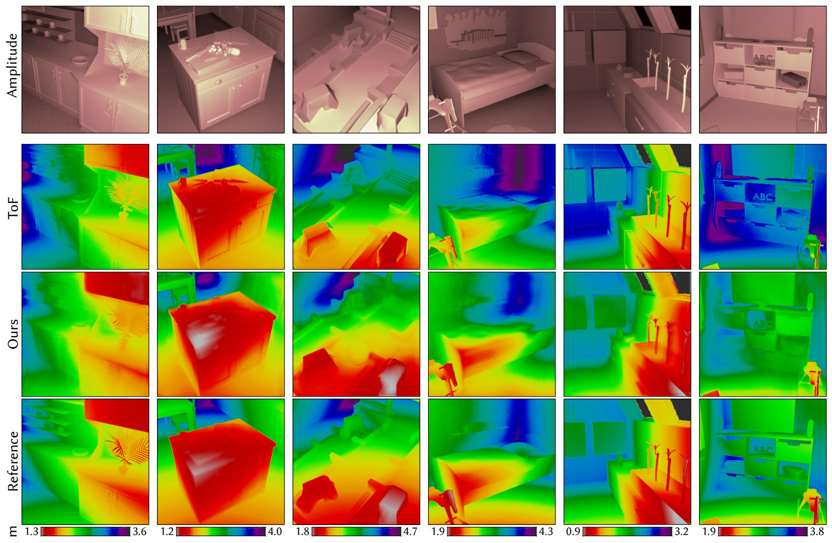


Figure 7.11: Validation results for synthetic scenes (validation set) at varying distances between 0.5 and 7 meters. Top row shows ToF amplitude. Second, third and fourth rows show ToF depth with MPI, our estimated depth, and reference depth (without MPI). Our solution manages to correct MPI errors in a wide range of scenes while preserving details.

manages to keep 50% of the per-pixel errors under 5 cm, while 75% of the errors in the PMD captures are *over* 9 cm.

SCENES IN THE WILD We now analyze several in-the-wild scenes, to illustrate the benefits of our approach in non-controlled conditions. The results are shown in Figure 7.14. We can see how our network successfully suppresses MPI in all cases, while still preserving details thanks to our residual learning approach. The magnitude of our MPI corrections is proportional to the measured ToF distance. This follows our observations in the error analysis (see Figure 7.4), where larger distances tend to yield larger errors since the relative error oscillates at around 10%.

7.7.5 *Video in Real Time*

Given the speed of our approach we are also able to process depth videos in real time. Regarding temporal coherence, we leverage the fact that our input (incorrect depth) is quite stable between frames, so our network produces temporally coherent results without explicitly enforcing it. Other inputs such as amplitude and/or phase could show less stability, compromising this temporal coherence. We show some

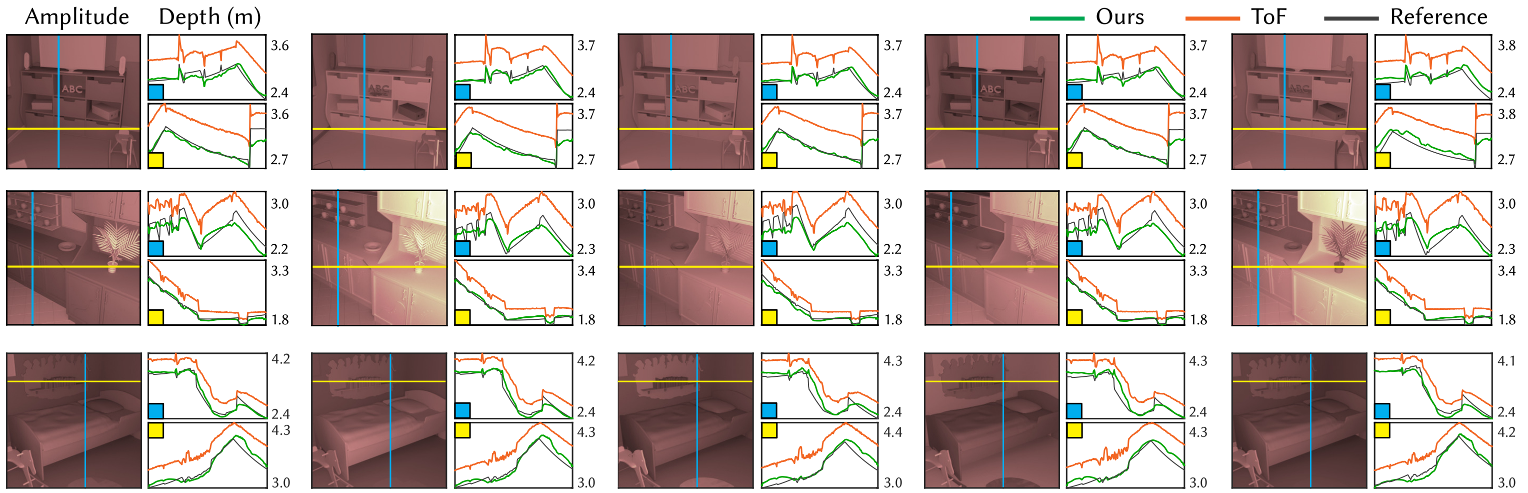


Figure 7.12: Validation results for three scenes (one per row) with randomly varying albedos (one albedo set per column). Images show amplitude, vertical and horizontal plots show depth. Our approach is robust under arbitrary albedo variations.

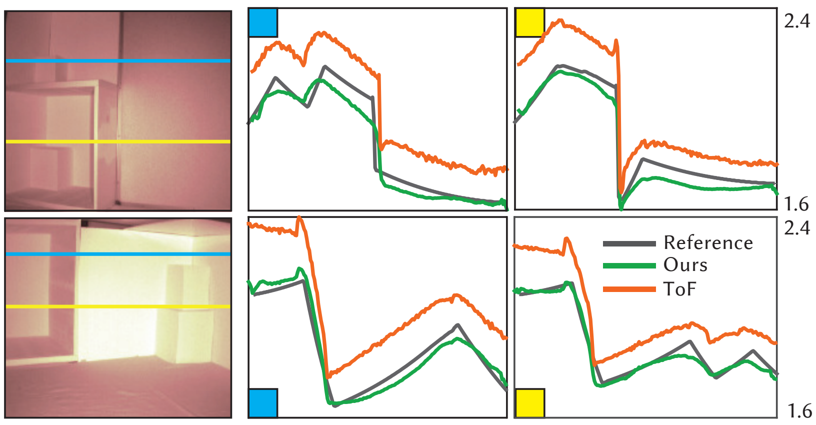


Figure 7.13: Error comparison for different captured combinations of the Cornell box, panels and prisms. Reference depth solutions were obtained replicating the scenes in simulation. We significantly decrease MPI errors in all the captured scenes, yielding errors under 5 cm for the 50% of the pixels, as we demonstrate in the error histograms (Figure 7.10).

frames in Figures 7.7 and 7.15. Full sequences can be found in the supplemental video[3].

7.8 DISCUSSION AND FUTURE WORK

We have presented a new approach for ToF imaging, to compensate the effect of multipath interference in real time, using an unmodified, off-the-shelf camera with a single frequency, and just the incorrect depth map as input. This is possible due to our carefully designed encoder-decoder (convolutional-deconvolutional) neural network, with a two-stage training process both from captured and synthetic data. Additionally, we provide our synthetic time-of-flight dataset that includes pairs of incorrect depth (affected by MPI) and its corresponding correct depth maps, as well as the trained network, for public use.

Several avenues of future work exist. First, as discussed in Section 7.5, we do not consider the wiggling error due to non-perfectly sinusoidal waves in our training dataset, since it is partially compensated by ToF cameras, and manufacturers do not provide information on this. If this information were available, we could incorporate the full camera pipeline (including non-sinusoidal waves and wiggling correction) into our training dataset, and re-train our CNN accounting for these residual errors. In addition, there are still challenging scenarios where results could be improved, as shown in Figure 7.16. The very high

[3] http://webdiis.unizar.es/~juliom/pubs/2017SIGA-DeepToF/2017SIGA_DeepToF_SuppVideo.mp4

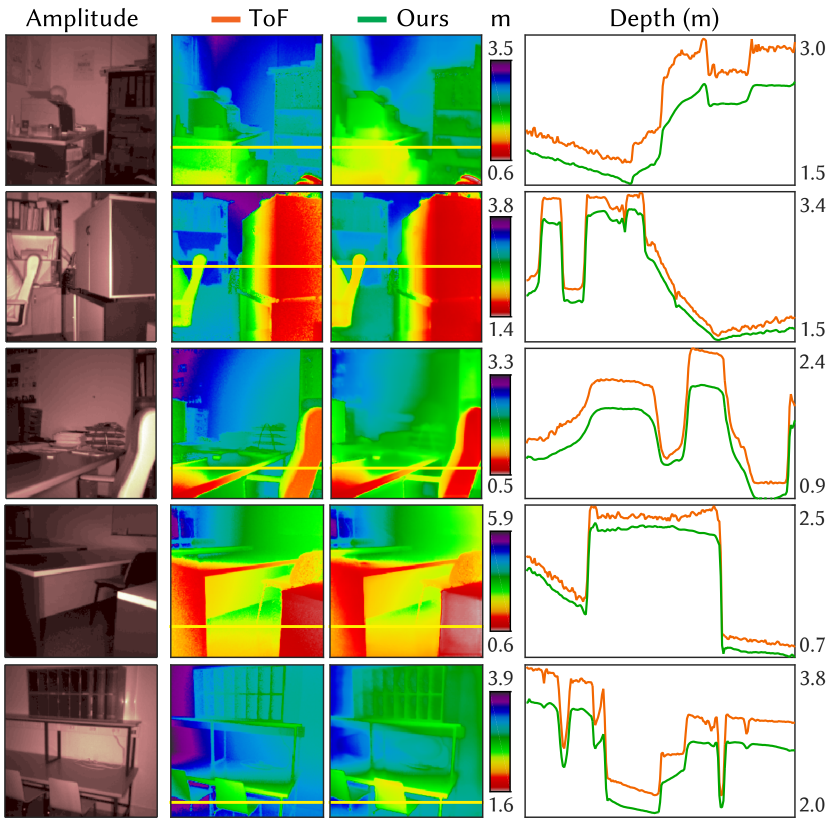


Figure 7.14: Comparison of conventional ToF and our corrected depth, in real world scenes. In accordance with our analysis of MPI errors from our synthetic dataset, longer depths yield higher errors (about 10% of the measured distance).

albedo of barium-painted surfaces creates large MPI errors, specially under specific camera-light configurations (left). Although our MPI correction provides better results than the captured ToF depth, there is still some residual error of about 10 cm in average. Also, our network fails to correct MPI in the presence of objects which are very close to the camera, such as the bottom-left box in the second example (Figure 7.16, right). This is most likely because most of our synthetic dataset contains depths between 1.5m and 4m. Despite this, as we showed in Figure 7.10, per-pixel error distributions are significantly better than captured ToF depth.

