Efficient Many-Light Rendering of Scenes with Participating Media

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik

des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Jan Novák

aus Turnov

Tag der mündlichen Prüfung: 09. Mai 2014
Erster Gutachter: Prof. Dr.-Ing. Carsten Dachsbacher
Zweiter Gutachter: Prof. Dr. Ir. Philip Dutré
Dritter Gutachter: Dr. Wojciech Jarosz

Vita and Publications

2009–2013	Research and teaching assistant Karlsruhe Institute of Technology, Germany
2007–2009	Master's degree in Electrical Engineering and Informatics Czech Technical University in Prague, Czech Republic
2003–2007	Bachelor's degree in Electrical Engineering and Informatics Czech Technical University in Prague, Czech Republic

Schmidt, T.-W., Novák, J., Meng, J., Kaplanyan, A. S., Reiner, T., Nowrouzezahrai, D., and Dachsbacher, C. [2013]. Path-space manipulation of physically-based light transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4):129:1–129:11.

Ulbrich, J., Novák, J., Rehfeld, H., and Dachsbacher, C. [2013]. Progressive visibility caching for fast indirect illumination. In *Proc. of Vision, Modeling and Visualization*, pp. 203–210. Eurographics Association.

Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., and Novák, J. [2014]. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33(1):88–104.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. [2012a]. Progressive virtual beam lights. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 31(4):1407–1413.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. [2012b]. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):60:1–60:11.

Novák, J. and Dachsbacher, C. [2012]. Rasterized bounding volume hierarchies. *Computer Graphics Forum (Proc. of Eurographics)*, 31(3):403–412.

Engelhardt, T., Novák, J., Schmidt, T.-W., and Dachsbacher, C. [2012]. Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 31(7):2145–2154.

Novák, J., Engelhardt, T., and Dachsbacher, C. [2011a]. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 119–124. ACM.

Novák, J., Havran, V., and Dachsbacher, C. [2011b]. Path regeneration for random walks. In *GPU Computing Gems*, edited by W. mei W. Hwu, chapter 26, pp. 401–411. Morgan Kaufmann.

Engelhardt, T., Novák, J., and Dachsbacher, C. [2010]. Instant multiple scattering for interactive rendering of heterogeneous participating media. Tech. rep., Karlsruhe Institute of Technology, Germany.

Novák, J., Havran, V., and Dachsbacher, C. [2010]. Path regeneration for interactive path tracing. In *Eurographics 2010 - Short Papers*, pp. 61–64. Eurographics Association.

iv

Acknowledgments

I would like to take the opportunity to thank many people who encouraged and helped me during my academic career. Without them, this dissertation would not have been possible.

I want to begin by expressing my deep gratitude to my parents, Milada and Jaroslav, my sister Kateřina, and my grandparents for their continuous emotional and financial support throughout my life. Their advise and encouragement during my studies was key to my success.

I am deeply obliged to my advisor Carsten Dachsbacher, who has motivated me and served as a great source of inspiration throughout my academic career. I thank him for being responsive and constructively critical, and for creating a friendly and stimulating environment that turned out to be essential for my Ph.D. studies. I would also like to thank my colleagues from the KIT's computer graphics group, in particular: Stephan Bergmann, Raphael Diziol, Thomas Engelhardt, Johannes Hanika, Anton Kaplanyan, Gábor Liktor, Johannes Meng, Gregor Mückl, Hauke Rehfeld, Tim Reiner, Max-Gerd Retzlaff, Thorsten Schmidt, Florian Simon, and Justus Ulbrich, with many of whom I had the privilege of collaborating and sharing the good and evil of writing scientific papers. Special thanks go to Gregor Mückl for helping me with the translation of the abstract and summary, and to Diana Kheil for explaining the meaning of countless German forms to me.

I am very grateful to Wojciech Jarosz and Derek Nowrouzezahrai who invited me to Disney Research Zürich for an internship, which developed into a much longer collaboration that had a profound impact on my professional growth. I owe them for fruitful discussions and influential viewpoints that helped me tackle many research problems. I also want to thank Vlastimil Havran, who established my initial interests in computer graphics, and to Christophe Hery, who brought me to the Pixar Animation Studios and introduced me to the intricacies of production rendering. My deep respect goes to colleagues from the Walt Disney Animation Studios, the Hyperion team in particular, for a welcoming and motivating atmosphere with a unique blend of art and technology.

Finally, I thank Karolína, my fiancée, for her support and sincere interest in what I have chosen to pursue, for patiently listening to my lengthy, often confusing explanations, and for sparking occasional activities in my right brain when talking about art and psychology. I owe her such immense gratitude for tirelessly overcoming large geographical distances that I was constantly putting in between us. Without her patience and encouragement, I would not have been able to write these lines. I dedicate this thesis to her.

to Karolína

Contents

List of Figures

List of Tables

Abstract

Many-light rendering has proven to be an efficient and versatile solution for generating synthetic images of virtual scenes. The concept of representing all lighting in the scene using a set of carefully chosen *virtual point lights* has, however, its specific pitfalls. In particular, contracting all radiant energy into a finite set of infinitesimal emitters leads to singularities, which make high-quality results hard to obtain. Most existing techniques thus simply accept visual artifacts or some form of a systematic error. We present several approaches based on virtual lights that aim at capturing the light transport without compromising quality, and while preserving the elegance and efficiency of many-light rendering.

We first analyze a crucial component of high-quality rendering with virtual point lights, the so-called bias compensation that counteracts the systematic error. By reformulating the integration scheme, we obtain two numerically efficient techniques; one tailored specifically for interactive, high-quality lighting on surfaces, and one for handling scenes with participating media. We then continue investigating the more general problem of solving the light transport in the presence of participating media and present two lighting primitives for many-light rendering: the *virtual ray light* and the *virtual beam light*. Representing scattered light with the first primitive significantly reduces the degree of the singularity in the integrand—the major deficiency of rendering with virtual point lights—thereby minimizing the artifacts and allowing for more efficient unbiased computation. The second primitive then avoids the singularity completely by redistributing the emitted energy over the volume of the beam. We demonstrate that these two lighting primitives enable faster convergence and provide better temporal stability than traditional many-light methods based on virtual point lights.

We also address the main bottleneck of (not only) many-light rendering—the visibility testing—and present a novel accelerating structure for fast, approximate ray tracing. Our *rasterized bounding volume hierarchies* decouple the accelerator from the input geometry by representing the geometry as a collection of hierarchically organized height fields. We show that in addition to fast ray tracing, the hierarchy has a low memory footprint, provides inherent surface parametrization, and natively supports level-of-detail rendering. We demonstrate its advantages in various applications.

Abstrakt

Der sogenannte Many-light Rendering Ansatz hat sich als effiziente und vielseitige Lösung für die physikalisch-basierte Berechnung des Lichttransports für die Erstellung synthetischer Bilder virtueller Szenen bewährt. Dieses Konzept, bei dem die gesamte Beleuchtung in der Szene durch eine Menge von *virtuellen Punktlichtquellen* approximiert wird, weist jedoch besondere Probleme auf, die bei der Bilderzeugung berücksichtigt werden müssen. Insbesondere führt das Konzentrieren der gesamten Strahlungsenergie auf eine endlichen Menge infinitesimaler Quellen zu Singularitäten, die als störende Artefakte in den erzeugten Bildern sichtbar sind. Die meisten der existierenden Verfahren aus dieser Klasse von Ansätzen nehmen diese Artefakte in Kauf, oder vermeiden sie durch Begrenzung der Singularitäten, was jedoch zu systematischen Fehlern (engl. bias) führt. In dieser Arbeit werden mehrere auf virtuellen Punktlichtquellen aufbauende Ansätze vorgestellt, deren Ziel es ist, den Lichttransport in einer virtuellen Szene ohne sichtbare Qualitätsverluste bei der Darstellung zu erfassen und dabei die Eleganz und Effizienz der Many-light Rendering-Methoden beizubehalten.

Zunächst wird der Ansatz der sogenannten Bias Compensation analysiert, der eine Möglichkeit darstellt dem systematischen Fehler entgegenzuwirken und eine wichtige Komponente für hochqualitatives Rendering ist. Eine Umformulierung des Integrationsschemas führt zu zwei numerisch effizienten Verfahren, wobei sich eines auf interaktive, qualitativ hochwertige Beleuchtung von Oberflächen konzentriert und das andere auf die Berechnung in Szenen mit Flächen und partizipierenden Medien. Das letztere, allgemeinere Problem des Lichttransports wird in später weiter untersucht und zwei neue Beleuchtungsprimitive für das Many-light Rendering vorgestellt: das *Virtual Ray Light* und das *Virtual Beam Light*. Die Darstellung des gestreuten Lichts in partizipierenden Medien durch das erstgenannte Primitiv, entlang von Strahlen anstatt an Punkten konzentriert, reduziert den Grad der Singularität und somit die Artefakte erheblich. Das zweite Primitiv vermeidet diese Singularitäten vollständig, indem die abgestrahlte Energie über das Volumen eines Strahlenbündels (beam) verteilt wird. Es wird gezeigt, dass diese Berechnung des Lichttransports mit diesen Beleuchtungsprimitiven schneller konvergiert und zudem zeitlich kohärentere Resultate als traditionelle Many-light Rendering-Methoden, die auf virtuellen Punktlichtquellen basieren, liefert.

Letztlich wird außerdem eine für die Performanz der Many-light Rendering-Methoden (und anderer Ray Tracing Verfahren) kritische Operation—der Sichtbarkeitstest—untersucht. Es wird eine neuartige Beschleunigungsstruktur für schnelles, approximatives Ray Tracing vorgestellt. Diese *Rasterized Bounding Volume Hierarchies* entkoppeln die Repräsentation die Geometrie für Schnittberechnungen von der Eingabegeometrie, indem diese als eine hierarchische Ansammlung von Höhenkarten, also bildbasiert, dargestellt wird. Es wird gezeigt, dass diese Datenstruktur nicht nur schnelles Ray Tracing erlaubt, sondern zudem geringeren Speicherbedarf benötigt, eine inhärente Parametrisierung der Oberflächen, sowie adaptives Detail bietet. Diese Aspekte werden anhand von unterschiedlichen Anwendungsfällen untersucht.

Summary

Creating photorealistic images—one of the extensively sought goals of computer graphics—is an intricate and computationally intensive task. Many applications that demand realistic-looking images require the propagation of light to be simulated accurately and with emphasis on physical correctness. In practice, this amounts to not only evaluating integro-differential equations that govern light transport, but also carrying out the evaluation in a reasonable time frame. This becomes challenging especially in scenes with participating media, where light scatters on surfaces as well as inside volumes. In this thesis, we propose several new algorithms and improvements to existing approaches that aim at efficient rendering of such scenes.

Many-Light Algorithms

We build atop *instant radiosity* [Keller 1997], an industry-verified algorithm that approximates costly light transport using a collection of *virtual point lights* (VPLs); thus it is commonly referred to as a *many-light* approach. In a nutshell, many-light algorithms trace a number of photon paths from light sources, create a VPL whenever a photon is reflected off a surface or scatters inside a medium (see Figure 1.a), and use these VPLs to illuminate the scene and thus approximate the expensive multi-bounce light transport. In Chapter 3, we provide a detailed explanation and an overview of existing many-light techniques; the chapter is based on a state-of-the-art survey:

> DACHSBACHER, C., KŘIVÁNEK, J., HAŠAN, M., ARBREE, A., WALTER, B., and NOVÁK, J. [2014]. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33(1):88–104.

In contrast to other approaches, many-light algorithms allow for coarse approximations in real-time, accurate rendering in minutes, as well as convergence in the limit. While being extremely efficient and versatile, many-light rendering suffers from a distinct drawback: the underlying mathematical formulation is plagued by a singularity. The singularity stems from contracting all the energy in the scene into a finite number of points, the VPLs. When a VPL illuminates the scene, its contribution to each receiving point is proportional to the inverse squared distance between the two points. Nearby surfaces and volumes thus receive significantly more energy then distant ones, causing the rendered image to suffer from distracting, high intensity "splotches".

These artifacts can be reduced by creating more VPLs; however, avoiding them completely is beyond computational resources of any reasonable renderer. Indeed, the only option to truly remove them is to *bound* the contribution of each VPL by some user defined maximum. The more we bound the fewer artifacts we have; however, at the same time we also selectively remove energy and thereby bias the computation. The energy loss leads to artificial darkening in concave areas and change in material appearance; and should be thus avoided or compensated for.

(a) Virtual Point Lights (VPLs)   **(b)** Virtual Ray Lights (VRLs)   **(c)** Virtual Beam Lights (VBLs)
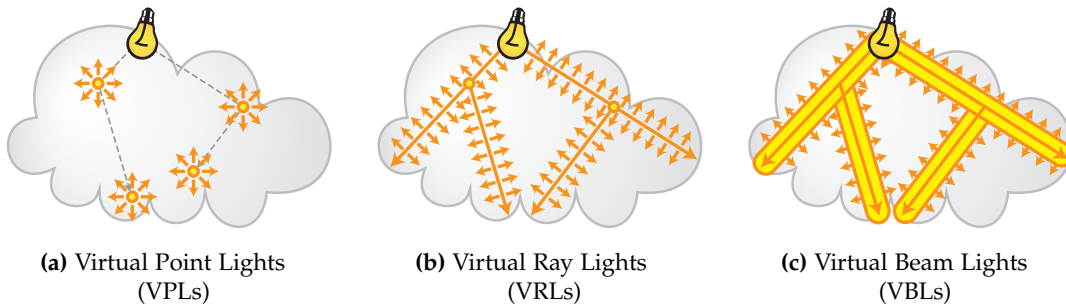
Figure 1: Many-light algorithms represent multi-bounce illumination with a set of virtual lights. In addition to improving the existing VPL-based approach (a), we present two new primitives: virtual ray lights (b) and virtual beam lights (c), that overcome the characteristic problem of point-based many-light techniques and enable more efficient integration of the light transport.

In Chapter 4, we study Monte Carlo approaches for recovering the lost energy. We note that existing techniques, despite their need to recover only a small portion of the overall illumination, often increase the rendering time by orders of magnitude and clutter the otherwise elegant algorithm. We propose a number of approximations that improve the efficiency and simplify the bias compensation. We demonstrate that the amount of recovered energy quickly drops with each additional bounce of light, and thus not more than two or three bounces are usually required. In the presence of participating media, we also show that assuming local homogeneity and no occlusion makes the technique GPU-friendly and significantly accelerates rendering of heterogeneous media. Based on these findings, we propose an *approximate bias compensation* that recovers the missing illumination at low costs and thus preserves the efficiency of point-based many-light algorithms. The technique is tailored for scenes with participating media, and was published in:

> ENGELHARDT, T., NOVÁK, J., SCHMIDT, T.-W., and DACHSBACHER, C. [2012]. Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 31(7):2145–2154.

We also demonstrate that the illumination that is removed by bounding, and successively recovered using the bias compensation, is highly localized. This allows us to reformulate the rendering algorithm and explicitly split the computation into: *bounded* light transport between distant points, which is computed using VPLs; and *residual* transport between nearby points. In Chapter 5, we show that the residual transport between surfaces can be estimated at interactive frame rates using a hierarchical integration in screen space. The technique was presented in:

> NOVÁK, J., ENGELHARDT, T., and DACHSBACHER, C. [2011a]. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 119–124. ACM.

The major drawback of all bias compensation techniques is that they in some sense reduce the elegance and simplicity of the original approach. The problem is conceptual: we first remove short-distance illumination of VPLs to subsequently recover it using a different integration scheme. This would not have been necessary if we managed to avoid the problem-causing singularity in the first place.

In participating media, scattering of light can be formulated continuously along the directions of travel. We leverage this observation and replace VPLs with *virtual ray lights* (VRLs), which are formed by entire linear segments instead of just vertices of the photon path (see Figure 1.b). Although computing the incident light due to a ray light is slightly more expensive than with a point light—we need to integrate the emitted energy along the length of the ray—distributing the energy along lines provably lowers the degree of the singularity. This in practice reduces the major weakness of many-light algorithms, i.e. the distracting "splotchy" artifacts, improves the convergence and temporal stability, and brings down the overall rendering cost. Virtual ray lights and their efficient sampling are described in detail in Chapter 6 and appeared in:

> NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., and JAROSZ, W. [2012b]. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):60:1–60:11.

In order to avoid the singularity completely, we propose to further inflate the VRL into a *virtual beam light* (VBL). Our motivation is to distribute the energy of the infinitesimal ray light over a cylindrical region with finite thickness (see Figure 1.c). As such, the singularity disappears from the integrand and we no longer need to bound the light transport and subsequently compensate for the missing energy. The price to pay is the slight overblurring of illumination; however, we formulate the rendering algorithm progressively, ensuring that the thickness of each VBL reduces over time, the blurring diminishes, and the estimation is asymptotically consistent. VBLs better preserve the appearance of materials and media, do not require special treatment to avoid artifacts, and represent the current state of the art in rendering scenes with participating media using virtual lights. The algorithm was originally published in:

> NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., and JAROSZ, W. [2012a]. Progressive virtual beam lights. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 31(4):1407–1413.

Acceleration of Visibility Computation

One of the common denominators of all rendering algorithms is that they need to resolve the mutual visibility between points in the scene. Many-light methods are no exception to this; the visibility is evaluated when tracing photons as well as tested when connecting shading points to virtual lights. In order to accelerate visibility computation, we present the *rasterized bounding volume hierarchy* (RBVH); an acceleration structure for fast, approximate ray tracing.

Our key idea is to represent finely tessellated, detail geometry with a collection of height fields (i.e. raster images storing elevation data) that are organized into a hierarchy. During construction, we identify parts that can be well represented with a single, arbitrarily-oriented height field and rasterize and organize them into a hierarchy of bounding volumes. To trace rays, we traverse the hierarchy and search for intersections with nearby height fields. In practice, this approach is faster than using traditional, polygon-based acceleration structures. RBVHs also provide inherent surface parameterization, which enables applications such as real-time on-surface painting and irradiance caching. Details are presented in Chapter 8, which is based on:

> NOVÁK, J. and DACHSBACHER, C. [2012]. Rasterized bounding volume hierarchies. *Computer Graphics Forum (Proc. of Eurographics)*, 31(3):403–412.

Zusammenfassung

Das Erstellen photorealistischer Bilder—seit Anbeginn eines der Ziele der Computergrafik—ist ein aufwändiger und rechenintensiver Prozess. Dies ist vor allem der Fall, da hierzu die Lichtausbreitung in einer virtuellen Szene berechnet werden muss. Viele Anwendungen sind zudem darauf angewiesen, dass diese Simulation präzise und auf der Basis physikalischer Modelle berechnet wird. In der Praxis kommt hinzu, dass die Auswertung der zugrundeliegenden Integro-Differentialgleichungen, die die Licht-Material-Interaktionen beschreiben, innerhalb vertretbarer Zeitspannen erfolgen muss. Dies stellt insbesondere in Szenen mit partizipierenden Medien eine Herausforderung dar, da in diesen das Licht nicht nur an Oberflächen, sondern an praktisch jeder Stelle im Raum gestreut werden kann. In dieser Arbeit werden mehrere neue Algorithmen, sowie Verbesserungen existierender Ansätze, vorgestellt, deren Ziel die effiziente Bildsynthese solcher Szenen ist.

Many-Light-Algorithmen

Der Ursprung des Many-light Rendering Ansatzes ist das *instant radiosity*-Verfahren [Keller 1997], in dem die Grundidee, den Lichttransport in einer Szene durch eine Ansammlung von virtuellen Punktlichtquellen (VPL) darzustellen, erstmals vorgestellt wurde. Im Prinzip berechnen Many-light-Algorithmen eine verhältnismäßig geringe Zahl von Photonen- oder Lichttransportpfaden ausgehend von den Lichtquellen einer Szene und erzeugen eine VPL, wo ein Photon an einer Oberfläche reflektiert oder in einem Volumen gestreut wird (siehe Abbildung 2.a). Die direkte Beleuchtung der Szene durch diese VPLs stellt dann eine Approximation des gesamten Lichttransports dar. Kapitel 3 stellt die zugrundeliegende Theorie dar und bietet einen Überblick über existierende Many-light Rendering-Verfahren; dieses Kapitel basiert auf einem Artikel:

DACHSBACHER, C., KŘIVÁNEK, J., HAŠAN, M., ARBREE, A., WALTER, B., and NOVÁK, J. [2014]. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33(1):88–104.

Eine Stärke des Many-light Ansatzes ist dessen Skalierbarkeit, die es erlaubt eine grobe Näherung in Echtzeit zu berechnen, vergleichsweise genaue Resultate in wenigen Minuten zu erhalten, und gleichzeitig Konvergenz gegen die tatsächliche Lösung garantiert. Trotz der hohen Effizienz und Vielseitigkeit weist der Ansatz einen Nachteil auf: die zugrundeliegende mathematische Formulierung enthält eine Singularität, die anschaulich durch die Kontraktion des gesamten Lichttransports einer Szene auf eine endliche Anzahl von Punkten, den VPLs, entsteht. Berechnet man die Beleuchtung der Szene durch eine VPL, so ist ihr Beitrag zu einem Oberflächenpunkt proportional zur umgekehrten quadratischen Entfernung. Nahegelegene Oberflächen und Punkte in Volumina erhalten daher deutlich mehr Energie als weiter entfernte, was zu Artefakten in Form von kleinen, hellen Bildregionen führt.

Diese Artefakte können durch die Erzeugung weiterer VPLs reduziert werden; jedoch ist es nahezu unpraktikabel sie dadurch komplett zu vermeiden. Tatsächlich ist die einzige Möglichkeit zur Beseitigung dieser Artefakte, den Beitrag jeder VPL zu einem Punkt auf einer Fläche bzw. im Volumen durch ein benutzerdefiniertes Maximum zu beschränken. Je stärker der Beitrag beschränkt wird, desto weniger Artefakte treten auf; allerdings wird gleichzeitig lokal Energie entfernt und ein systematischer Fehler (engl. Bias) in die Berechnung eingeführt. Dieser Energieverlust führt zu einer Abdunkelung konkaver Flächenstücke und zu Änderung des Aussehens von Materialien und sollte daher vermieden oder besser ausgeglichen werden (engl. Bias Compensation).

In Kapitel 4 werden Monte-Carlo-Ansätze zur Quantifizierung (und somit zur Zurückgewinnung) dieser entfernten Energie untersucht. Es wird gezeigt, dass diese Energiemenge nur einen geringen Teil der transportierten Energie darstellt, existierende Verfahren aber die Gesamtrechenzeit oft um Größenordnungen erhöhen. Es werden eine Reihe von Näherungen vorgestellt, die die Effizient der Zurückgewinnung deutlich steigern und deren Berechnung vereinfachen. Insbesondere wird ausgenutzt, dass die Menge zurückgewonnener Energie mit jeder zusätzlichen Reflexion exponentiell abnimmt. Außerdem wird gezeigt, dass in partizipierenden Medien, durch die Annahmen lokaler Homogenität und fehlender Verdeckung, das Verfahren effizient auf Grafik-Hardware durchführbar ist und die Bildsynthese mit heterogenen Medien signifikant beschleunigt werden kann. Ausgehend von diesen Resultaten erstellen wir eine *Approximate Bias Compensation* vor, die die fehlende Energie mit niedrigen Kosten wiederherstellt und dadurch die Effizienz der Many-light-Methoden bewahrt. Das Verfahren zielt ist auf Szenen mit partizipierenden Medien aus und wurde in:

> Engelhardt, T., Novák, J., Schmidt, T.-W., and Dachsbacher, C. [2012]. Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 31(7):2145–2154

veröffentlicht.