Our work assumes diffuse (or nearly diffuse) reflectance. Although we have shown that it works well in several real-world scenarios with more general reflectances, it presents some problems in the presence of highly glossy materials. While incorporating such reflectances into our

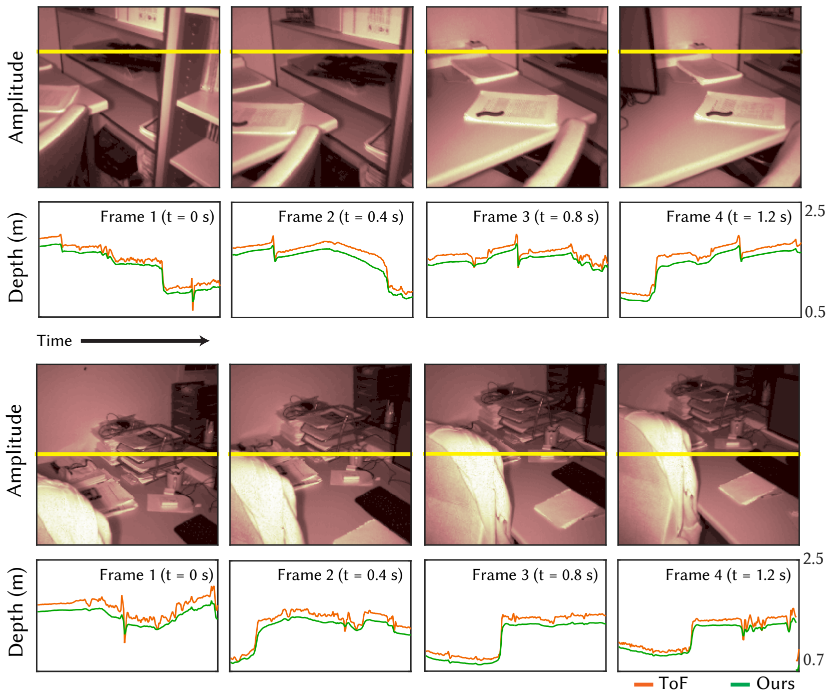


Figure 7.15: Our approach can also be applied to correct MPI errors of in-the-wild videos of real scenes, in real time and keeping temporal consistency. Here we show the depth profiles of a few frames of two of our videos. The complete sequences can be found in the supplemental video[3].

training dataset would help, our approach is likely to fail for extremely glossy or transparent surfaces; in such scenarios, other multi-frequency approaches [130, 219] could be better suited.

APPENDICES

7.A LIGHT TRANSPORT IN IMAGE SPACE

Section 7.3 shows how most of the information on multipath interference from a scene is available in image space. This allows us to approximate Equation (7.4) by limiting the integration domain P to the differential paths $\overline{p}$ that reach pixel p from visible geometry. Moreover, given the discretized domain of an image, we can model Equation (7.4) using the transport matrix $\mathbf{T}_i$ [202, 206], which relates the ideal response

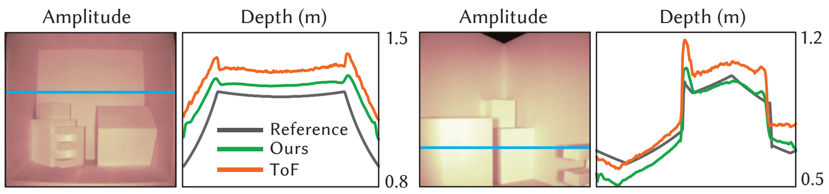


Figure 7.16: Due to the high albedo of our barium-mixed diffuse paint (≈ 0.85), some specific camera-light configurations may yield large MPI errors (left). Objects very close to the camera (right, bottom-left box) yield a higher error since most of our training dataset has depths from 1.5m to 4m (see Figure 7.4). Still, our approach manages to improve MPI errors in both cases, providing results significantly closer to the reference depth.

in pixel p_v with the outgoing response at pixel p_u, for a measurement phase shift i. Thus

$$\begin{aligned}\hat{\mathbf{c}}_i(p_u) &= \mathbf{c}_i(p_u) + \sum_v \mathbf{T}_i(p_u, p_v)\,\mathbf{c}_i(p_v)\\ &= \mathbf{c}_i(p_u) + \mathbf{T}_i * \mathbf{c}_i,\end{aligned} \tag{7.5}$$

where $*$ is the convolution in p_v, and $\mathbf{c}_i$ is the full phase-shifted image. While ideally this means that we can compute the correct phase-shifted image $\mathbf{c}_i$ by applying a deconvolution on the captured $\hat{\mathbf{c}}_i$, as $\mathbf{c}_i = \hat{\mathbf{c}}_i *_v (I + \mathbf{T}_i)^{-1}$ with I the identity matrix, in practice this is not possible since the transport matrix $\mathbf{T}_i$ is unknown. Capturing it is an expensive process, and we cannot make strong simplifying assumptions on the locality of light transport (i.e. sparsity in $\mathbf{T}_i$), since light reflected from far away pixels might have an important contribution on pixel p. However, as we show in Section 7.4, we can learn the resulting deconvolution operator by means of a convolutional neural network.

7.B DEPTH STATISTICS

To validate our synthetic dataset (Section 7.5), we follow previous works on depth image statistics [100, 101] on both real and synthetic datasets. In particular, we analyze single-pixel, derivative and bivariate statistics, as well as joint statistics of Haar wavelet coefficients. To compare the results we use three different metrics: chi-squared error [211], the Jensen-Shannon distance [51, 172], and Pearson's correlation coefficient. The first is a weighted Euclidean error ranging from 0 to ∞ (less is better); the second one measures the similarity between two distributions, ranging from 0 to $\sqrt{ln(2)} = 0.833$ (less is better); the last measures correlation, where a value of 0 indicates two independent variables, and ± 1 indicates a perfect linear direct or inverse relationship.

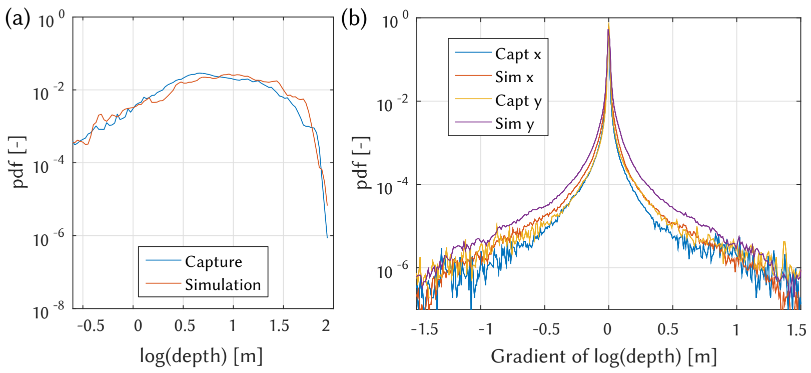


Figure 7.17: (a) Logarithmic depth histogram of real and simulated depths, showing a good match between both sets. (b) Derivative in x and y directions of real and simulated depths, also showing a similar trend. Please refer to the text for quantitative data and other statistical analyses.

Figure 7.17a shows the depth histograms of both sets of images. They are very similar, with a chi-squared error of 0.032, Jensen-Shannon distance of 0.129, and a correlation of 0.90. Figure 7.17b shows how the vertical and horizontal gradients of both datasets also follow a similar trend, with a chi-squared error of 0.051 and 0.028, Jensen-Shannon distance of 0.162 and 0.118, and correlation coefficients of 0.948 and 0.969 for the vertical and horizontal gradients respectively. Both sets have high kurtosis values, as reported by Huang et al. [101].

Next, we carried out a bivariate statistical analysis to pairs of pixels at a fixed separation distance, following the co-ocurrence equation [101, 167]. Capture and simulation match with a chi-squared error of 0.094, Jensen-Shannon distance of 0.245, and correlation of 0.915. Last, we selected three Haar filters (horizontal, vertical, and diagonal) in order to analyze joint statistics in the wavelet domain. Capture and simulation match with a chi-squared error of 0.152 and 0.172, Jensen-Shannon distance of 0.314 and 0.336, and correlation coefficients of 0.892 and 0.896, for the horizontal-vertical and horizontal-diagonal pairs respectively.

These values indicate that our simulated images share very similar depth statistics with existing real-world depth images, so they can be used as reliable input for our network.

Part IV

CONCLUSION

8 CONCLUSIONS

In this thesis we have focused on solving different challenges of computational light transport. We have chosen *light propagation time* as the central axis to distinguish between two sorts of light transport: *steady-state*, where information in the temporal dimension is deemed irrelevant; and *transient-state*, in which time is an essential component in the problems to solve. In the former we have addressed challenges in traditional uses of light transport in rendering applications, while for transient-state we have focused on obtaining and leveraging accurate light transport data for transient imaging problems. In both steady- and transient-state contributions we have faced long-standing and novel problems alike.

STEADY-STATE LIGHT TRANSPORT A common problem addressed in this thesis is variance reduction in Monte Carlo based rendering. In Chapter 2 we addressed this on steady-state rendering by proposing a new radiance caching method for participating media. Inspired by existing approaches that use radiance derivatives to predict radiance changes, we reuse radiance samples in participating media, and avoid tracing rays when radiance can be accurately extrapolated with first-order translational derivatives. Our key contributions are an improved way to compute derivatives that solves inaccuracies of previous works, and extending these derivatives to second order to improve error prediction when driving sampling density. While modern production path tracers have departed from radiance caching algorithms [116, 152, 230, 267], they are still the ground truth for architectural design [128], and are included in rendering engines such as *Radiance* [163, 265] or *V-Ray* [74] for indirect illumination interpolation. Furthermore, our expressions to compute up to second-order local derivatives are not limited just to radiance caching. They are a good estimator of local frequency, which can be used to drive sampling rates or compute optimal kernels for density estimations.