Ein weiteres Charakteristikum ist, dass die zurückgewonnene Energie nur in einem räumlich sehr beschränkten Bereich nennenswerte Beiträge liefert. Dies erlaubt es, die Lichttransportgleichung umzuformulieren und die Berechnung explizit aufzuteilen: in einen *beschränkten* Lichttransport zwischen weit entfernten Punkten, der mit VPLs berechnet wird, und einem *residualen* Lichttransport zwischen benachbarten Punkten. In Kapitel 5 wird gezeigt, dass der verbleibende Lichttransport zwischen Oberflächen mittels einer hierarchischen Integration im Bildraum abgeschätzt werden kann und dies Berechnungen in interaktiver Geschwindigkeit ermöglicht. Dieses Verfahren wurde in:

> Novák, J., Engelhardt, T., and Dachsbacher, C. [2011a]. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 119–124. ACM

vorgestellt.

Trotz dieser Verbesserungen bleibt der wesentlicher Nachteil aller bias compensation-Verfahren, dass sie in irgendeiner Weise die Eleganz und Einfachheit des ursprünglichen Ansatzes reduzieren. Dabei handelt es sich allerdings um ein konzeptionelles Problem: zuerst werden die Beiträge von VPLs über kurze Entfernungen begrenzt (um Artefakte zu vermeiden), um den

(a) Virtuelle Punktlichtquellen (VPLs)

(b) Virtuelle lineare Lichtquellen (VRLs)

(c) Virtuelle Bündeln (VBLs)
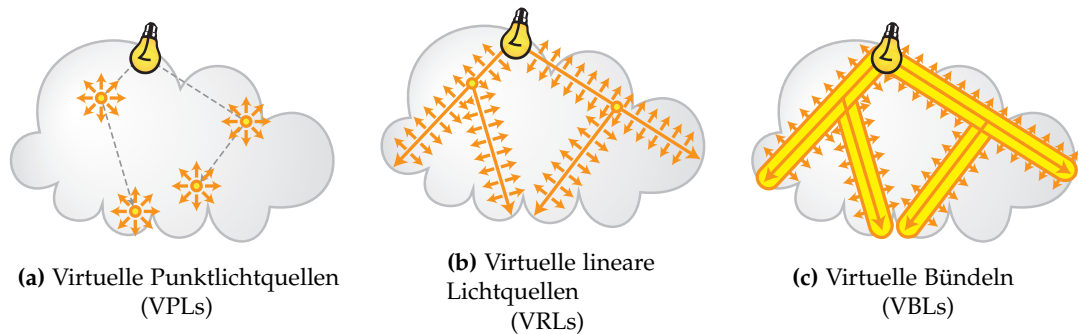
Abbildung 2: Many-light-Ansätze stellen mehrfach reflektiertes Licht als direkte Beleuchtung durch eine Menge von virtuellen Lichtquellen dar. Neben Verbesserungen an bestehenden VPL-basierten Ansätzen (a) werden in dieser Arbeit zwei neue Beleuchtungsprimitive vorgestellt: virtuelle lineare Strahllichter (b) und Virtuelle Bündeln (c). Diese reduzieren bzw. vermeiden das inhärente Problem der punktbasierten Many-Light-Ansätze und führen zu einer effizienteren Berechnung des Lichttransports.

fehlenden Lichttransport danach mittels eines anderen Integrationsschemas wieder hinzuzufügen. Dies wäre nicht nötig, wenn die Singularität, die dieses Problem erzeugt, von vornherein vermieden werden könnte.

In Kapitel 6 wird gezeigt, dass es in partizipierenden Medien möglich ist, das Konzept von virtuellen Punktlichtquellen durch *lineare Lichtquellen* (Virtual Ray Lights, VRLs) zu ersetzen. Diese stellen die Verbindungsstrecke zweier Interaktionen entlang eines Photonenpfades dar (siehe Abbildung 2.b). Die Berechnung des einfallenden Lichts von einer VRL ist etwas teurer ist als von einem Punktlicht, da die abgestrahlte Energie über den ganzen Strahl hinweg integriert werden muss. Dennoch verringert das Verteilen der Energie über die ganze Länge des Strahls den Grad der Singularität und reduziert letztendlich die Gesamtkosten für die Bildsynthese, da die Artefakte signifikant reduziert werden. VRLs und deren effiziente Abtastung werden in Kapitel 6 detailliert beschrieben und sind in:

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., and JAROSZ, W. [2012b]. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):60:1–60:11

erschienen.

Um die Singularität vollständig zu vermeiden, können die strahlförmigen VRLs zu *virtuellen Bündeln* (Virtual Beam Lights, VBLs) ausgedehnt werden. Die zugrundeliegende Idee ist, die Energie des infinitesimalen Strahllichts über ein zylindrisches Gebiet mit endlicher Dicke auszudehnen (siehe Abbildung 2.c). Dadurch verschwindet die Singularität gänzlich aus dem Integranden und es ist nicht mehr nötig, den Lichttransport zu beschränken und nachträglich die fehlende Energie wieder zurückzugewinnen. Diese Idee wird in einem progressiven Algorithmus umgesetzt, der anschaulich zu Beginn leicht unscharfe Beleuchtung berechnet, aber mit zunehmender Rechenzeit und durch Reduzierung der Dicke der VBLs zur tatsächlichen Lösung konvergiert. VBLs erhalten das Aussehen von Materialien und Medien besser, benötigen keine Behandlung von Sonderfällen zur Vermeidung von Artefakten und repräsentieren den aktuellen Stand der Technik für die Bildsynthese von partizipierenden Medien mit virtuellen Lichtquellen dar. Der Algorithmus wurde in:

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., and JAROSZ, W. [2012a]. Progressive virtual beam lights. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 31(4):1407–1413

veröffentlicht.

Beschleunigung der Sichtbarkeitsberechnung

Praktisch alle Verfahren, die die Lichtausbreitung in einer Szene berechnen, müssen während der Berechnung überprüfen, ob zwei Punkte in einer Szene zueinander sichtbar sind, oder ob der Lichttransport durch Flächen oder partizipierende Medien dazwischen behindert wird. Many-light-Methoden bilden hier keine Ausnahme denn die Sichtbarkeit wird sowohl bei der Konstruktion von Photonenpfaden, als auch bei der Berechnung der Beiträge von virtuellen Lichtquellen zu Oberflächenpunkten oder Punkten in Volumen überprüft. Um diese Sichtbarkeitsberechnung zu beschleunigen, wird in dieser Arbeit die *Rasterized Bounding Volume Hierarchy* (RBVH) Beschleunigungsstruktur für schnelles, approximatives Raytracing vorgestellt.

Die zugrundeliegende Idee ist, feintessellierte und detaillierte Geometrie durch Höhenfelder (d.h. Rasterbilder, die Höhendaten speichern) darzustellen. Während des Aufbaus der Datenstruktur werden Teile der Eingabegeometrie identifiziert, die sich gut durch ein einzelne, beliebig orientierte Höhenfelder darstellen lassen. Diese werden anschließend rasterisiert und in einer Hierarchie von Hüllkörpern angeordnet (ähnlich zu den klassischen Hüllkörperhierachien für die Beschleunigung von Raytracing). Um Strahlen zu verfolgen, wird die Hierarchie traversiert und nach Schnitten mit Höhenfeldern gesucht. In der Praxis ist dieser Ansatz schneller als traditionelle, polygonbasierte Beschleunigungsstrukturen. RBVHs bieten außerdem eine inhärente Oberflächenparameterisierung, die weitere Anwendungen, wie das Bemalen der Oberflächen in Echtzeit und sogenanntes Irradiance Caching ermöglicht. Die Details hierzu werden in Kapitel 8 vorgestellt, welches auf folgender Veröffentlichung beruht:

NOVÁK, J. and DACHSBACHER, C. [2012]. Rasterized bounding volume hierarchies. *Computer Graphics Forum (Proc. of Eurographics)*, 31(3):403–412.

Introduction

*A painter should begin every canvas with a wash of black, because
all things in nature are dark except where exposed by the light.*

— Leonardo da Vinci (1452–1519)

Ever since humans became self-aware, capturing and reproducing visual experience has become an important instrument of communication. The first attempts to share stories through imagery date back to prehistoric times. Starting with cave drawings, followed by perfecting the painting craftsmanship, through photography and motion picture, all the way to the contemporary, computer assisted image synthesis, we are on a long quest for delivering unparalleled visual experience; be it for capturing reality or rendering imagination.

When an artist wants to project the real world onto a canvas, he knows he needs to reproduce many intricate phenomena that light creates. From shadows, color-bleeding, caustics, to aerial perspectives and effects due to dispersion, he must precisely replicate the outcomes of interactions of light with the environment. With photography and cinematography, this became much easier and the need for delicate painting skills was alleviated through advances in technology. However, analog and digital light sensors can only capture what is real. Virtual worlds thus still remain the domain of artists, and only since recently, also of computers. To convert virtual scenes into believable images, we need to mimic physical processes that light undergoes. This is indeed a very challenging task requiring not only a full understanding of how light interacts and propagates through matter, but also the ability to simulate it accurately. With the recent advent of computers, this became more tractable and we witnessed tremendous improvements in computer generated imagery in the last three decades.

Yet there are still many problems that are not satisfyingly resolved. One of them is the propagation of light in complex scenes. On a molecular level, this may seem trivial as the transport is governed by a few well-known principles and laws. However, the exquisite intricacy comes with the vast number of atoms forming the world around us, and the vast number of photons that interact with them. And it is the extreme number of interactions, which we strive to simulate; ideally in a fraction of a second, that makes the problem extremely challenging. It seems reasonable to ask whether we need to simulate all of these interactions. Most likely not. The question is however ill-conditioned as many aspects of the human visual system, and perception in general, are still not sufficiently understood. As such, drawing lines between what should and what does not need to be simulated can easily leave us stuck in the uncanny valley[1].

---

[1] The term "uncanny valley" was coined in human aesthetics and robotics signifying visual look and movement that is almost, but not quite, human. The emotional response to such nearly human-like characters is often repulsive and sickening.

Realistic image synthesis is an area of computer graphics that is concerned with creating images indistinguishable from photographs. Any progress made in this field quickly finds its use in product design, film industry, architecture, but also in scientific visualization as efficient solutions to light transport can frequently be applied to similar problems, e.g. in medicine.

The primary goal of realistic image synthesis is to reproduce all visual phenomena due to the transport of light. Secondary—but often equally important—is the speed at which the images are generated. Achieving a good balance between these two requires i) a suitable mathematical framework for solving the integro-differential transport equations, ii) the right approximations that yield speed-ups but do not compromise quality, and iii) an efficient computational approach for realizing the solution on modern hardware. As these criteria do not always align with each other, finding the right flavor has been, and still is, the aim of research in photorealistic rendering.

In this thesis, we build upon the so-called *many-light* algorithms, which exhibit great versatility and cover a wide range of quality and performance applications. At the core of these algorithms is an important observation that the general light transport, which is costly to simulate as photons may undergo many interactions before reaching the sensor, can be formulated as a much simpler problem of evaluating only *direct* illumination from a collection of specifically generated virtual light sources. As such, most of these algorithms fulfill the first and the third criterion of choosing a suitable mathematical framework and an efficient computational approach. However, they often over-simplify the simulation and thus compromise the quality of resulting images.

In a nutshell, many-light algorithms represent all indirect illumination using a set of *virtual point lights* (VPLs). This brings several advantages, e.g. non-recursive evaluation, typically low amount of noise, and the chance to efficiently handle visibility computation. Nevertheless, there is one distinct drawback characteristic to VPL-based rendering: contracting all indirect lighting into a finite number of infinitesimal point lights introduces singularities, i.e. points where the integrand is undefined. Furthermore, since the radiant flux density due to a point light is inversely proportional to the squared distance to the source, surfaces and volumes that are close to virtual lights appear much brighter than other points in the scene. While the algorithm is still unbiased, the resulting images suffer from high, possibly unbounded variance, which surfaces as occasional bright "splotches" in the image. In order to remove these artifacts, most many-light algorithms bound the contribution of each virtual light. Unfortunately, this non-uniformly darkens the render and changes the appearance of materials.