In Chapter 3 we presented contributions to real-time rendering, but under a novel research direction. Far from classic trends in rendering that aim to improve computational efficiency, we addressed energy consumption as a new constraint in the graphics pipeline. Energy efficiency is nowadays a significant requirement in a world flooded by battery-powered devices. In Chapter 3 we demonstrated that both image quality and energy efficiency can be achieved under a proper characterization of the scenes. We showed that by pre-computing energy footprints of a scene we can drive shading parameters during rendering time to satisfy certain energy threshold while maximizing image quality. We detected and addressed challenges under this new rendering framework, such as efficient parameter space exploration, temporal consistency, and adaptive subdivision of the energy maps. Our framework is the first real-time software-based approach to address energy-efficient rendering from a general perspective. Along with the demonstrated benefits, we also brought out a variety of challenges still to be addressed, such as support for dynamic scenarios, or finding energy mappings between high-level rendering settings and low-level instructions. Recent works after our publication have already demonstrated this is a research path with great potential. Follow-up work by Zhang et al. [290] have already addressed the expensiveness of the pre-computation step by estimating energy-consumption on the fly, and Vasiou et al. [253] performed a thorough study that relates energy consumption to the different steps performed during ray tracing. With a proper mapping between energy consumption, shading operations, and perceived quality, saliency-driven modulation of rendering operations could arise as an ambitious way to reduce energy consumption based on user visual attention.

TRANSIENT LIGHT TRANSPORT Though transient light transport simulation is very useful in transient imaging applications, providing efficient Monte Carlo methods to reduce variance in this new domain has barely been investigated. In Chapter 5 we introduced a proper theoretical framework for this purpose. Under this framework, we analyzed variance behavior of Monte Carlo rendering in transient-state, and mitigated it with new strategies for uniform sampling in time, and with reconstruction techniques to reuse the sampled paths. In Chapter 6 we continued this trend by extending the photon beams algorithm [118] to transient state, and deriving the proper convergence rates for progressive spatial and temporal density estimations to guarantee the consistency of the method. These two works are the first contributions that address variance issues on a simulation paradigm of high importance nowadays due to the recent outburst of transient imaging methods. Analogously to what the formulation of the path integral [254, 255] supposed to steady-state rendering, our adaptation to transient state is a mandatory step to bring on further investigations in more robust simulation algorithms for time-resolved light transport. While we provide a solid foundation for this, there is still plenty of work to bring on. Although the reconstruction methods proposed in our framework are general, the sampling strategies are still limited to participating media. Surfaces present a more constrained space, where driving sampled paths along restricted manifolds while guaranteeing uniform sample distributions in time becomes more challenging. Nevertheless, we believe our framework will foster new methods in this regard. Our transient photon beams algorithm serves as a first example that our framework can help to develop more efficient algorithms for transient rendering. By observing the clear benefits of photon-based methods in transient state, we can anticipate that unifying all existing path- and photon-based techniques in transient state would be an appropriate way to robustly handle time-resolved transport in complex material and media combinations. Enabling the simulation of complex light effects in a time-resolved manner is very important, since transient imaging applications end up dealing with arbitrary real situations. Scene configurations with constrained geometry and materials are possible in lab-controlled setups, but the research process progressively aims to increase their complexity to enable practical applications.

Motivated by this, in Chapter 7 we targeted multipath interference reduction (MPI) in *Time-of-Flight* (ToF) depth cameras, increasing the geometric complexity of the captured scenarios. For that purpose, we generated accurate time-resolved light transport in architectural geometry, and used it to mimic the operational principles of ToF depth cameras. With these simulations we could easily isolate and analyze MPI errors under a wide variety of scenarios, avoiding other error sources typical of ToF hardware. We demonstrated that this data can be combined with novel deep-learning techniques to correct MPI depth errors in real-time. Our method is the first approach in transient imaging to perform real-time MPI corrections in a vast amount of complex geometric configurations. We have demonstrated that data-driven methods that use synthetic data are an effective procedure to tackle transient imaging problems, and that our transient rendering framework serves as a solid basis for this. Just a few months after the publication of our method, several follow-up works have proposed to correct MPI with a similar methodology [244], even supporting additional ToF artifacts and providing richer databases [76] using our transient rendering framework. This trend upholds the benefits of using accurate synthetic data for transient imaging problems, but there is still a long way to go. While the geometric complexity and the nature of supported artifacts has increased, most of these works are limited to diffuse-only, media-free scenarios. Although we have extended bidirectional path tracing and photon-based methods to transient state, exploring more sophisticated algorithms is a pending task. Accounting for light propagation time in hybrid methods for surfaces [63, 85] and media [157], and deriving the optimal spatio-temporal kernels would lead to more robust simulations of light in motion. This would allow to efficiently handle transient light transport in a wide variety of scenarios, which could be an importance source of data for imaging applications, not only targeting MPI reduction, but also applications such as reflectance acquisition or material classification [195, 196, 208, 243, 249, 278]. Tackling MPI errors in ToF depth capture (i.e. visible geometry) is just the tip of the iceberg on a wide range of transient imaging applications for geometry reconstruction. Methods for reconstructing non-line-of-sight (NLOS) geometry [79, 257] present much more challenging cases of use, where the relevant information is encoded in higher-order light bounces at later timings. In those cases radiance signal is weaker, noisier, and more convoluted due to multiple bounces in the geometry. Using forward models to assist the reconstruction process is an effective approach, but it requires further investigation in algorithms that explore the spatio-temporal manifolds both robustly and efficiently. Arellano et al. [8] have already boosted performance of 3rd-bounce hidden geometry reconstruction by a factor of 1000, thanks to a convenient implementation of back-projection in GPU. But still, pushing reconstruction to higher bounces is an established goal in transient imaging, which will require more efficient forward models that can handle the increased complexity in a practical way.

PERSONAL CONCLUSIONS When I became interested in computer graphics in my teens, I could only have dreamed to be involved in all the amazing projects and collaborations that put together this thesis. I feel very lucky to have had the chance to work with brilliant people, be part of a collaborative and internationalized research group, and work with companies that seemed unreachable not so long ago. The last four years have supposed an unceasing stream of information that has expanded my knowledge in many different areas, from maths and physics, to their practical application in graphics and imaging problems. Complementary to this, I think that I have significantly improved my abstraction abilities when facing new problems, which I strongly believe is a key component as valuable as knowledge. I feel this thesis has helped me to develop myself as a fruitful researcher, become a better engineer, and has allowed me to acquire many other skills that go beyond technical aspects.

From a more personal point of view, the elaboration of this thesis has been a path of many flavors. Perseverance and blind faith have been essential to climb all the steps along a way where panic, stress, and desperation have been always bystanders. Despite all the sleepless nights, it has also been a fun path full of joy and satisfaction when hard work paid off. But the most important ingredients that made this thesis possible are all the people that have been around me during these years: in and out of the workplace, either helping me to push through or dragging me away from the desk, they have helped me to not hang it up.

BIBLIOGRAPHY

[1] Amit Adam, Christoph Dann, Omer Yair, Shai Mazor, and Sebastian Nowozin. "Bayesian time-of-flight for realtime shape, illumination and albedo." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.5 (2017), pp. 851–864.

[2] Martin David Adams and Penny J. Probert. "The interpretation of phase and intensity data from AMCW light detection sensors for reliable ranging." In: *The International Journal of Robotics Research* 15.5 (1996).

[3] T. Akenine-Möller and J. Strom. "Graphics Processing Units for Handhelds." In: *Proceedings of the IEEE* 96.5 (2008), pp. 779–789.

[4] Andrew Allen and Nikunj Raghuvanshi. "Aerophones in Flatland: Interactive Wave Simulation of Wind Instruments." In: *ACM Trans. Graph.* 34.4 (July 2015), 134:1–134:11.

[5] Marco Ament, Christoph Bergmann, and Daniel Weiskopf. "Refractive radiative transfer equation." In: *ACM Trans. Graph.* 33.2 (2014).

[6] Lakulish Antani, Anish Chandak, Micah Taylor, and Dinesh Manocha. "Direct-to-Indirect Acoustic Radiance Transfer." In: *IEEE Transactions on Visualization and Computer Graphics* 18.2 (2012).

[7] Arthur Appel. "Some techniques for shading machine renderings of solids." In: *Proceedings of the April 30–May 2, 1968, spring joint computer conference*. ACM. 1968, pp. 37–45.

[8] Victor Arellano, Diego Gutierrez, and Adrian Jarabo. "Fast Back-Projection for Non-Line of Sight Reconstruction." In: *Optics Express* 25.10 (2017).

[9] Jose-Maria Arnau, Joan-Manuel Parcerisa, and Polychronis Xekalakis. "Eliminating Redundant Fragment Shader Executions on a Mobile GPU via Hardware Memoization." In: *SIGARCH Comput. Archit. News* 42.3 (June 2014), pp. 529–540.

[10] James Arvo. "The Irradiance Jacobian for Partially Occluded Polyhedral Sources." In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 343–350.

[11] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. "Kernel-predicting convolutional networks for denoising Monte Carlo renderings." In: *ACM Trans. Graph* 36.4 (2017), p. 97.

[12] Laurent Belcour. "Efficient rendering of layered materials using an atomic decomposition with statistical operators." In: *ACM Transactions on Graphics* 37.4 (2017), p. 1.

[13] Laurent Belcour, Kavita Bala, and Cyril Soler. "A Local Frequency Analysis of Light Scattering and Absorption." In: *ACM Trans. Graph.* 33.5 (Sept. 2014), 163:1–163:17.

[14] George I Bell and Samuel Glasstone. *Nuclear reactor theory*. Tech. rep. US Atomic Energy Commission, Washington, DC (United States), 1970.

[15] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing." In: *Future Gener. Comput. Syst.* 28.5 (May 2012), pp. 755–768.

[16] M. Bertram, E. Deines, J. Mohring, J. Jegorovs, and H. Hagen. "Phonon tracing for auralization and visualization of sound." In: *IEEE Visualization '05*. 2005.

[17] Ayush Bhandari and Ramesh Raskar. "Signal Processing for Time-of-Flight Imaging Sensors." In: *IEEE Signal Processing Magazine* 33.5 (2016).

[18] Ayush Bhandari, Micha Feigin, Shahram Izadi, Christoph Rhemann, Mirko Schmidt, and Ramesh Raskar. "Resolving multipath interference in Kinect: An inverse problem approach." In: *IEEE SENSORS*. 2014.

[19] Benedikt Bitterli. *Rendering resources*. https://benedikt-bitterli.me/resources/. 2016.

[20] Benedikt Bitterli. *Virtual Femto Photography*. https://benedikt-bitterli.me/femto.html. 2016.

[21] Benedikt Bitterli and Wojciech Jarosz. "Beyond Points and Beams: Higher-Dimensional Photon Samples for Volumetric Light Transport." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36.4 (2017).

[22] Max Born and Emil Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 2002.

[23] Jean-Yves Bouguet. *Camera calibration toolbox for Matlab*. 2004.

[24] Jens Busck. "Underwater 3-D optical imaging with a gated viewing laser radar." In: *Optical Engineering* 44.11 (2005).

[25] Jens Busck and Henning Heiselberg. "Gated viewing and high-accuracy three-dimensional laser radar." In: *Applied Optics* 43.24 (2004).

[26] Mike Cammarano and Henrik Wann Jensen. "Time Dependent Photon Mapping." In: *Eurographics Workshop on Rendering*. 2002.

[27] Kenneth M Case, George Placzek, and Frederic Hoffmann. "Introduction to the theory of neutron diffusion, v. 1." In: (1953).

[28] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Dover, 1960.

[29] Haidong Chen, Ji Wang, Weifeng Chen, Huamin Qu, and Wei Chen. "An image-space energy-saving visualization scheme for OLED displays." In: *Computers & Graphics* 38 (2014), pp. 61 –68.

[30] Weifeng Chen, Wei Chen, Haidong Chen, Zhengfang Zhang, and Huamin Qu. "An energy-saving color scheme for direct volume rendering." In: *Computers & Graphics* 54 (2016), pp. 57 –64.

[31] W.-C. Cheng and M. Pedram. "Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling." In: *IEEE Transactions on Consumer Electronics* 50.1 (2004), pp. 25–32.

[32] Inchang Choi, Daniel S. Jeon, Giljoo Nam, Diego Gutierrez, and Min H. Kim. "High-Quality Hyperspectral Reconstruction Using a Spectral Prior." In: *ACM Transactions on Graphics (SIGGRAPH Asia 2017)* 36.6 (2017).

[33] Johnson Chuang, Daniel Weiskopf, and Torsten Möller. "Energy Aware Color Sets." In: *Computer Graphics Forum* 28.2 (2009), pp. 203–211.

[34] A. Cohade and S. de los Santos. "Power Efficient Programming: How Funcom increased play time in Lego Minifigures." In: *Game Developer's Conference*. 2015.

[35] Mark Colbert and Jaroslav Křivánek. "GPU-based importance sampling." In: *GPU Gems 3* (2007), pp. 459–476.

[36] Richard M Conroy, Adrian A Dorrington, Rainer Künnemeyer, and Michael J Cree. "Range imager performance comparison in homodyne and heterodyne operating modes." In: *IS&T/SPIE Electronic Imaging*. 2009.

[37] Robert L. Cook, Thomas Porter, and Loren Carpenter. "Distributed ray tracing." In: *SIGGRAPH Comput. Graph.* 18.3 (1984).

[38] Eugene D'Eon and Geoffrey Irving. "A quantized-diffusion model for rendering translucent materials." In: *ACM Trans. Graph.* 30.4 (2011).

[39] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." In: *Trans. Evol. Comp* 6.2 (2002), pp. 182–197.

[40] Mian Dong and Lin Zhong. "Power Modeling and Optimization for OLED Displays." In: *IEEE Transactions on Mobile Computing* 11.9 (2012), pp. 1587–1599.

[41] Mian Dong, Yung-Seok Kevin Choi, and Lin Zhong. "Power-saving Color Transformation of Mobile Graphical User Interfaces on OLED-based Displays." In: *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*. ISLPED '09. San Fancisco, CA, USA, 2009, pp. 339–342.

[42] Adrian A Dorrington, Michael J Cree, Andrew D Payne, Richard M Conroy, and Dale A Carnegie. "Achieving sub-millimetre precision with a solid-state full-field heterodyning range imaging camera." In: *Measurement Science and Technology* 18.9 (2007).

[43] Adrian A Dorrington, John Peter Godbaz, Michael J Cree, Andrew D Payne, and Lee V Streeter. "Separating true range measurements from multi-path and scattering interference in commercial range cameras." In: *IS&T/SPIE Electronic Imaging*. 2011.

[44] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao. "Stacked Convolutional Denoising Auto-Encoders for Feature Representation." In: *IEEE Trans. Cybernetics* 99 (2016), pp. 1–11.

[45] James J Duderstadt and William Russell Martin. "Transport theory." In: *Transport theory., by Duderstadt, JJ; Martin, WR. Chichester (UK): John Wiley & Sons, 10+ 613 p.* (1979).

[46] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. "A frequency analysis of light transport." In: *ACM Trans. Graph.* 24.3 (2005).

[47] Philip Dutré, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. AK Peters, 2006.

[48] David Eigen and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture." In: *Proceedings of ICCV*. 2015.

[49] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network." In: *Proceedings of NIPS*. 2014.

[50] Oskar Elek, Pablo Bauszat, Tobias Ritschel, Marcus Magnor, and Hans-Peter Seidel. "Spectral Ray Differentials." In: *Computer Graphics Forum (Proceedings of EGSR)* 33.4 (2014).

[51] Dominik Maria Endres and Johannes E Schindelin. "A new metric for probability distributions." In: *IEEE Transactions on Information theory* 49.7 (2003), pp. 1858–1860.

[52] D Falie. "Improvements of the 3D images captured with time-of-flight cameras." In: *arXiv preprint arXiv:0909.5656* (2009).

[53] Luca Fascione, Johannes Hanika, Rob Pieké, Ryusuke Villemin, Christophe Hery, Manuel Gamito, Luke Emrose, and André Mazzone. "Path tracing in production." In: *ACM SIGGRAPH 2018 Courses*. ACM. 2018, p. 15.

[54] Micha Feigin, Ayush Bhandari, Shahram Izadi, Christoph Rhemann, Mirko Schmidt, and Ramesh Raskar. "Resolving multipath interference in kinect: An inverse problem approach." In: *IEEE Sensors Journal* 16.10 (2015).

[55] Stephen R. Forrest. "The road to high efficiency organic light emitting devices." In: *Organic Electronics* 4.2 - 3 (2003), pp. 45 –48.

[56] Daniel Freedman, Yoni Smolin, Eyal Krupka, Ido Leichter, and Mirko Schmidt. "SRA: Fast removal of general multipath for ToF sensors." In: *European Conference on Computer Vision*. 2014.

[57] Stefan Fuchs. "Multipath interference compensation in time-of-flight camera images." In: *IEEE International Conference on Pattern Recognition*. 2010.

[58] Stefan Fuchs and Gerd Hirzinger. "Extrinsic and depth calibration of ToF-cameras." In: *IEEE Computer Vision and Pattern Recognition*. 2008.

[59] Stefan Fuchs, Michael Suppa, and Olaf Hellwich. "Compensation for Multipath in ToF Camera Measurements Supported by Photometric Calibration and Environment Integration." In: *Proceedings of the International Conference on Computer Vision Systems*. ICVS'13. St. Petersburg, Russia: Springer-Verlag, 2013, pp. 31–41.

[60] Thomas Funkhouser, Nicolas Tsingos, and Jean-Marc Jot. "Survey of Methods for Modeling Sound Propagation in Interactive Virtual Environment Systems." In: *Presence and Teleoperation* (2003).

[61] Epic Games. *Unreal Engine*. https://www.unrealengine.com/. 2015.

[62] Liang Gao, Jinyang Liang, Chiye Li, and Lihong V Wang. "Single-shot compressed ultrafast photography at one hundred billion frames per second." In: *Nature* 516.7529 (2014).

[63] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. "Light Transport Simulation with Vertex Connection and Merging." In: *ACM Trans. Graph.* 31.6 (2012).

[64] Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. "Joint Importance Sampling of Low-Order Volumetric Scattering." In: *ACM Trans. Graph.* 32.6 (2013).

[65] Michaël Gharbi, YiChang Shih, Gaurav Chaurasia, Jonathan Ragan-Kelley, Sylvain Paris, and Frédo Durand. "Transform Recipes for Efficient Cloud Photo Enhancement." In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 228:1–228:12.

[66] Andrew S Glassner. *Principles of digital image synthesis*. Elsevier, 2014.

[67] John P Godbaz, Michael J Cree, and Adrian A Dorrington. "Mixed pixel return separation for a full-field ranger." In: *IEEE International Conference Image and Vision Computing New Zealand '08*. 2008.

[68] John P Godbaz, Michael J Cree, and Adrian A Dorrington. "Multiple return separation for a full-field ranger via continuous waveform modelling." In: *IS&T/SPIE Electronic Imaging*. 2009.

[69] John P Godbaz, Michael J Cree, and Adrian A Dorrington. "Extending amcw lidar depth-of-field using a coded aperture." In: *Asian Conference on Computer Vision 2010*. 2010.

[70] John P Godbaz, Michael J Cree, and Adrian A Dorrington. "Closed-form inverses for the mixed pixel/multipath interference problem in amcw lidar." In: *IS&T/SPIE Electronic Imaging*. 2012.

[71] Jay S. Gondek, Gary W. Meyer, and Jonathan G. Newman. "Wavelength dependent reflectance functions." In: *SIGGRAPH*. 1994.

[72] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[73] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. "Modeling the interaction of light between diffuse surfaces." In: *SIGGRAPH Comput. Graph.* 18.3 (1984).

[74] Chaos Group. *V-Ray*. https://www.chaosgroup.com/. 2018.

[75] Adrien Gruson, Mickaël Ribardière, Martin Šik, Jiří Vorba, Rémi Cozot, Kadi Bouatouch, and Jaroslav Křivánek. "A Spatial Target Function for Metropolis Photon Tracing." In: *ACM Trans. Graph.* 36.1 (2016).

[76] Qi Guo, Iuri Frosio, Orazio Gallo, Todd Zickler, and Jan Kautz. "Tackling 3D ToF Artifacts Through Learning and the FLAT Dataset." In: *arXiv preprint arXiv:1807.10376* (2018).

[77] Mohit Gupta, Achuta Kadambi, Ayush Bhandari, and Ramesh Raskar. *Computational Time of Flight*. In ICCV Courses. 2015.