Our goal is to improve the quality of many-light algorithms while preserving their favorable properties, i.e. high performance, scalability, and elegance. We focus on illumination that emphasizes the presence of virtual lights, and has so far been considered difficult to render with many-light techniques. In particular, we aim at scenes containing glossy materials and/or participating media. While these two phenomena may seem orthogonal, they present similar challenges to many-light rendering. They both emphasize the discrete nature of the solution, resulting in pronounced artifacts that plague the final image. They require more virtual lights to capture the light transport accurately than in the case of diffuse scenes with vacuum, and finally, they contribute immensely to the realism and visual complexity of the rendered image.

In the chapters to follow, we develop new integration techniques and introduce novel lighting primitives that allow many-light algorithms to handle scenes with glossy surfaces and participating media more efficiently. We outline our original contributions followed by an overview of the thesis in the next two sections.

1.1 Overview of Original Contributions

The work presented in this thesis builds upon many-light methods. We outline our contributions that improve the bias compensation for point-based many-light techniques, present new virtual light primitives for rendering in the presence of participating media, and describe a new acceleration structure for fast visibility tests.

1.1.1 Efficient Bias Compensation

Our first contribution targets the efficiency of bias compensation, which can be used to "correct" the bounded, biased solution of typical many-light algorithms. We start by analyzing the energy that is lost due to clamping the singularity. Then we reformulate the bias compensation to enable more efficient integration, and also propose several simplifying assumptions that have negligible impact on quality, but yield orders of magnitude faster solutions. For greater efficiency, we propose two techniques tailored for surfaces and volumes independently, both of which can be easily parallelized enabling further acceleration on modern graphics hardware.

1.1.2 Ray and Beam Lights

In participating media, scattering of light happens continuously along the direction of travel. We leverage this observation and propose a novel lighting primitive, the *virtual ray light* (VRL), which is formed by an entire line segment instead of just the vertex of a photon path. Distributing the energy along the line provably lowers the degree of the singularity. This in practice reduces the distracting "splotchy" artifacts—the major weakness of many-light algorithms—and thus improves the convergence, enhances temporal stability, and brings down the overall rendering cost. In order to avoid the singularity completely, we also propose to inflate the ray light into a *virtual beam light* (VBL) and further redistribute the energy over a cylindrical region with finite thickness. To ensure convergent results, we formulate the rendering algorithm progressively, ensuring that the error due to redistributing the energy diminishes in the limit.

1.1.3 Rasterized Bounding Volume Hierarchies

A frequent bottleneck of rendering algorithms is the computation of visibility, and many-light algorithms are no exception. To tackle this problem we present a *rasterized bounding volume hierarchy* (RBVH); an acceleration structure for fast, approximate ray tracing. We make the observation that decoupling visibility testing from the input, possibly finely tessellated, geometry can lead to higher tracing performance. In order to construct an RBVH, we identify surfaces that can be projected onto a grid without folding. Then we organize them into a hierarchy of bounding volumes and rasterize them into a texture atlas. We show that RBVHs can achieve higher tracing performance than traditional BVHs, and natively provide a storage for various surface signals. We also describe a hybrid approach that enhances the robustness of our approach. The benefits of the hierarchy are demonstrated on several applications such as real-time on-surface painting, illumination caching, or rendering of point clouds.

1.2   Organization of the Thesis

The thesis is organized into nine chapters. After this introduction, we lay out the fundamentals of radiative light transport in Chapter 2. We first briefly review the history of physics concerned with the propagation and nature of light. Then we point out important radiometric models and quantities, describe how light interacts with surfaces and media, and gradually derive fundamental equations governing the propagation of light.

In Chapter 3, we focus on many-light algorithms, describe the generation and lighting with virtual lights, and provide a detail overview of existing approaches categorized with respect to scalability and target application.

Efficient bias compensation is the main concern of Chapters 4 and 5. We analyze the impact of bounding the singularity and derive an efficient compensation technique for scenes with participating media. We also formulate a new hierarchical integration scheme that enables an efficient bias compensation between surfaces.

In Chapter 6, we present a new lighting primitive—the virtual ray light—and devise importance sampling strategies for minimizing the variance in scenes with isotropically as well as anisotropically scattering media.

In Chapter 7, we extend the ray light primitive into a beam light, removing the singularity completely. We also describe a progressive version of the algorithm that ensures asymptotically convergent results.

Finally, in Chapter 8, we present a new hierarchical data structure for accelerating approximate visibility queries. We compare its construction and performance to the current state of the art and demonstrate its versatility on several applications.

We conclude the thesis in Chapter 9 and provide additional derivations, analysis, and interpretations in Appendix A.

Fundamentals of Radiative Light Transport

There is a crack in everything.
That's how the light gets in.

— LEONARD COHEN (1934)

Visible light, as defined by the International Lighting Vocabulary [CIE 1987], is a radiation that is capable of directly causing a visual sensation. In this chaphter, we lay out the fundamentals of radiative transport and detail the most important principles that are employed in the chapters to follow. We start by briefly reviewing the history and evolution of ideas concerned with the nature of light. Then we outline four areas of optics, which study the phenomena at different levels of abstraction. We also introduce quantities for expressing the amount of radiant energy with respect to different measures, and explain principles that light obeys when interacting with surfaces and media. This knowledge will allow us to formulate transport equations that govern propagation of light and define the radiative equilibrium. Methods suitable for evaluating these equations are described at the end of the chapter.

2.1   Historical Introduction

The first systematic writings formalizing the propagation of light date back to antiquity, when Greek philosophers Empodocles and Euclid (cca. 300 BC) formulated the principles concerned with the *reflection* of light. Although they also speculated on effects due to light being refracted, it took until the seventeenth century when Snell experimentally discovered the *law of refraction*, and Fermat pronounced the *principle of least time*, stating that "nature always acts by the shortest course". Following these observations, Grimaldi and Hooke discovered the tendency of light to bend around obstacles, commonly denoted as *diffraction*. In 1666, Newton observed white light being split into component colors when passing through a refractive prism, known as the *dispersion* of light, and concluded that each spectral color has its specific index of refraction. Concurrently, Huygens was conducting experiments revealing *polarization* of light. These and several other findings demanded the light transport to be governed with a fundamental theory. To that end, two competing hypotheses were enunciated. In the following we briefly describe their historical evolution; more in-depth information can be found in Born and Wolf [1999].

The first hypothesis was pioneered by Huygens who improved and extended the existing *wave theory*, originally postulated by Aristotle. Physicists adhering to the wave theory presumed that the space is filled with elastic *aether*, where every point receiving luminous disturbance can act as a source of new disturbance, which is further propagated in spherical waves. While the wave theory respected most of the formerly established laws and principles, it had, at the time, difficulties reasoning about the cause of polarization. This was mostly due to the concurrent unfamiliarity with transverse waves;[1] and longitudinal waves,[2] which were already known from the propagation of sound, would not provide the desired answer.

The second, concurrent hypothesis, which laid the basis for many experiments undertaken by Newton, was based on the *corpuscular theory*. The origins of the corpuscular theory date back to antiquity, when Democritus speculated on all matter, including light, being formed by corpuscles (i.e. small particles) that travel along straight lines with finite velocity. The widely recognized authority of Newton and the lack of evidence for transverse waves made the corpuscular theory prevail over the wave theory for nearly a century.

The revival of the wave theory took place at the beginning of the nineteenth century, when Young clarified the effects of interference using transverse oscillations of the aether. Within the same theoretical framework, Fresnel postulated principles governing the intensity and polarization of light undergoing reflection and refraction. The decisive moment in favor of the wave theory, however, came with an experiment originally suggested by Arago. He proposed to compare the speed of light in media of different optical thickness (e.g. air and water); an experiment for which the theories yielded contradictory results. While Newton's corpuscular theory claimed light to travel faster in optically thicker media, the wave theory correctly predicted smaller velocity, and was indisputably confirmed by two independent realizations of Arago's experiment in 1850.

The advent of the elastic wave theory continued with equations and principles derived by Navier, Cauchy, Rayleigh, Doppler, and others. However, the supposed presence of the aether, imagined as an *elastic solid*, raised several questions: e.g. why is there no observable resistance of the aether against celestial bodies such as planets and stars? These concerns, stemming from all nature being reasoned about using strictly mechanical principles, were rendered irrelevant with the recognition of electromagnetic fields. The research of electromagnetism evolved almost independently of optics, but only up to the moment when Kohlrausch and Weber calculated the speed of electromagnetic waves be equal the speed of light. This led Maxwell, whose equations form the basis of the wave theory as we know it today, to assume light have the form of electromagnetic waves. In the upcoming years, the quest for explaining the nature of light using mechanical models was gradually abandoned and the complexity of electromagnetic field, which cannot be reduced to anything simpler, finally accepted.

The last significant turning point to set both theories complementary rather than exclusive came after Planck's discovery that electromagnetic energy is emitted in small *quanta*, also called photons. This helped Einstein in clarifying the *photoelectric effect* using the corpuscular theory. With the wave theory being able to explain diffraction and interference, and the corpuscular theory elucidating the photoelectric effect, physicists settled on a conclusion that light exhibits a duality; the *particle-wave* duality. This means that some phenomena can be satisfyingly explained only with one of the theories, but none of them can single-handedly describe the light transport in its entire complexity.

---

[1] Oscillation of matter in directions perpendicular to the direction of wave propagation
[2] Oscillation of matter parallel, i.e. back and worth, to the direction of wave propagation.

2.2   Models of Light

The brief historical introduction from the previous section indicates that the propagation of light and its interaction with matter can be studied at various levels of abstraction. It is not always necessary to consider all aspects of its transport; indeed, neglecting properties that are irrelevant to a given task can simplify the problem and concentrate the focus on the phenomena of interest. In the following paragraphs, we outline four different areas of optics introducing them from the simplest to the most complete. The classification is adopted from Saleh and Teich [2007].

Geometrical Optics.   The most straightforward theory of light transport, describing the phenomena at the level of geometric rules, is called *geometrical optics* (sometimes also called *ray optics*). In a sense, geometrical optics amounts to the corpuscular theory and its use is most appropriate when the wavelength of light is negligible compared to the size of objects encountered along its path. Saleh and Teich [2007] characterize geometrical optics as a limit of the wave theory when the wavelength is infinitesimally small, and Born and Wolf [1999] demonstrate that Maxwell's equations in such case obey geometric rules. Under these simplifications, light can be expressed in the form of light rays. The transport of these rays is governed by a number of postulates, such as that light travels along straight lines in homogeneous media, it obeys Fermat's principle of least time, and the amount of refraction is dependent on the relative speeds of light in the two media. The model allows light to be emitted, reflected, refracted, scattered in the media, or absorbed. These interactions are enough to simulate illumination effects that computer graphics is most concerned with, and thus, geometrical optics represents the most frequently used model in rendering nowadays.

Wave Optics.   A more complex theory, which contains the geometrical optics as a special case, describes the propagation of light in the form of waves. *Wave optics* assume light to be composed of a single scalar function of position and time, called the *wave function*, which obeys the wave equation [Saleh and Teich 2007]. Waves with wavelength between 10nm and 1mm are called optical waves and are further classified as ultraviolet (10nm to 390nm), visible (390nm to 760nm), or infrared (760nm to 1mm). All optical waves are subject to the principle of superposition, i.e. the sum of any two optical waves is also an optical wave with intensity attributed to interference. For example, let us consider two waves with the same intensity. In the case of constructive interference when the phases of both waves align perfectly, the total intensity equals four times the intensity of each of the original wave. In the case of destructive interference, the waves "cancel out" and the resulting intensity is zero. In addition to interference, wave optics can also simulate the effects of diffraction and dispersion of light.

Electro-Magnetic Optics.   Light, as all other electromagnetic radiation, propagates in the form of two coupled vector fields, i.e. the electric field and the magnetic field. *Electro-magnetic optics* studies the propagation of light with no simplifying assumptions, at the level of these two vector fields. At its core are the Maxwell's equations, which describe the divergence and rotation of each of the vector field with respect to the medium the radiation propagates through. The vectors of the electric and magnetic fields are perpendicular to each other and oscillate transversely to the optical axis (i.e. the direction of propagation). If the oscillations are within a plane, we call the light linearly polarized. In case when the vectors rotate around the optical axis, the light

is said to be either elliptically or circularly polarized, depending whether the projections of the vectors' endpoints onto a plane perpendicular to the optical axis follow an ellipse or a circle, respectively. Maxwell's equations provide a complete guide to solving the propagation of light; however, the level of detail is too high for most practical rendering scenarios and they are thus rarely used in realistic image synthesis.

Quantum Optics. Although electro-magnetic optics can precisely describe the propagation of light, the exchange of energy (e.g. due to the photoelectric effect) is left for interpretation to quantum theory. The fundamental principle of *quantum optics* is that light consists of photons that carry electromagnetic energy and momentum. They also exhibit a wave-like character that allows them to interfere and diffract. The electric field of a photon reacts with the charges present in atoms of matter. The photon is annihilated (absorbed) by the matter when it transfers its energy to an atom raising it to a higher energy level. In the opposite case, when the atom transits to a state of lower energy, a photon is emitted. In computer graphics, the laws of quantum optics can be used to simulate materials exhibiting luminescence and phosphorescence.

2.3 Domains, Measures, and Conventions

In this section, we introduce several important domains, related measures, and conventions to facilitate mathematical formulations delineated in the following text.

In many situations, it is convenient to treat surfaces and media independently. A medium is defined as all matter that fills a region of space. The interface between two adjacent (non-interpenetrating) media is then denoted as a surface. Let $\mathbb{R}^3$ be a three-dimensional space, $\partial\mathbb{V} \subset \mathbb{R}^3$ the union of all surfaces, $\mathbb{V} = \mathbb{R}^3 \setminus \partial\mathbb{V}$ the space between all surfaces referred to as media, and $\mathcal{S}^2 = \{\omega \in \mathbb{R}^3; |\omega| = 1\}$ the set of all directions. Then the Cartesian product $\mathbb{R}^3 \times \mathcal{S}^2$ represents the space of all possible rays, called the *ray space*. A ray with origin $\mathbf{x}$ and direction ω is denoted as $(\mathbf{x}, \omega)$. The ray space can also be represented as $\mathbb{R}^3 \times \mathbb{R}^3$ with a ray denoted as $(\mathbf{x} \to \mathbf{y})$.

Measures $\mathcal{A}(S)$, $\mathcal{V}(V)$, and $\sigma(\Omega)$ represent the *area measure* [m^2], *volume measure* [m^3], and *solid angle measure* [sr] of subsets $S \subseteq \partial\mathbb{V}$, $V \subseteq \mathbb{V}$, and $\Omega \subseteq \mathcal{S}^2$, respectively. In addition to those we define two projected measures. For a smooth surface S with normal $n(\mathbf{x})$, the *projected area measure* $\mathcal{A}^\perp(S)$ defined with respect to direction ω is the perpendicular projection of $\mathcal{A}(S)$ onto a plane perpendicular to ω:

$$\mathcal{A}^\perp(S) = \int_S |n(\mathbf{x}) \cdot \omega| \, d\mathcal{A}(\mathbf{x}). \tag{2.1}$$

Similarly, the *projected solid angle measure* $\sigma^\perp(\Omega)$ is defined as:

$$\sigma^\perp(\Omega) = \int_\Omega |n(\mathbf{x}) \cdot \omega| \, d\sigma(\omega), \tag{2.2}$$

and can be understood as projecting $\sigma(\Omega)$ onto a plane perpendicular to $n(\mathbf{x})$.

In the rest of the text, we use the following conventions: points $\in \mathbb{R}^3$ are written in bold and primarily denoted by $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$. Directions are written in italics and mostly denoted ω. For spatio-directional quantities varying with $\mathbf{x}$ and ω, we use arrows $\leftarrow$ and $\rightarrow$ to emphasize that

quantity $h(\mathbf{x} \leftarrow \omega)$ is *arriving* at $\mathbf{x}$ from direction ω, and quantity $h(\mathbf{x} \rightarrow \omega)$ is *leaving* $\mathbf{x}$ in direction ω, respectively. Additionally, we also use the convention that directions always originate at $\mathbf{x}$ and point towards the source or the destination of h, i.e. ω may face the opposite direction than in which h propagates.

2.4 Radiometric Quantities

In order to simulate the propagation of light, we need to quantify the energy associated with light transport. The field of physics that is concerned with measuring the amount of electromagnetic radiation with respect to one or more parameters is called radiometry. It defines several quantities and their physical meaning and units. In the following, we review the most relevant ones for realistic image synthesis.

Radiant Flux. *Radiant flux* (or *radiant power*), commonly denoted Φ, is the basic radiometric quantity expressing the amount of energy that passes through a region of space per unit time. The unit of radiant flux is the Watt [W] and can be further decomposed into SI units as joules per second $[\mathrm{J} \cdot \mathrm{s}^{-1}]$. Flux is often used to quantify the amount of energy emitted from light sources. One can imagine this as measuring all light that is originating from a light source and passing through an imaginary sphere that encloses it. It is worth noting that the size of the sphere does not matter since radiant flux quantifies the total amount of emitted (or received) energy, not its density.

Irradiance, Radiant Exitance, and Fluence. It is often convenient to express the amount of radiant flux per unit area. *Irradiance*, denoted E, is the quantity that measures the *area density* of flux incident on a surface. Similarly to the previous example, consider a sphere with the center aligned to an omni-directional point light. The area density of radiant flux incident on the inner surface of the sphere will depend on its radius r: $E = \Phi/4\pi r^2$. The quadratic term in the denominator will become one of our major concerns in later chapters.

In the general case, irradiance is a function of position $\mathbf{x}$, defined as the differential flux received by a differential area $\mathrm{d}\mathcal{A}(\mathbf{x})$ around $\mathbf{x}$:

$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}\mathcal{A}(\mathbf{x})}. \tag{2.3}$$

Given this relation, we can derive an integral that expresses the total flux Φ received by a surface with area $\mathcal{A}$:

$$\Phi = \int_{\mathcal{A}} E(\mathbf{x})\mathrm{d}\mathcal{A}(\mathbf{x}). \tag{2.4}$$

When flux is self-radiated from a surface, we talk about *radiant emittance*. If we also add the radiant density that the surface reflects, i.e. we combine the emission with the reflected light, we obtain *radiant exitance* (M), which is sometimes also denoted *radiosity* (B); the name was adopted from the heat transfer literature. The volumetric analog to irradiance, i.e. the radiant flux reaching a small region of space from all directions, is called *fluence* (F).

| **(a)** Radiant flux | **(b)** Irradiance | **(c)** Radiant intensity | **(d)** Field radiance | **(e)** Surface radiance |
| Φ | $E(\mathbf{x})$ | $I(\omega)$ | $L(\mathbf{x} \leftarrow \omega)$ | $L(\mathbf{x} \leftarrow \omega)$ |
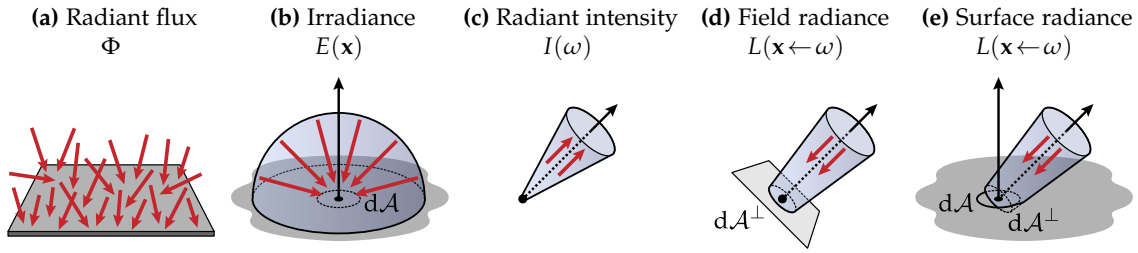
Figure 2.1: Fundamental radiometric quantities.

All of the aforementioned densities have the same unit $[\mathrm{W} \cdot \mathrm{m}^{-2}]$. In some publications they are also referred to as *intensity*; however, this often leads to confusion with radiant intensity, which is introduced next. One can always distinguish them by inspecting the units.

Radiant Intensity. We can also measure the *directional density* of flux. This quantity is called *radiant intensity* (I) with the units of watts per steradian $[\mathrm{W} \cdot \mathrm{sr}^{-1}]$. It is often used as an alternative way of specifying power of point light sources. Radiant intensity is defined as the differential flux emitted along directions confined to a differential solid angle $\mathrm{d}\sigma(\omega)$ around direction ω:

$$I(\omega) = \frac{\mathrm{d}\Phi(\omega)}{\mathrm{d}\sigma(\omega)}. \tag{2.5}$$

Computing the radiant flux emitted from a light source with known radiant intensity amounts to integrating I over the solid angle Ω, through which the light emits photons:

$$\Phi = \int_{\Omega} I(\omega)\mathrm{d}\sigma(\omega). \tag{2.6}$$

Radiance. Probably the most important quantity in rendering is *radiance*, commonly denoted L. It measures the light arriving to a small region from a small spread of directions. The term *field radiance* is often used to describe incident light at a general point $\mathbf{x}$ in three-dimensional space. Field radiance is defined as the radiant flux arriving through differential solid angle $\mathrm{d}\sigma(\omega)$ and passing through differential area $\mathrm{d}\mathcal{A}^{\perp}(\mathbf{x})$ on a hypothetical plane, which is perpendicular to ω and contains $\mathbf{x}$ (see Figure 2.1.d for an illustration):

$$L(\mathbf{x}, \omega) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \omega)}{\mathrm{d}\mathcal{A}^{\perp}(\mathbf{x})\mathrm{d}\sigma(\omega)}. \tag{2.7}$$

When $\mathbf{x}$ is on a surface with normal $n(\mathbf{x})$, it is often convenient to express the incoming light with respect to the differential area $\mathrm{d}\mathcal{A}$ on the surface, instead of $\mathrm{d}\mathcal{A}^{\perp}$ on the hypothetical plane. This amounts to defining $\mathrm{d}\mathcal{A}^{\perp}$ as $\mathrm{d}\mathcal{A}$ projected onto the hypothetical plane, i.e. $\mathrm{d}\mathcal{A}^{\perp} = |n(\mathbf{x}) \cdot \omega|\,\mathrm{d}\mathcal{A}$. The dot product captures how much the surface faces the incoming light and accounts for spreading of light rays at grazing angles. The equation for computing the *surface radiance* from

radiant flux reads:

$$L(\mathbf{x}, \omega) = \frac{\mathrm{d}^2 \Phi(\mathbf{x}, \omega)}{|n(\mathbf{x}) \cdot \omega| \, \mathrm{d}\mathcal{A}(\mathbf{x}) \mathrm{d}\sigma(\omega)}. \tag{2.8}$$

In some cases it is beneficial to express radiance in terms of the projected solid angle $\mathrm{d}\sigma^\perp(\omega) = |n(\mathbf{x}) \cdot \omega| \, \mathrm{d}\sigma(\omega)$:

$$L(\mathbf{x}, \omega) = \frac{\mathrm{d}^2 \Phi(\mathbf{x}, \omega)}{\mathrm{d}\mathcal{A}(\mathbf{x}) \mathrm{d}\sigma^\perp(\omega)}. \tag{2.9}$$

As long as it is obvious from the context, we will not distinguish between field and surface radiance, and always refer to the quantity as radiance in the rest of the text. The units of radiance are watts per steradian per square meter $[\mathrm{W} \cdot \mathrm{sr}^{-1}\mathrm{m}^{-2}]$. Analogously to the incident radiance, we can also define the outgoing (e.g. emitted or reflected) radiance.