[78] Mohit Gupta, Shree K Nayar, Matthias B Hullin, and Jaime Martin. "Phasor imaging: A generalization of correlation-based time-of-flight imaging." In: *ACM Trans. Graph.* 34.5 (2015).

[79] Otkrist Gupta, Thomas Willwacher, Andreas Velten, Ashok Veeraraghavan, and Ramesh Raskar. "Reconstruction of hidden 3D shapes using diffuse reflections." In: *Opt. Express* 20.17 (2012).

[80] Diego Gutierrez, Adolfo Muñoz, Oscar Anson, and Francisco Seron. "Non-linear volume photon mapping." In: *Eurographics Symposium on Rendering*. 2005.

[81] Diego Gutierrez, Srinivasa G. Narasimhan, Henrik Wann Jensen, and Wojciech Jarosz. "Scattering." In: *ACM SIGGRAPH ASIA 2008 Courses*. 2008.

[82] Toshiya Hachisuka and Henrik Wann Jensen. "Stochastic Progressive Photon Mapping." In: *ACM Trans. Graph.* 28.5 (2009).

[83] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. "Progressive photon mapping." In: *ACM Trans. Graph. (TOG)* 27.5 (2008), p. 130.

[84] Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. "A Progressive Error Estimation Framework for Photon Density Estimation." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 29.6 (2010), 144:1–144:12.

[85] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. "A path space extension for robust light transport simulation." In: *ACM Trans. Graph. (TOG)* 31.6 (2012), p. 191.

[86] Toshiya Hachisuka, Wojciech Jarosz, Iliyan Georgiev, Anton Kaplanyan, and Derek Nowrouzezahrai. "State of the Art in Photon Density Estimation." In: *ACM SIGGRAPH ASIA 2013 Courses*. 2013.

[87] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. "Virtual Spherical Lights for Many-light Rendering of Glossy Scenes." In: *ACM Trans. Graph.* 28.5 (2009).

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity Mappings in Deep Residual Networks." In: *Proceedings of ECCV*. 2016.

[89] Yong He, Tim Foley, Natalya Tatarchuk, and Kayvon Fatahalian. "A System for Rapid, Automatic Shader Level-of-detail." In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 187:1–187:12.

[90] Martial Hebert and Eric Krotkov. "3D measurements from imaging laser radars: how good are they?" In: *Image Vision Comput.* 10.3 (1992).

[91] Paul S. Heckbert. "Radiosity in Flatland." In: *Computer Graphics Forum* 2 (1992), pp. 181–192.

[92] Felix Heide, Matthias Hullin, James Gregson, and Wolfgang Heidrich. "Low-budget Transient Imaging Using Photonic Mixer Devices." In: *ACM Trans. Graph.* 32.4 (2013).

[93] Felix Heide, Lei Xiao, Wolfgang Heidrich, and Matthias B Hullin. "Diffuse mirrors: 3D reconstruction from diffuse indirect illumination using inexpensive time-of-flight sensors." In: *IEEE Computer Vision and Pattern Recognition*. 2014.

[94] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pająk, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. "FlexISP: A flexible camera image processing framework." In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), p. 231.

[95] Robert Herzog, Karol Myszkowski, and Hans-Peter Seidel. "Anisotropic Radiance-Cache Splatting for Efficiently Computing High-Quality Global Illumination with Lightcuts." In: 28.2 (2009), pp. 259–268.

[96] Barmak Heshmat, Guy Satat, Christopher Barsi, and Ramesh Raskar. "Single-shot ultrafast imaging using parallax-free alignment with a tilted lenslet array." In: *CLEO: Science and Innovations*. 2014.

[97] Nicolas Holzschuch and François X. Sillion. "Accurate Computation of the Radiosity Gradient with Constant and Linear Emitters." In: *Rendering Techniques 1995 (Eurographics Symposium on Rendering)*. Dublin, Ireland, June 1995, pp. 186–195.

[98] Nicolas Holzschuch and Francois Sillion. "An Exhaustive Error-Bounding Algorithm for Hierarchical Radiosity." In: *Computer Graphics Forum*. Vol. 17. 4. Wiley Online Library. 1998, pp. 197–218.

[99] Jensen Huang. *NVIDIA Global Citizenship: Letter from our CEO*. 2017.

[100] Jinggang Huang and David Mumford. "Statistics of natural images and models." In: *Proceedings of CVPR*. Vol. 1. IEEE. 1999, pp. 541–547.

[101] Jinggang Huang, Ann B Lee, and David Mumford. "Statistics of range images." In: *Proceedings of CVPR*. Vol. 1. IEEE. 2000, pp. 324–331.

[102] Matthias B Hullin. "Computational imaging of light in flight." In: *SPIE/COS Photonics Asia*. 2014.

[103] Homan Igehy. "Tracing ray differentials." In: *Proceedings of SIGGRAPH* 33 (1999), pp. 179–186.

[104] Ivo Ihrke, Gernot Ziegler, Art Tevs, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. "Eikonal Rendering: Efficient Light Transport in Refractive Objects." In: *ACM Trans. Graph.* 26.3 (2007).

[105] Subu Iyer, Lu Luo, Robert Mayo, and Parthasarathy Ranganathan. "Energy-Adaptive Display System Designs for Future Mobile Environments." In: *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*. MobiSys '03. 2003, pp. 245–258.

[106] Wenzel Jakob, Christian Regg, and Wojciech Jarosz. "Progressive Expectation-Maximization for Hierarchical Volumetric Photon Mapping." In: *Computer Graphics Forum*. Vol. 30. 4. Wiley Online Library. 2011, pp. 1287–1297.

[107] Sonam Jamtsho and Derek D Lichti. "Modelling scattering distortion in 3D range camera." In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.5 (2010).

[108] Adrian Jarabo. "Femto-photography: Visualizing light in motion." MA thesis. Universidad de Zaragoza, 2012.

[109] Adrian Jarabo and Victor Arellano. "Bidirectional Rendering of Vector Light Transport." In: *Computer Graphics Forum* To appear (2018).

[110] Adrian Jarabo, Belen Masia, Andreas Velten, Christopher Barsi, Ramesh Raskar, and Diego Gutierrez. "Rendering Relativistic Effects in Transient Imaging." In: *Congreso Español de Informática Gráfica*. 2013.

[111] Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. "A Framework for Transient Rendering." In: *ACM Transactions on Graphics (SIGGRAPH Asia 2014)* 33.6 (2014).

[112] Adrian Jarabo, Belen Masia, Andreas Velten, Christopher Barsi, Ramesh Raskar, and Diego Gutierrez. "Relativistic Effects for Time-Resolved Light Transport." In: *Computer Graphics Forum* 34.8 (2015).

[113] Adrian Jarabo, Belen Masia, Julio Marco, and Diego Gutierrez. "Recent Advances in Transient Imaging: A Computer Graphics and Vision Perspective." In: *Visual Informatics* 1.1 (2017).

[114] Adrian Jarabo, Carlos Aliaga, and Diego Gutierrez. "A Radiative Transfer Framework for Spatially-Correlated Materials." In: *ACM Transactions on Graphics* 37.4 (2018).

[115] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. "Irradiance Gradients in the Presence of Participating Media and Occlusions." In: *Computer Graphics Forum (Proceedings of EGSR)* 27.4 (June 2008), pp. 1087–1096.

[116] Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. "Radiance Caching for Participating Media." In: *ACM Transactions on Graphics (Presented at SIGGRAPH)* 27.1 (Mar. 2008), 7:1–7:11.

[117] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. "The Beam Radiance Estimate for Volumetric Photon Mapping." In: *Computer Graphics Forum (Proceedings of Eurographics)* 27.2 (Apr. 2008), pp. 557–566.

[118] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. "A Comprehensive Theory of Volumetric Radiance Estimation Using Photon Points and Beams." In: *ACM Transactions on Graphics (Presented at SIGGRAPH)* 30.1 (Jan. 2011), 5:1–5:19.

[119] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. "Progressive Photon Beams." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 30.6 (2011).

[120] Wojciech Jarosz, Volker Schönefeld, Leif Kobbelt, and Henrik Wann Jensen. "Theory, Analysis and Applications of 2D Global Illumination." In: *ACM Transactions on Graphics (Presented at SIGGRAPH)* 31.5 (Sept. 2012), 125:1–125:21.

[121] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[122] Henrik Wann Jensen and Per H Christensen. "Efficient simulation of light transport in scenes with participating media using photon maps." In: *SIGGRAPH '98*. ACM. 1998, pp. 311–320.

[123] David Jiménez, Daniel Pizarro, Manuel Mazo, and Sira Palazuelos. "Modeling and correction of multipath interference in time of flight cameras." In: *IEEE Computer Vision and Pattern Recognition*. 2012.

[124] David Jiménez, Daniel Pizarro, Manuel Mazo, and Sira Palazuelos. "Modeling and correction of multipath interference in Time-of-Flight cameras." In: *Image and Vision Computing* 32.1 (2014), pp. 1–13.

[125] Jorge Jimenez, Belen Masia, Jose I. Echevarria, Fernando Navarro, and Diego Gutierrez. "GPU Pro 2." In: ed. by Wolfgang Engel. AK Peters Ltd., 2011. Chap. Practical Morphological Anti-Aliasing.

[126] Björn M Johnsson. "Energy Analysis for Graphics Processors using Novel Methods & Efficient Multi-View Rendering." eng. PhD thesis. Lund University, 2014, p. 148.

[127] Björn Johnsson, Per Ganestam, Michael Doggett, and Tomas Akenine-Möller. "Power Efficiency for Software Algorithms Running on Graphics Processors." In: *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. EGGH-HPG'12. Paris, France: Eurographics Association, 2012, pp. 67–75.

[128] Nathaniel Louis Jones. "Validated interactive daylighting analysis for architectural design." PhD thesis. Massachusetts Institute of Technology, 2017.

[129] Adrian PP Jongenelen, Dale A Carnegie, Andrew D Payne, and Adrian A Dorrington. "Maximizing precision over extended unambiguous range for TOF range imaging systems." In: *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*. 2010.

[130] A. Kadambi, R. Whyte, A. Bhandari, L. Streeter, C. Barsi, A. Dorrington, and R. Raskar. "Coded time of flight cameras: sparse deconvolution to address multipath interference and recover time profiles." In: *ACM Trans. Graph.* 32.6 (2013).

[131] Achuta Kadambi, Jamie Schiel, and Ramesh Raskar. "Macroscopic Interferometry: Rethinking Depth Estimation With Frequency-Domain Time-Of-Flight." In: *IEEE Computer Vision and Pattern Recognition*. 2016.

[132] Vladimir Kajalin. "Screen space ambient occlusion." In: *Shader X* 7.413 (2009), p. 24.

[133] James T. Kajiya. "The Rendering Equation." In: *SIGGRAPH*. 1986.

[134] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. "Learning-based View Synthesis for Light Field Cameras." In: *ACM Trans. Graph.* 35.6 (Nov. 2016), 193:1–193:10.

[135] H.J. Kalli and E.D. Cashwell. *Evaluation of three Monte Carlo estimation schemes for flux at a point*. Tech. rep. LA-6865-MS. New Mexico, USA: Los Alamos Scientific Lab, 1977.

[136] Simon Kallweit, Thomas Müller, Brian McWilliams, Markus Gross, and Jan Novák. "Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks." In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 231.

[137] Anton S Kaplanyan and Carsten Dachsbacher. "Adaptive progressive photon mapping." In: *ACM Trans. Graph. (TOG)* 32.2 (2013), p. 16.

[138] S Karayev, Y Jia, J Barron, M Fritz, K Saenko, and T Darrell. "A category-level 3-D object dataset: putting the Kinect to work." In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2011, pp. 1167–1174.

[139] Tom Kavli, Trine Kirkhus, Jens T Thielemann, and Borys Jagielski. "Modelling and compensating measurement errors caused by scattering in time-of-flight cameras." In: *Optical Engineering+ Applications*. 2008.

[140] Alexander Keller. "Instant radiosity." In: *SIGGRAPH '97*. 1997.

[141] Alexander Keller, Jaroslav Křivánek, Jan Novák, Anton Kaplanyan, and Marco Salvi. "Machine learning and rendering." In: *ACM SIGGRAPH 2018 Courses*. ACM. 2018, p. 19.

[142] Maik Keller and Andreas Kolb. "Real-time simulation of time-of-flight sensors." In: *Simulation Modelling Practice and Theory* 17.5 (2009).

[143] Maik Keller, Jens Orthmann, Andreas Kolb, and Valerij Peters. "A Simulation Framework for Time-Of-Flight Sensors." In: *International Symposium on Signals, Circuits and Systems 2007*. 2007.

[144] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. "Gradient-Domain Path Tracing." In: *ACM Trans. Graph.* 34.4 (2015).

[145] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[146] Ahmed Kirmani, Tyler Hutchison, James Davis, and Ramesh Raskar. "Looking Around the Corner using Ultrafast Transient Imaging." In: *International Journal of Computer Vision* 95.1 (2011).

[147] Ahmed Kirmani, Arrigo Benedetti, and Philip A Chou. "Spumic: Simultaneous phase unwrapping and multipath interference cancellation in time-of-flight cameras using spectral methods." In: *IEEE International Conference on Multimedia and Expo*. 2013.

[148] Jonathan Klein, Christoph Peters, Jaime Martín, Martin Laurenzis, and Matthias B Hullin. "Tracking objects outside the line of sight using 2D intensity images." In: *Scientific Reports* 6 (2016).

[149] Claude Knaus and Matthias Zwicker. "Progressive photon mapping: A probabilistic approach." In: *ACM Trans. Graph. (TOG)* 30.3 (2011), p. 25.

[150] Nick Knighton and Bruce Bugbee. *A mixture of barium sulfate and white paint is a low-cost substitute reflectance standard for Spectralon®*. 2005.

[151] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. "Time-of-Flight Sensors in Computer Graphics." In: *Computer Graphics Forum* 29.1 (2010).

[152] Jaroslav Křivánek and Pascal Gautron. *Practical Global Illumination with Irradiance Caching*. Synthesis lectures in computer graphics and animation. Morgan & Claypool, 2009.

[153] Jaroslav Křivánek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. "Improved radiance gradient computation." In: *Proceedings of the 21st spring conference on Computer graphics*. SCCG '05. Budmerice, Slovakia: ACM, 2005, pp. 155–159.

[154] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. "Radiance caching for efficient global illumination computation." In: *Visualization and Computer Graphics, IEEE Transactions on* 11.5 (2005), pp. 550 –561.

[155] Jaroslav Krivánek, Kadi Bouatouch, Sumanta N Pattanaik, and Jiri Zara. "Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping." In: *Rendering Techniques* 2006 (2006), pp. 127–138.

[156] Jaroslav Křivánek, Iliyan Georgiev, Anton Kaplanyan, and Juan Canada. "Recent Advances in Light Transport Simulation: Theory and Practice." In: *ACM SIGGRAPH 2013 Courses*. 2013.

[157] Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. "Unifying points, beams, and paths in volumetric light transport simulation." In: *ACM Trans. Graph.* 33.4 (2014).

[158] Christopher Kulla and Marcos Fajardo. "Importance sampling techniques for path tracing in participating media." In: *Computer Graphics Forum* 31.4 (2012).

[159] Chong-Min Kyung and Sungjoo Yoo. *Energy-Aware System Design: Algorithms and Architectures*. Springer Publishing Company, Incorporated, 2014.

[160] Martin Lambers, Stefan Hoberg, and Andreas Kolb. "Simulation of Time-of-Flight Sensors for Evaluation of Chip Layout Variants." In: *IEEE Sensors Journal* 15.7 (2015).

[161] Robert Lange and Peter Seitz. "Solid-state time-of-flight range camera." In: *IEEE Journal of quantum electronics* 37.3 (2001).

[162] Robert Lange, Peter Seitz, Alice Biber, and Stefan C Lauxtermann. "Demodulation pixels in CCD and CMOS technologies for time-of-flight ranging." In: *Electronic Imaging*. 2000.

[163] Greg Ward Larson and Rob Shakespeare. *Rendering with Radiance: the art and science of lighting visualization*. Booksurge Llc, 2004.

[164] Pedro Latorre, Francisco Seron, and Diego Gutierrez. "Birefringency: Calculation of Refracted Ray Paths in Biaxial Crystals." In: *The Visual Computer* 28.4 (2012).

[165] Martin Laurenzis and Andreas Velten. "Nonline-of-sight laser gated viewing of scattered photons." In: *Opt. Eng.* 53.2 (2014).

[166] Martin Laurenzis, Frank Christnacher, and David Monnin. "Long-range three-dimensional active imaging with superresolution depth mapping." In: *Opt. Lett.* 32.21 (2007).

[167] Ann B Lee, JG Huang, and DB Mumford. "Random collage model for natural images." In: *Int. J. of Computer Vision* (2000).

[168] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. "Gradient-Domain Metropolis Light Transport." In: *ACM Trans. Graph.* 32.4 (2013).

[169] Bo Li, Chunhua Shen, Yuchao Dai, A. van den Hengel, and Mingyi He. "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs." In: *Proceedings of CVPR*. 2015.

[170] Li Li, Lei Wu, Xingbin Wang, and Ersheng Dang. "Gated viewing laser imaging with compressive sensing." In: *Applied Optics* 51.14 (2012).

[171] Ivan T Lima, Anshul Kalra, and Sherif S Sherif. "Improved importance sampling for Monte Carlo simulation of time-domain optical coherence tomography." In: *Biomedical optics express* 2.5 (2011).

[172] Jianhua Lin. "Divergence measures based on the Shannon entropy." In: *IEEE Transactions on Information theory* 37.1 (1991), pp. 145–151.

[173] Jingyu Lin, Yebin Liu, Matthias B. Hullin, and Qionghai Dai. "Fourier Analysis on Transient Imaging with a Multifrequency Time-of-Flight Camera." In: *IEEE Computer Vision and Pattern Recognition*. 2014.

[174] Marvin Lindner, Ingo Schiller, Andreas Kolb, and Reinhard Koch. "Time-of-Flight Sensor Calibration for Accurate Range Sensing." In: *Comput. Vis. Image Underst.* 114.12 (2010).

[175] Fayao Liu, Chunhua Shen, and Guosheng Lin. "Deep Convolutional Neural Fields for Depth Estimation from a Single Image." In: *Proceedings of CVPR*. 2015.

[176] Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. "Improved Sampling for Gradient-domain Metropolis Light Transport." In: *ACM Trans. Graph.* 33.6 (2014).

[177] Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Fredo Durand, and Matthias Zwicker. "Gradient-Domain Bidirectional Path Tracing." In: *Proc. of EGSR*. 2015.

[178] Julio Marco. “Transient Light Transport in Participating Media.” MA thesis. Universidad de Zaragoza, 2013.

[179] Julio Marco, Quercus Hernandez, Adolfo Muñoz, Yue Dong, Adrian Jarabo, Min Kim, Xin Tong, and Diego Gutierrez. “DeepToF: Off-the-Shelf Real-Time Correction of Multipath Interference in Time-of-Flight Imaging.” In: *ACM Transactions on Graphics (SIGGRAPH Asia 2017)* 36.6 (2017).

[180] Julio Marco, Wojciech Jarosz, Diego Gutierrez, and Adrian Jarabo. “Transient Photon Beams.” In: *Spanish Computer Graphics Conference (CEIG)* (2017).

[181] Julio Marco, Adrian Jarabo, Wojciech Jarosz, and Diego Gutierrez. “Second-Order Occlusion-Aware Volumetric Radiance Caching.” In: *ACM Transactions on Graphics* 37.2 (2018).

[182] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. “Stacked convolutional auto-encoders for hierarchical feature extraction.” In: *Proceedings of Int. Conf. Artificial Neural Networks*. 2011, pp. 52–59.

[183] Belen Masia, Gordon Wetzstein, Piotr Didyk, and Diego Gutierrez. “A survey on computational displays: Pushing the boundaries of optics, computation, and perception.” In: *Computers & Graphics* 37.8 (2013), pp. 1012–1038.

[184] Pavlos Mavridis and Georgios Papaioannou. “MSAA-Based Coarse Shading for Power-Efficient Rendering on High Pixel-Density Displays.” In: *High Performance Graphics*. 2015.

[185] Soham Uday Mehta, Brandon Wang, Ravi Ramamoorthi, and Fredo Durand. “Axis-aligned Filtering for Interactive Physically-based Diffuse Indirect Lighting.” In: *ACM Trans. Graph.* 32.4 (July 2013), 96:1–96:12.