In order to compute irradiance $E(\mathbf{x})$ and radiant exitance $M(\mathbf{x})$ at a surface point $\mathbf{x}$ radiating through the upper hemisphere $\mathcal{H}^2$, we need to integrate the incident and outgoing radiance, respectively:

$$E(\mathbf{x}) = \int_{\mathcal{H}^2_{n(\mathbf{x})}} L(\mathbf{x} \leftarrow \omega) \, (n(\mathbf{x}) \cdot \omega) \, \mathrm{d}\sigma(\omega), \tag{2.10}$$

$$M(\mathbf{x}) = \int_{\mathcal{H}^2_{n(\mathbf{x})}} L(\mathbf{x} \rightarrow \omega) \, (n(\mathbf{x}) \cdot \omega) \, \mathrm{d}\sigma(\omega). \tag{2.11}$$

Computing the radiant flux Φ received by a surface amounts to integrating the cosine-weighted incident radiance $L(\mathbf{x} \leftarrow \omega)$ over the area $\mathcal{A}$ of the surface and over the upper hemisphere $\mathcal{H}^2$ of directions about the surface normal $n(\mathbf{x})$:

$$\Phi = \int_{\mathcal{A}} \int_{\mathcal{H}^2_{n(\mathbf{x})}} L(\mathbf{x} \leftarrow \omega) \, (n(\mathbf{x}) \cdot \omega) \, \mathrm{d}\sigma(\omega) \mathrm{d}\mathcal{A}(\mathbf{x}). \tag{2.12}$$

An important property of radiance is that the sensitivity of human eyes and camera sensors is proportional to the incident radiance. Another characteristic of radiance is that it stays invariant along straight lines in vacuum. More precisely, assuming no participating medium or surface between points $\mathbf{x}$ and $\mathbf{y}$, the incident radiance at $\mathbf{x}$ can be expressed as the outgoing radiance from $\mathbf{y}$:

$$L(\mathbf{x} \leftarrow \omega) = L(\mathbf{y} \rightarrow -\omega), \tag{2.13}$$

where $\omega = \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{x} - \mathbf{y}\|}$. These two properties make radiance the quantity of choice for many rendering algorithms.

For brevity, in the rest of the text we will use shorthand notations $\mathrm{d}\mathbf{x}$ and $\mathrm{d}\omega$ for the differential area $\mathrm{d}\mathcal{A}(\mathbf{x})$ around point $\mathbf{x}$ and the differential solid angle $\mathrm{d}\sigma(\omega)$ around direction ω, respectively.

2.5   Interaction of Light with Surfaces

In the following sections, we detail local interactions of light with surfaces and media, respectively, gradually extending towards equations that govern the transport from a more global perspective.

2.5.1   Bidirectional Distribution Functions

When light reaches a surfaces, it can be reflected, refracted, or split and take part in both of these interactions. The directions and fractions of the reflected and refracted light are governed by the law of reflection, Snell's law of refraction, and Fresnel's equation. These derive the behavior of light from the Fermat's principle of least time with respect to optical parameters of media at both sides of the surface. Although the laws are easy to implement, applying them at a microscopic level would make rendering of macroscopic objects (i.e. objects that we can see with naked eyes) intractable. It is also worth noting that geometric variations below the scale of the wavelength can be ignored as they do not affect the wave. As such, rendering algorithms often require surfaces to be described at a mesoscopic level (approx. $1\,\mu m$ to $1\,mm$), incorporating the effects of microscopic scattering directly into the surface description. The description, referred to as the *bidirectional scattering distribution function* (BSDF) [Heckbert 1991], captures the average behavior of light in a small volumetric region around the point of interest, including the absorption and emission of light.

In general, the amount of light leaving point $\mathbf{x}$ in direction ω depends on the amount of radiance arriving at $\mathbf{x}$ from all directions. From Equation (2.10) we can derive the differential irradiance arriving from direction ω' at the differential area around $\mathbf{x}$:

$$\mathrm{d}E(\mathbf{x} \leftarrow \omega') = L(\mathbf{x} \leftarrow \omega') \left| n(\mathbf{x}) \cdot \omega' \right| \mathrm{d}\omega'. \tag{2.14}$$

If we now express the relative amount of this light that is scattered by the surface in direction ω, we obtain the definition of the BSDF $f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega')$:

$$f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') = \frac{\mathrm{d}L(\mathbf{x} \rightarrow \omega)}{\mathrm{d}E(\mathbf{x} \leftarrow \omega')} = \frac{\mathrm{d}L(\mathbf{x} \rightarrow \omega)}{L(\mathbf{x} \leftarrow \omega') \left| n(\mathbf{x}) \cdot \omega' \right| \mathrm{d}\omega'}. \tag{2.15}$$

In other words, the BSDF specifies how much of the differential irradiance arriving from a particular direction continues along another direction after interacting with the surface. Given this definition, we can relate the outgoing and incident radiance. The radiance leaving point $\mathbf{x}$ in direction w_o is defined as the product integral of the BSDF and the differential irradiance (i.e. the cosine-weighted incident radiance) over the sphere $\mathcal{S}^2$ of all possible directions:

$$
\begin{aligned}
L(\mathbf{x} \rightarrow \omega) &= \int_{\mathcal{S}^2} f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') \, \mathrm{d}E(\mathbf{x} \leftarrow \omega') \mathrm{d}\omega' \\
&= \int_{\mathcal{S}^2} f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') \, L(\mathbf{x} \leftarrow \omega') \left| n(\mathbf{x}) \cdot \omega' \right| \mathrm{d}\omega'.
\end{aligned}
\tag{2.16}
$$

It is sometimes convenient to consider only the reflected light, i.e. light that exits through the same hemisphere as it arrived. In such cases we talk about *bidirectional reflectance distribution function* (BRDF), and we ignore the light that radiates through the opposite hemisphere. Analogously, we can use *bidirectional transmittance distribution function* (BTDF) to describe the distribution of

(a) Diffuse BRDF   **(b)** Glossy BRDF   **(c)** Specular BRDF
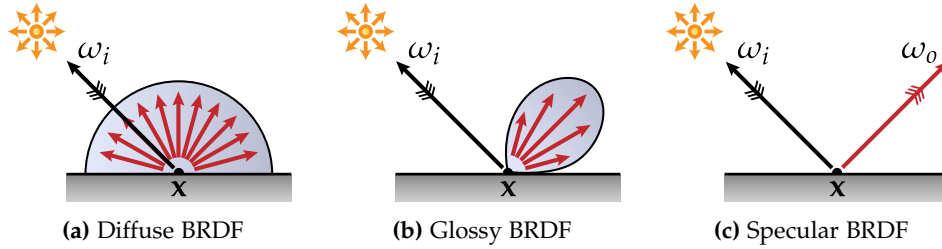
Figure 2.2: Examples of BRDFs: (a) diffuse BRDF reflects light uniformly into all directions, (b) glossy BRDFs concentrate the scattered light within lobes, (c) specular BRDFs, being Dirac delta functions, reflect all light into a single direction defined by the law of reflection or refraction.

only the transmitted light. These two complementary functions allow to model the reflection and transmission independently and were formalized by Nicodemus et al. [1977]. Combining two BRDFs and two BTDFs (a pair for each side of the surface) makes up for a complete BSDF.

All of the aforementioned distribution functions assume that light incident at some point leaves the surface at the same point. In cases when this assumption becomes inappropriate, and we still do not want to simulate the full light transport in the medium "below" the surface, we can use the *bidirectional surface scattering reflectance distribution function* (BSSRDF) [Nicodemus et al. 1977] $f_s(\mathbf{x} \rightarrow \omega, \mathbf{x}' \leftarrow \omega')$. Computing the outgoing radiance $L(\mathbf{x} \rightarrow \omega)$ then requires additional integration over the area around $\mathbf{x}$:

$$L(\mathbf{x} \rightarrow \omega) \quad = \quad \int_{\mathcal{A}} \int_{\mathcal{H}^2_{n(\mathbf{x})}} f_s(\mathbf{x} \rightarrow \omega, \mathbf{x}' \leftarrow \omega') \, L(\mathbf{x}' \leftarrow \omega') \, (n(\mathbf{x}') \cdot \omega') \, \mathrm{d}\omega' \, \mathrm{d}\mathbf{x}'. \qquad (2.17)$$

In practice, distribution functions are often wavelength dependent and can take arbitrary non-negative values. In general, it is however desired for the distribution function to be physically based and follow two additional constraints:

Energy conservation. The total amount of scattered flux must be less than or equal to the flux incident on the surface. This is captured by the following equation:

$$\int_{\mathcal{S}^2_{\mathbf{x}}} f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') \, |n(\mathbf{x}) \cdot \omega'| \, \mathrm{d}\omega' \leq 1. \qquad (2.18)$$

Distribution functions that do not conserve energy can prevent global illumination algorithms from finding the radiant equilibrium: by bouncing around the scene the light gets more and more amplified, in which case, the rendering algorithm may not converge.

Reciprocity. The *amount* of scattered light is invariant to the direction of light flow. More precisely: reversing the incident and outgoing directions does not affect the amount of light being scattered. This is nicely demonstrated by an experiment where a light source illuminates a reflective surface and the reflected light is measured by a sensor. If we exchange the source with the sensor then the reflected light measured by the sensor stays the same.[3] The principle of reciprocity, which is often called after its discoverer Hermann von Helmholtz [1924], ensures that the interaction of light with surfaces is symmetric, i.e. equal for both directions of travel.

---

[3] Assuming that the light source, the surface, and the sensor are sufficiently small.

Satisfying the Helmholtz reciprocity is important especially for algorithms that resolve the light transport from both directions, e.g. bidirectional path tracing, since they assume that constructing a path in the reverse direction yields the same result. Since reflection is known to be symmetric,[4] physically based BRDFs are reciprocal. However, this does not hold for general BSDFs. For instance, when light is refracted on an interface between two media with different refractive indices, the corresponding BTDF is not symmetric. In such cases, bidirectional estimators require special treatment of non-symmetrical scattering using adjoint BSDFs, see Veach [1996], Veach [1997], and Christensen [2003] for details.

Figure 2.2 shows three example BRDFs. Since this thesis primarily focuses on participating media, we refrain from discussing surface scattering in detail and refer the interested reader to standard literature, such as Dutré et al. [2006] and Pharr and Humphreys [2010].

2.5.2   Rendering Equation

We shall now present the *rendering equation* that governs light transport in environments consisting of light sources and surfaces. The rendering equation, as introduced by Kajiya [1986],[5] does not attempt to model all aspects of the light transport. Essentially, it is only an approximation of geometric optics in environments with no participating media. As such, effects due to polarization, diffraction, varying refractive index, interaction with media, etc. have to be handled with generalizations based on e.g. path integral techniques [Feynman and Hibbs 1965], or radiative transfer [Chandrasekhar 1960]. The latter we review in Section 2.7.

Hemispherical Formulation.   Motivated by the law of conservation of energy, the rendering equation defines the *steady-state* or *equilibrium* radiance leaving a point as the sum of the emitted $L_e(\mathbf{x}\rightarrow\omega)$ and reflected radiance $L_r(\mathbf{x}\rightarrow\omega)$:

$$L(\mathbf{x}\rightarrow\omega) = L_e(\mathbf{x}\rightarrow\omega) + L_r(\mathbf{x}\rightarrow\omega). \tag{2.19}$$

Given Equation (2.16), the reflected radiance can be expressed in terms of incident radiance $L(\mathbf{x}\leftarrow\omega')$:

$$L(\mathbf{x}\rightarrow\omega) = L_e(\mathbf{x}\rightarrow\omega) + \int_{\mathcal{S}^2} f_s(\omega\leftarrow\mathbf{x}\leftarrow\omega')\, L(\mathbf{x}\leftarrow\omega')\, \big|n(\mathbf{x})\cdot\omega'\big|\, d\omega'. \tag{2.20}$$

The above rendering equation formulates the equilibrium radiance locally (i.e. with respect to a single point) by distinguishing between incident and exitant quantities. If there is no participating medium, we can express the incident radiance at one point as the exitant radiance at another point using the ray casting function $r(\mathbf{x},\omega)$:

$$L(\mathbf{x}\leftarrow\omega) = L(r(\mathbf{x},\omega)\rightarrow-\omega), \tag{2.21}$$

---

[4]In general, not even reflection is symmetric. There are situations when optical paths are not reversible and light propagates along a different path when the direction of travel is inverted. Helmholtz noticed that this happens, for instance, to polarized light in the presence of external magnetic field. As these effects are beyond our interest, we shall assume reflection to be symmetric.

[5]Similarly to Chandrasekhar [1960], who uses the term *specific intensity*, Kajiya [1986] refers to the differential flux passing between two points using the term *intensity*. It is more common nowadays to denote this quantity *spectral radiance*, or simply *radiance* for brevity. This avoids confusion with *radiant intensity*.

where $r(\mathbf{x}, \omega)$ returns the surface point seen from $\mathbf{x}$ along ray $(\mathbf{x}, \omega)$. By combining the previous two equations we obtain:

$$L(\mathbf{x} \rightarrow \omega) = L_e(\mathbf{x} \rightarrow \omega) + \int_{\mathcal{S}^2} f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') \, L(r(\mathbf{x}, \omega') \rightarrow -\omega') \, |n(\mathbf{x}) \cdot \omega'| \, \mathrm{d}\omega'. \qquad (2.22)$$

The equation formulates light transport in terms of outgoing radiance only, and reveals the recursive nature of the rendering equation. This form of the rendering equation is also known as the *light transport equation*.

Area Formulation. An alternative to integrating the incident radiance over all possible directions is to integrate the contribution from all surface points. To achieve that we need to replace the differential solid angle with the differential surface area:

$$\mathrm{d}\sigma(\omega') = \frac{|n(\mathbf{y}) \cdot -\omega'|}{\|\mathbf{x} - \mathbf{y}\|^2} \mathrm{d}\mathcal{A}(\mathbf{y}), \qquad (2.23)$$

where $\mathbf{y} = r(\mathbf{x}, \omega')$. Equation (2.22) can be then written as:

$$L(\mathbf{x} \rightarrow \omega) = L_e(\mathbf{x} \rightarrow \omega) + \int_{\mathcal{A}} f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') V(\mathbf{x} \leftrightarrow \mathbf{y}) G(\mathbf{x} \leftrightarrow \mathbf{y}) L(\mathbf{y} \rightarrow -\omega') \mathrm{d}\mathbf{y}, \qquad (2.24)$$

where ω' is the direction towards $\mathbf{y}$, i.e. $\omega' = \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{x} - \mathbf{y}\|}$. The *visibility term V* is a binary function returning 1 if $\mathbf{x}$ and $\mathbf{y}$ are mutually visible, and 0 otherwise. The *geometry term G* accounts for the mutual orientation and distance between the two surface points by combining the original dot product from Equation (2.22) with the scaling factor from Equation (2.23):

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) = \frac{|n(\mathbf{x}) \cdot \omega'| \; |n(\mathbf{y}) \cdot -\omega'|}{\|\mathbf{x} - \mathbf{y}\|^2}. \qquad (2.25)$$

Equation (2.24) defines the light transport w.r.t three surface points (direction ω can be defined by a point $\mathbf{z}$ as $\omega = \frac{\mathbf{z} - \mathbf{x}}{\|\mathbf{x} - \mathbf{z}\|}$) and thus is sometimes referred to as the *three-point form* of the light transport equation.

2.6 Interaction of Light with Media

In the previous section, we introduced means to describe local interactions of light with interfaces between media with different refraction indices. In this section, we focus on the transport *through* a medium. If the medium has a constant index of refraction, the light propagates along straight lines. When the index of refraction is continuously changing, the light travels along curved trajectories. Additionally, the interaction of light with matter involves three main processes: emission, scattering, and absorption.

In the following, we first outline the physical processes that occur when light travels through media. Then we provide formalisms of characteristics that affect these processes, and briefly categorize media accordingly. Finally, we demonstrate how to sample the free path of a photon and how to evaluate transmittance between two points.
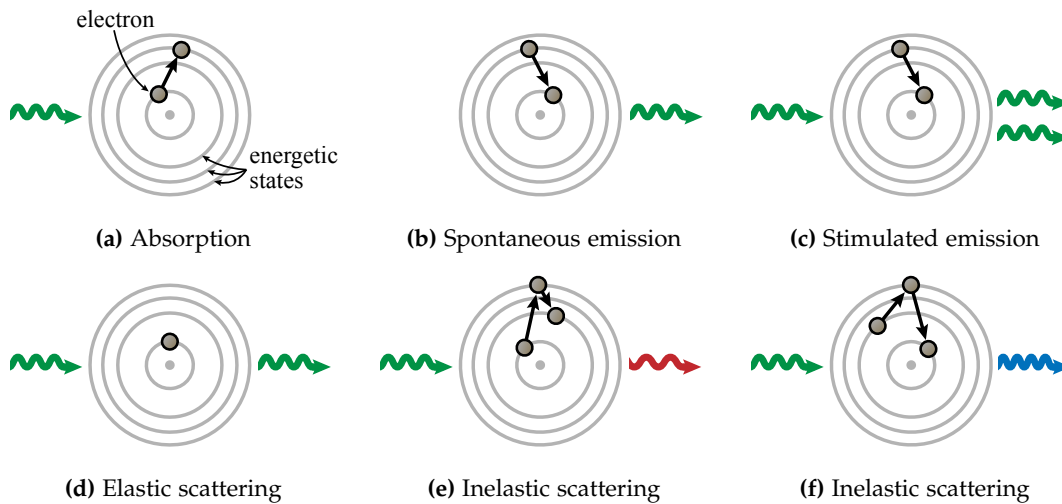
Figure 2.3: Examples of interactions between an atom and a photon. The wavy curves represent photons with different wavelengths. The shorter the wavelength the higher the energy and vice versa.

2.6.1 Absorption

Absorption is a process during which electromagnetic radiation is transformed into a different form of energy, e.g. heat or structural changes. If a photon with the right energy (i.e. frequency) collides with an atom or molecule, it can be absorbed. See Figure 2.3.a for an illustration. The energy of the photon must be sufficient to excite one or more electrons in the outer shell of the atom from their current state to a state with higher energy. If the energy is not sufficient, the photon will be (elastically) scattered or transmitted.

Absorption occurs only when the medium contains absorptive elements, such as pigments or dyes. The process of absorption is selective, meaning that pigments and dyes can usually absorb photons with particular wavelengths only, depending on the molecule's chromophore. The chromophore is thus responsible for the spectral distribution of light that survives the absorption [Baranoski and Krishnaswamy 2010]. In the context of wave propagation, the process of absorption is also called attenuation.

2.6.2 Emission

The process of exciting electrons into higher energy states can result also from collisions with another atoms. After some time (approx. 10 ns), these electrons spontaneously transit back to one of the lower energy states and the difference between the two states is emitted in the form of a photon (see Figure 2.3.b). The frequency of the emitted light is given by the Einstein equation, $f = |\Delta E| / h$, where h is the Planck's constant. In gaseous media, where the interactions between atoms are rather weak, the spectrum of the emitted light consists of a number of narrow bands. This is because the atoms have discrete energy levels allowing for only a finite number of energetic differences. However, when atoms move fast and interact strongly, their high velocities (relative to the observer) lead to the Doppler effect broadening the bands. If the bands overlap the spectrum of the emitted light becomes continuous.

(a) Planckian locus

(b) Examples of black body

(c) Hues of the Planckian locus
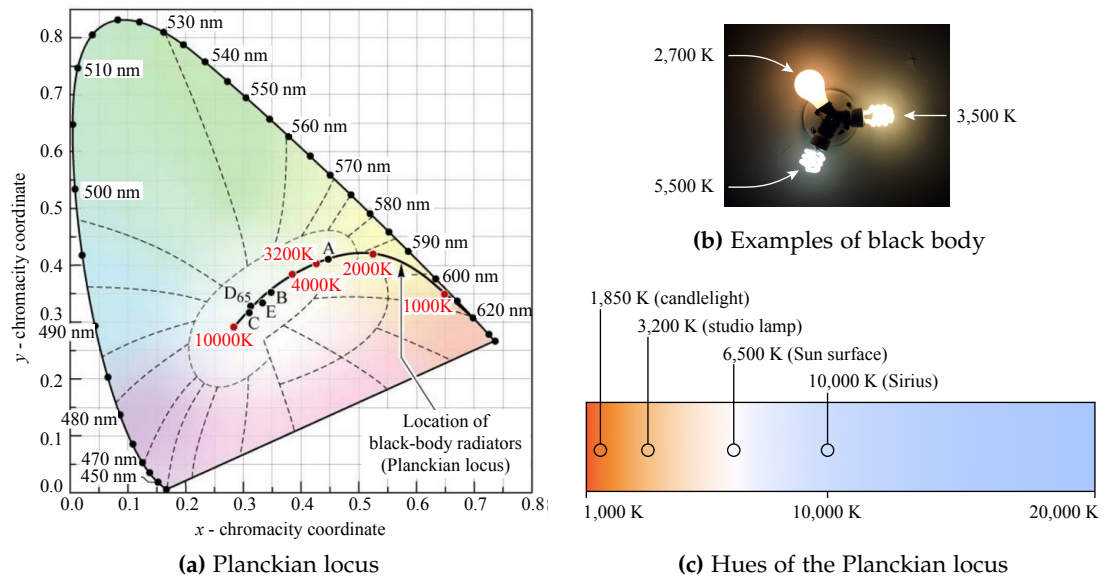
Figure 2.4: (a) Planckian locus in the CIE 1931 chromaticity diagram; image courtesy of E. F. Schubert, (b) shows examples of black body radiators, and (c) maps temperature of the black body to perceived hue.

An example of an ideal thermal radiator with a continuous spectrum is a *black body*, which absorbs all incident radiation, thermalizes it, and then emits uniformly with spectrum dependent solely on the temperature of the black body. In fact, all matter with temperature above absolute zero emits radiation. Most thermal radiators with room temperatures emit light in the infra-red region and only when the temperature reaches about 500°C the light becomes visible. Mapping between the temperature of a black body and the spectrum of emitted light is described by the Planckian locus, which defines the change of a color with respect to a temperature in a particular chromacity space. Figure 2.4.a shows the Planckian locus in the CIE 1931 chromacity diagram, and three examples of high-temperature radiators.

Temperature is not the only source of emission. Luminescent materials have the ability to temporarily store incident radiation by exciting their atoms to higher energy states. When these drop back to their ground states, the surplus energy is radiated in the form of light. *Fluorescence* is one example of almost instantaneous emission of energy obtained e.g. from ultraviolet light. Since the electrons return to the ground state via multiple transitions, the high energy of the ultraviolet light is split into several photons, some of which may have the frequency of visible light. In cases when the atoms stay excited for a longer time (on the order of milliseconds to hours) we talk about *phosphorescence*. The longer response is caused be exciting atoms to so-called "forbidden" meta-stable states, where the transition does not take the energetically most efficient path [Tipler and Mosca 2004, Section 31].

Emission can be also stimulated, as illustrated in Figure 2.3.c. If a photon collides with a molecule that is in an excited meta-stable state, the molecule transits into its ground state emitting a photon that has the same direction and phase as the colliding photon. The reaction can be chained leading to an amplification of the original light. Stimulated emission is the process utilized in lasers to produce high energy collimated beams.

2.6.3 Scattering

There are generally two types of scattering: elastic and inelastic. When a photon does not carry enough of energy to excite a molecule into a higher state, it is elastically scattered and no energy is transformed; see Figure 2.3.d. Examples of elastic scattering include Rayleigh and Lorenz-Mie scattering. Inelastic scattering occurs when a photon carries just the right amount of energy to excite the molecule, which, however, immediately after the excitation descends back to a lower energetic state. If the molecule gained energy, then we observe a shift in the spectra of the scattered light towards lower frequencies (see Figure 2.3.e). In the opposite case the molecule loses energy and the light becomes higher frequency (see Figure 2.3.f). Inelastic scattering shall not be confused with the effect of fluorescence, which requires a certain resonance time before the photon is emitted.