[186] Stephan Meister, Rahul Nair, Bernd Jähne, and Daniel Kondermann. *Photon Mapping based Simulation of Multi-Path Reflection Artifacts in Time-of-Flight Sensors*. Tech. rep. Heidelberg Collaboratory for Image Processing, 2013.

[187] Stephan Meister, Rahul Nair, and Daniel Kondermann. “Simulation of Time-of-Flight Sensors using Global Illumination.” In: *Vision, Modeling & Visualization*. 2013.

[188] K. Mitra and S. Kumar. “Development and comparison of models for light-pulse transport through scattering-absorbing media.” In: *Applied Optics* 38.1 (1999).

[189] Hans P. Moravec. “3D graphics and the wave theory.” In: *SIGGRAPH Comput. Graph.* 15.3 (1981).

[190] T. Moshnyaga and E. Morikawa. "LCD display energy reduction by user monitoring." In: *IEEE international conference on computer design*. 2005.

[191] Adolfo Muñoz. "Higher Order Ray Marching." In: *Computer Graphics Forum* 33.8 (2014), pp. 167–176.

[192] A. Musbach, G. W. Meyer, F. Reitich, and S. H. Oh. "Full Wave Modelling of Light Propagation and Reflection." In: *Computer Graphics Forum* 32.6 (2013).

[193] NVIDIA Corporation. *NVIDIA Management Library*. https://developer.nvidia.com/nvidia-management-library-nvml. 2015.

[194] NVIDIA Corporation. *NVIDIA Global Citizenship: Energy Efficiency*. 2017.

[195] Nikhil Naik, Shuang Zhao, Andreas Velten, Ramesh Raskar, and Kavita Bala. "Single view reflectance capture using multiplexed scattering and time-of-flight imaging." In: *ACM Trans. Graph.* 30 (6 2011).

[196] Nikhil Naik, Christopher Barsi, Andreas Velten, and Ramesh Raskar. "Estimating wide-angle, spatially varying reflectance using time-resolved inversion of backscattered light." In: *JOSA A* 31.5 (2014).

[197] Nikhil Naik, Achuta Kadambi, Christoph Rhemann, Shahram Izadi, Ramesh Raskar, and Sing Bing Kang. "A light transport model for mitigating multipath interference in time-of-flight sensors." In: *IEEE Computer Vision and Pattern Recognition*. 2015.

[198] Rahul Nair, Stephan Meister, Martin Lambers, Michael Balda, Hannes Hofmann, Andreas Kolb, Daniel Kondermann, and Bernd Jähne. "Ground truth for evaluating time of flight imaging." In: *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. 2013.

[199] Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, H-P Seidel, and Tobias Ritschel. "Deep shading: convolutional neural networks for screen space shading." In: *Computer graphics forum*. Vol. 36. 4. Wiley Online Library. 2017, pp. 65–78.

[200] Srinivasa G Narasimhan, Mohit Gupta, Craig Donner, Ravi Ramamoorthi, Shree K Nayar, and Henrik Wann Jensen. "Acquiring scattering properties of participating media by dilution." In: *ACM Transactions on Graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 1003–1012.

[201] P. Narra and D. Zinger. "An effective LED dimming approach." In: *IEEE industry applications conference*. 2004.

[202] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. "All-frequency shadows using non-linear wavelet lighting approximation." In: *ACM Trans. Graph.* 22.3 (2003).

[203] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. "Progressive Virtual Beam Lights." In: *Computer Graphics Forum* 31.4 (2012).

[204] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. "Virtual Ray Lights for Rendering Scenes with Participating Media." In: *ACM Trans. Graph.* 31.4 (2012).

[205] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. "Monte Carlo Methods for Volumetric Light Transport Simulation." In: *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37.2 (2018). To appear.

[206] Matthew O'Toole, Felix Heide, Lei Xiao, Matthias B. Hullin, Wolfgang Heidrich, and Kiriakos N. Kutulakos. "Temporal Frequency Probing for 5D Transient Analysis of Global Light Transport." In: *ACM Trans. Graph.* 33.4 (2014).

[207] Rachel Orti, Stephane Riviere, Fredo Durand, and Claude Puech. "Radiosity for dynamic scenes in flatland with the visibility complex." In: *Computer Graphics Forum*. Vol. 15. 3. 1996, pp. 237–248.

[208] Rohit Pandharkar. "Hidden object doppler: estimating motion, size and material properties of moving non-line-of-sight objects in cluttered environments." PhD thesis. Massachusetts Institute of Technology, 2011.

[209] EM Patterson, CE Shelden, and BH Stockton. "Kubelka-Munk optical properties of a barium sulfate white reflectance standard." In: *Applied Optics* 16.3 (1977), pp. 729–732.

[210] Jon Peddie. "Trends and Forecasts in Computer Graphics – power-efficient rendering." In: *Jon Peddie Research*. 2013.

[211] Ofir Pele and Michael Werman. "The quadratic-chi histogram distance family." In: *Proceedings of ECCV*. Springer. 2010, pp. 749–762.

[212] Fabio Pellacini. "User-configurable Automatic Shader Simplification." In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 445–452.

[213] Vijitha Periyasamy and Manojit Pramanik. "Importance sampling-based Monte Carlo simulation of time-domain optical coherence tomography with embedded objects." In: *Applied Optics* 55.11 (2016).

[214] Ken Perlin. "Improving Noise." In: *ACM Trans. Graph.* 21.3 (2002).

[215] Christoph Peters, Jonathan Klein, Matthias B Hullin, and Reinhard Klein. "Solving trigonometric moment problems for fast transient imaging." In: *ACM Trans. Graph.* 34.6 (2015).

[216] Phil Pitts, Arrigo Benedetti, Malcolm Slaney, and Phil Chou. *Time of Flight Tracer*. Tech. rep. Microsoft, 2014.

[217] Jeff Pool. "Energy-precision tradeoffs in the graphics pipeline." PhD thesis. University of North Carolina at Chapel Hill, 2012.

[218] PowerVR. "PowerVR: A Master Class in Graphics Technology and Optimization." In: *Imagination Technologies*. 2012.

[219] Hui Qiao, Jingyu Lin, Yebin Liu, Matthias B Hullin, and Qionghai Dai. "Resolving transient time profile in ToF imaging via log-sum sparse regularization." In: *Opt. Lett.* 40.6 (2015).

[220] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. "A First-order Analysis of Lighting, Shading, and Shadows." In: *ACM Trans. Graph.* 26.1 (Jan. 2007).

[221] P. Ranganathan, E. Geelhoed, M. Manahan, and K. Nicholas. "Energy-aware user interfaces and energy-adaptive displays." In: *Computer* 39.3 (2006), pp. 31–38.

[222] Ramesh Raskar and James Davis. *5d time-light transport matrix: What can we reason about scene properties?* Tech. rep. MIT, 2008.

[223] Mickaël Ribardière, Samuel Carré, and Kadi Bouatouch. "Adaptive records for volume irradiance caching." In: *The Visual Computer* 27.6 (2011), pp. 655–664.

[224] H. Rief, A. Dubi, and T. Elperin. "Track Length Estimation Applied to Point Detector." In: *Nuclear Science and Engineering* 87 (1984).

[225] Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. "Image-space Control Variates for Rendering." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35.6 (2016), 169:1–169:12.

[226] Iman Sadeghi, Adolfo Muñoz, Philip Laven, Wojciech Jarosz, Francisco Seron, Diego Gutierrez, and Henrik Wann Jensen. "Physically-based Simulation of Rainbows." In: *ACM Trans. Graph.* 31.1 (2012).

[227] Henrik Schäfer, Frank Lenzen, and Christoph S Garbe. "Model based scattering correction in time-of-flight cameras." In: *Opt. Express* 22.24 (2014).

[228] Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben, and Jon Sporring. "Photon Differentials." In: *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. GRAPHITE '07. Perth, Australia: ACM, 2007, pp. 179–186.

[229] Rudolf Schwarte, Zhanping Xu, Horst-Guenther Heinol, Joachim Olk, Ruediger Klein, Bernd Buxbaum, Helmut Fischer, and Juergen Schulte. "New electro-optical mixing and correlating sensor: facilities and applications of the photonic mixer device (PMD)." In: *Lasers and Optics in Manufacturing III*. 1997.

[230] Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. "Practical Hessian-Based Error Control for Irradiance Caching." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31.6 (Nov. 2012).

[231] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 1992.

[232] Ana Serrano, Felix Heide, Diego Gutierrez, Gordon Wetzstein, and Belen Masia. "Convolutional sparse coding for high dynamic range imaging." In: *Computer Graphics Forum*. Vol. 35. 2. Wiley Online Library. 2016, pp. 153–163.

[233] F. Shearer. *Power Management in Mobile Devices*. Elsevier Inc., 2007.

[234] Shikhar Shrestha, Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. "Computational Imaging with Multi-Camera Time-of-Flight Systems." In: *ACM Trans. Graph.* 35.4 (2016).

[235] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor segmentation and support inference from RGBD images." In: *Computer Vision–ECCV 2012* (2012), pp. 746–760.

[236] Samuel Siltanen, Tapio Lokki, Sami Kiminki, and Lauri Savioja. "The room acoustic rendering equation." In: *J. Acoust. Soc. Am.* 122.3 (2007).

[237] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Taylor & Francis, 1986.

[238] Pitchaya Sitthi-amorn, Nicholas Modly, Westley Weimer, and Jason Lawrence. "Genetic programming for shader simplification." In: *ACM Trans. Graph.* 30.6 (2011), p. 152.

[239] Adam Smith, James Skorupski, and James Davis. *Transient Rendering*. Tech. rep. UCSC-SOE-08-26. School of Engineering, University of California, Santa Cruz, 2008.

[240] Ben Spencer and Mark W. Jones. "Into the blue: Better caustics through photon relaxation." In: *Computer Graphics Forum* 28.2 (2009), pp. 319–328.

[241] Efstathios Stavrakis, Marios Polychronis, Nectarios Pelekanos, Alessandro Artusi, Panayiotis Hadjichristodoulou, and Yiorgos Chrysanthou. "Toward energy-aware balancing of mobile graphics." In: *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 2015, pp. 94110D–94110D.

[242] Hao Su, Haoqiang Fan, and Leonidas Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image." In: *Proceedings of CVPR*. 2017.

[243] Shuochen Su, Felix Heide, Robin Swanson, Jonathan Klein, Clara Callenberg, Matthias Hullin, and Wolfgang Heidrich. "Material Classification Using Raw Time-Of-Flight Measurements." In: *IEEE Computer Vision and Pattern Recognition*. 2016.

[244] Shuochen Su, Felix Heide, Gordon Wetzstein, and Wolfgang Heidrich. "Deep End-to-End Time-of-Flight Imaging." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6383–6392.

[245] Xin Sun, Kun Zhou, Stephen Lin, and Baining Guo. "Line space gathering for single scattering in large scenes." In: *ACM Transactions on Graphics (TOG)*. Vol. 29. 4. ACM. 2010, p. 54.

[246] Frank Suykens and Yves D. Willems. "Path differentials and applications." In: *Rendering Techniques 2001 (Proceedings of the Eurographics Workshop on Rendering)*. Eurographics Association. London, United Kingdom, 2001, pp. 257–268.

[247] Ryuichi Tadano, Adithya Kumar Pediredla, Kaushik Mitra, and Ashok Veeraraghavan. "Spatial Phase-Sweep: Increasing temporal resolution of transient imaging using a light source array." In: *arXiv preprint arXiv:1512.06539* (2015).

[248] Michael D Tocci, Chris Kiser, Nora Tocci, and Pradeep Sen. "A versatile HDR video production system." In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 4. ACM. 2011, p. 41.

[249] Chia-Yin Tsai, Ashok Veeraraghavan, and Aswin C Sankaranarayanan. "Shape and Reflectance from Two-Bounce Light Transients." In: *IEEE International Conference on Computational Photography*. 2016.

[250] Karthik Vaidyanathan, Marco Salvi, Robert Toth, Tim Foley, Tomas Akenine-Möller, Jim Nilsson, Jacob Munkberg, Jon Hasselgren, Masamichi Sugihara, Petrik Clarberg, et al. "Coarse Pixel Shading." In: *High Performance Graphics*. 2014.

[251] Keith S. Vallerio, Lin Zhong, and Niraj K. Jha. "Energy-Efficient Graphical User Interface Design." In: *IEEE Transactions on Mobile Computing* 5.7 (July 2006), pp. 846–859.

[252] Adriaan Van Oosterom and Jan Strackee. "The Solid Angle of a Plane Triangle." und. In: *IEEE Transactions on Biomedical Engineering* BME-30.2 (1983), pp. 125 –126.

[253] Elena Vasiou, Konstantin Shkurko, Ian Mallett, Erik Brunvand, and Cem Yuksel. "A detailed study of ray tracing performance: render time and energy cost." In: *The Visual Computer* (2018), pp. 1–11.

[254] Eric Veach. "Robust Monte Carlo methods for light transport simulation." PhD thesis. Stanford, 1997.

[255] Eric Veach and Leonidas J. Guibas. "Optimally combining sampling techniques for Monte Carlo rendering." In: *SIGGRAPH '95*. 1995.

[256] Eric Veach and Leonidas J. Guibas. "Metropolis Light Transport." In: *SIGGRAPH '97*. 1997.

[257] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Moungi G. Bawendi, and Ramesh Raskar. "Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging." In: *Nature Communications* 3 (2012).

[258] Andreas Velten, Di Wu, Adrian Jarabo, Belen Masia, Christopher Barsi, Everett Lawson, Chinmaya Joshi, Diego Gutierrez, Moungi G. Bawendi, and Ramesh Raskar. "Relativistic ultrafast rendering using time-of-flight imaging." In: *ACM SIGGRAPH 2012 Talks*. 2012.

[259] Andreas Velten, Di Wu, Adrian Jarabo, Belen Masia, Christopher Barsi, Chinmaya Joshi, Everett Lawson, Moungi Bawendi, Diego Gutierrez, and Ramesh Raskar. "Femto-Photography: Capturing and Visualizing the Propagation of Light." In: *ACM Trans. Graph.* 32.4 (2013).

[260] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. "Embree: a kernel framework for efficient CPU ray tracing." In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 143.

[261] Peng Wang, Xiaohui Shen, Zhe Lin, S. Cohen, B. Price, and A. Yuille. "Towards unified depth and semantic prediction from a single image." In: *Proceedings of CVPR*. 2015.

[262] Rui Wang, Xianjin Yang, Yazhen Yuan, Wei Chen, Kavita Bala, and Hujun Bao. "Automatic Shader Simplification Using Surface Signal Approximation." In: *ACM Trans. Graph.* 33.6 (Nov. 2015), 226:1–226:11.

[263] Rui Wang, Bowen Yu, Julio Marco, Tianlei Hu, Diego Gutierrez, and Hujun Bao. "Real-time Rendering on a Power Budget." In: *ACM Transactions on Graphics (SIGGRAPH 2016)* 35.4 (2016).

[264] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. "Image quality assessment: from error visibility to structural similarity." In: *Image Processing, IEEE Transactions on* 13.4 (2004), pp. 600–612.

[265] Greg Ward. *Radiance: A Validated Lighting Simulation Tool*. https://www.radiance-online.org. 2017.

[266] Gregory J. Ward and Paul S. Heckbert. "Irradiance Gradients." In: 1992, pp. 85–98.

[267] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. "A Ray Tracing Solution for Diffuse Interreflection." In: *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '88. New York, NY, USA: ACM, 1988, pp. 85–92.

[268] Andrea Weidlich and Alexander Wilkie. "Realistic Rendering of Birefringency in Uniaxial Crystals." In: *ACM Trans. Graph.* 27.1 (2008).

[269] Daniel Weiskopf, Ute Kraus, and Hanns Ruder. "Searchlight and Doppler effects in the visualization of special relativity: a corrected derivation of the transformation of radiance." In: *ACM Trans. Graph.* 18.3 (1999).

[270] Stephen H. Westin, James R. Arvo, and Kenneth E. Torrance. "Predicting reflectance functions from complex surfaces." In: *SIGGRAPH '92*. 1992.

[271] Gordon Wetzstein, Ivo Ihrke, Douglas Lanman, and Wolfgang Heidrich. "Computational plenoptic imaging." In: *Computer Graphics Forum*. Vol. 30. 8. Wiley Online Library. 2011, pp. 2397–2426.

[272] Refael Whyte, Lee Streeter, Michael J Cree, and Adrian A Dorrington. "Resolving multiple propagation paths in time of flight range cameras using direct and global separation methods." In: *Opt. Eng.* 54.11 (2015).

[273] Alexander Wilkie, Robert F. Tobler, and Werner Purgathofer. "Combined Rendering of Polarization and Fluorescence Effects." In: *Eurographics Workshop on Rendering Techniques '01*. 2001.

[274] Michael Maurice Rudolph Williams. "Mathematical Methods in Particle Transport Theory." In: (1971).

[275] R. Woo, Chi-Weon Yoon, Jeonghoon Kook, Se-Joong Lee, and Hoi-Jun Yoo. "A 120-mW 3-D rendering engine with 6-Mb embedded DRAM and 3.2-GB/s runtime reconfigurable bus for PDA chip." In: *Solid-State Circuits, IEEE Journal of* 37.10 (2002), pp. 1352–1355.

[276] Di Wu, Matthew O'Toole, Andreas Velten, Amit Agrawal, and Ramesh Raskar. "Decomposing global light transport using time of flight imaging." In: *IEEE Computer Vision and Pattern Recognition*. 2012.

[277] Di Wu, Gordon Wetzstein, Christopher Barsi, Thomas Willwacher, Matthew O'Toole, Nikhil Naik, Qionghai Dai, Kyros Kutulakos, and Ramesh Raskar. "Frequency Analysis of Transient Light Transport with Applications in Bare Sensor Imaging." In: *European Conference on Computer Vision*. 2012.

[278] Di Wu, Andreas Velten, Matthew O'Toole, Belen Masia, Amit Agrawal, Qionghai Dai, and Ramesh Raskar. "Decomposing Global Light Transport using Time of Flight Imaging." In: *International Journal of Computer Vision* 107.2 (2014).

[279] Rihui Wu, Adrian Jarabo, Jinli Suo, Feng Dai, Yongdong Zhang, Qionghai Dai, and Diego Gutierrez. "Adaptive polarization-difference transient imaging for depth estimation in scattering media." In: *Opt. Lett.* 6 (2018), pp. 1299–1302.

[280] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. "Sun3d: A database of big spaces reconstructed using sfm and object labels." In: *Proceedings of ICCV*. 2013, pp. 1625–1632.

[281] Lei Xiao, Felix Heide, Matthew O'Toole, Andreas Kolb, Matthias B Hullin, Kyros Kutulakos, and Wolfgang Heidrich. "Defocus deblurring and superresolution for time-of-flight depth cameras." In: *IEEE Computer Vision and Pattern Recognition*. 2015.

[282] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. "Multi-Scale Continuous CRFs as Sequential Deep Networks for Monocular Depth Estimation." In: *Proceedings of CVPR*. 2017.

[283] Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. "Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces." In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 33.4 (2014).

[284] Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. "Position-Normal Distributions for Efficient Rendering of Specular Microstructure." In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 35.4 (2016).

[285] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. "Image Super-Resolution Via Sparse Representation." In: *IEEE TIP* 19.11 (2010).

[286] K. M. Yoo and R. R. Alfano. "Time-resolved coherent and incoherent components of forward light scattering in random media." In: *Opt. Lett.* 15.6 (1990).

[287] Jure Žbontar and Yann LeCun. “Computing the Stereo Matching Cost with a Convolutional Neural Network.” In: *Proceedings of CVPR*. 2015.

[288] Tizian Zeltner and Wenzel Jakob. “The layer laboratory: a calculus for additive and subtractive composition of anisotropic surface reflectance.” In: *ACM Transactions on Graphics* 37.4 (2018), p. 74.

[289] Yong Zhang, Hongliang Yi, and Heping Tan. “One-dimensional transient radiative transfer by lattice Boltzmann method.” In: *Opt. Express* 21.21 (2013).

[290] Yunjin Zhang, Marta Ortin, Victor Arellano, Rui Wang, Diego Gutierrez, and Hujun Bao. “On-the-Fly Power-Aware Rendering.” In: *Computer Graphics Forum* (2018).

[291] C. Zhu and Q. Liu. “Review of Monte Carlo modeling of light transport in tissues.” In: *J. Biomed. Opt.* 18.5 (2013).

[292] Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. “Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering.” In: *Computer Graphics Forum (Proceedings of Eurographics)* 34.2 (May 2015), pp. 667–681.

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

Final Version as of November 9, 2018 (version 4.2).