2.6.4 Formalization of Interactions

We now formalize the aforementioned characteristics of media in a collection of parameters, which can be used during light transport simulations.

Interaction Cross-sections. A *microscopic cross-section*, or simply cross-section, is an area measure of the likelihood that a particular interaction between two particles takes place. It refers to the effective area that a particle presents to another particle (e.g. a photon) for a particular interaction. The larger the effective area, the higher the chance that a photon will interact with the particle. The SI units of cross-section are square meters; however, physicist often express the value in barns ($1b = 10^{-28}m^2$) to deal with values in the range of tenths to few barns. Based on the type of interaction, we distinguish between the *absorption cross-section* σ_a and the *scattering cross-section* σ_s. The sum of the two then defines the *extinction cross-section* σ_t, which represents the total effective area of absorption and scattering.

Interaction Coefficients. Although being defined on a microscopic level, cross-sections are rather global parameters of the medium. The actual local probability of an interaction relates to the local *density* $\rho(\mathbf{x})$ $[m^{-3}]$ of the medium, i.e. the number of particles within a unit volume. Taking this into account yields *macroscopic cross-sections*, which are in computer graphics commonly referred to as *absorption coefficient* $\kappa_a(\mathbf{x})$, *scattering coefficient* $\kappa_s(\mathbf{x})$, and *extinction coefficient* $\kappa_t(\mathbf{x})$:

$$\kappa_a(\mathbf{x}) = \rho(\mathbf{x})\sigma_a, \tag{2.26}$$

$$\kappa_s(\mathbf{x}) = \rho(\mathbf{x})\sigma_s, \tag{2.27}$$

$$\kappa_t(\mathbf{x}) = \rho(\mathbf{x})\sigma_t = \kappa_a(\mathbf{x}) + \kappa_s(\mathbf{x}). \tag{2.28}$$

The coefficients define the probability that a photon traveling along a path of unit length will interact with a particle and take the corresponding interaction. The value of all of the three coefficients is wavelength dependent and defined with respect to a unit distance, which ranges from millimeters for solids and thick liquids to meters for gaseous media. The extinction coefficient is particularly useful when defining the optical thickness and transmittance between two points.

Albedo.   Similarly to surfaces, we can also compute the *single-scattering albedo α* (abbr. albedo) of the medium:

$$\alpha(\mathbf{x}) = \frac{\kappa_s(\mathbf{x})}{\kappa_t(\mathbf{x})}, \tag{2.29}$$

which describes how much the medium scatters light. If it equals 1, the medium is said to be perfectly scattering; if it equals 0, the medium does not scatter and only absorbs light.

Phase Function.   The angular distribution of the scattered light is modeled by the phase function $f_p(\omega' \rightarrow \omega)$, which is the volumetric analog to the BSDF. However, unlike the BSDF, the phase function is normalized:

$$\int_{\mathcal{S}^2} f_p(\omega' \rightarrow \omega) \mathrm{d}\omega = 1, \tag{2.30}$$

and serves as a density function defining the probability that a photon, arriving along direction ω' and scattering, continues in direction ω. As a convention, the direction of incidence points towards the scattering point, and the direction of exitance away from it. This is also different from the BSDF, where both directions face away from the surface point. In this text, we assume the phase function to be independent of the scattering location and use the shorthand notation $f_p(\omega' \rightarrow \omega)$ instead of the more general, but lengthy $f_p(\omega' \rightarrow \mathbf{x} \rightarrow \omega)$.

2.6.5   Examples of Interactions

In this section, we provide several examples of media with different scattering characteristics. We start with simple models and gradually proceed towards the more complex ones.

Isotropic Scattering.   When the phase function is isotropic, the medium scatters uniformly into all directions. Since the function must be normalized, there is only one isotropic phase function with a constant value of $1/4\pi$. The isotropic phase function is an analog to a Lambertian surface.

Anisotropic Scattering.   In order to model scattering by small particles in intergalactic dust clouds, Henyey and Greenstein [1941] devised a phase function, often abbreviated HG phase function, that defines the anisotropy using a single *asymmetry parameter g*. The function changes in one dimension only and can be thus parametrized in terms of only the angle θ between the incident and outgoing direction:

$$f_{p\mathrm{HG}}(\theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}}. \tag{2.31}$$

The asymmetry parameter $g \in \langle -1, 1 \rangle$ represents the mean cosine of the deviation from the direction of incidence. The higher the value the more light scatters in forward directions, negative values yield backward scattering. For $g = 0$ the phase function reduces to isotropic scattering. Thanks to the physical meaning of g, one can easily fit HG to an arbitrary phase function just by

(a) Isotropic PF

(b) Backward scattering
$(g = -0.4)$

(c) Forward scattering
$(g = 0.8)$
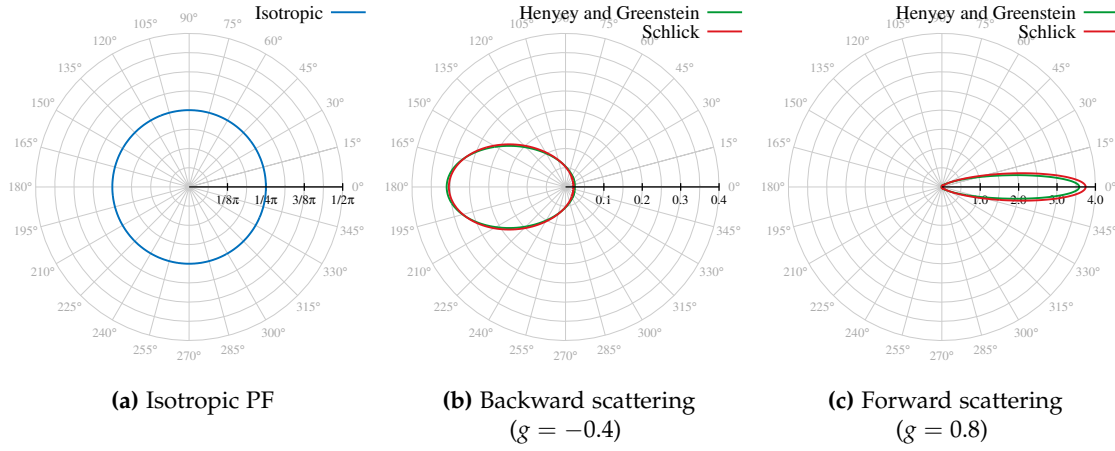
Figure 2.5: Polar plots with different phase functions: (a) shows the isotropic phase function, (b) and (c) demonstrate examples of backward and forward scattering, respectively, modeled using the Henyey and Greenstein phase function and Schlick simplified version with k obtained from g using the polynomial relation from Equation (2.34). Note that the plots have different linear scales to accommodate the full shape of the phase function.

integrating its product with the cosine of the angle between ω' and ω:

$$g = \int_{\mathcal{S}^2} f_p(\omega' \to \omega)(\omega' \cdot \omega)\mathrm{d}\omega. \tag{2.32}$$

By discretizing the above integral, one can also fit to measured data. The usability of HG ranges from gases and liquids to biological tissues.

In order to avoid the expensive fractional exponent, Blasi et al. [1993] developed a simplified version which is commonly known as the Schlick phase function:

$$f_{p\text{Schlick}}(\theta) = \frac{1}{4\pi} \frac{1 - k^2}{(1 - k\cos\theta)^2}. \tag{2.33}$$

The asymmetry parameter $k \in \langle -1, 1 \rangle$ has similar meaning to g. Pharr and Humphreys [2010] derived a polynomial relation between k and g:

$$k = 1.55g - 0.55g^3, \tag{2.34}$$

which allows for an approximate representation of HG using the Schlick phase function.

Although both of the previously described phase functions are rotationally symmetric and monotonous on the interval $(0, \pi)$, more complex shapes can be easily obtained using weighted sums of multiple asymmetric lobes.

Rayleigh Scattering. Rayleigh scattering, named after its discoverer Lord Rayleigh [1871], refers to interactions of light with particles and molecules that are much smaller than the wavelength of the light (up to one tenth of the wavelength). It can occur in transparent solids and liquids, but most often we experience it in gases in the atmosphere. Rayleigh scattering is wave-

(a) Rayleigh scattering cross-section

(b) Rayleigh phase function
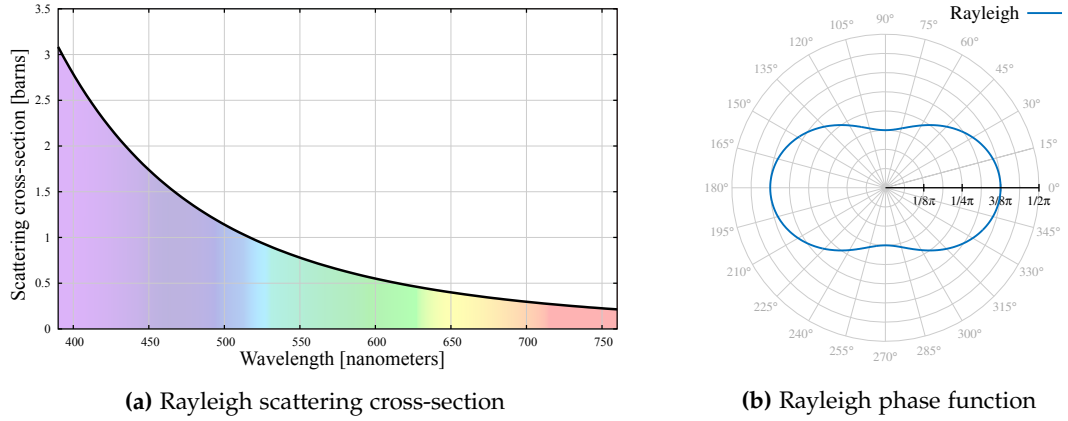
Figure 2.6: Rayleigh scattering is defined by its two characteristic properties: (a) the wavelength dependent scattering cross-section, and (b) the Rayleigh phase function.

length dependent with the scattering cross-section defined as:

$$\sigma_s(\lambda, d, \eta) = \frac{2\pi^5 d^6}{3\lambda^4} \left(\frac{\eta^2 - 1}{\eta^2 + 2} \right)^2, \tag{2.35}$$

where λ is the wavelength of the light, and d and η are the diameter and refractive index of particles, respectively. Since the probability of light being scattered varies as $1/\lambda^4$, higher frequencies towards the blue end of the spectrum are scattered more often than lower frequencies. This explains the blue color of the sky during the day, and reddish horizons during sunset, when most of the blue light is out-scattered before reaching the observer. The dependency of the scattering cross-section on the wavelength of visible light is depicted in Figure 2.6.a.

The phase function of Rayleigh scattering, shown in Figure 2.6.b, is defined as:

$$f_{p\text{Rayleigh}}(\theta) = \frac{3}{16\pi}(1 + \cos^2\theta), \tag{2.36}$$

and captures the fact that most of the light is scattered in forward and backward directions; scattering at right angles achieves only about half of the intensity.

Lorenz-Mie Scattering. When light interacts with particles that are comparable in size to its wavelength, we can no longer neglect the shape of the particle or the wave character of light. For such cases, e.g. a planar radiation arriving at a sphere, Ludvig Lorenz [1890] and Gustav Mie [1908] independently developed a solution to Maxwell's equations. The solution (sometimes also called Mie theory) is given as an infinite series that describes the amount and distribution of light after a collision with a homogeneous set of spheres. Similar solutions can be also derived for cylinders and ellipsoids. Figure 2.7 shows the phase function of Mie scattering for three differently sized spherical particles. Since the derivation and the resulting solutions are fairly involved, Nishita et al [1987] proposed to use an experimental approximation devised by Gibbson [1958] for two particularly interesting types of scattering in "hazy" and "murky"

(a) Mie, particles with $r = 1\mu m$ **(b)** Mie, particles with $r = 10\mu m$ **(c)** Mie, particles with $r = 100\mu m$

(d) Mie hazy and murky
(log plot)

(e) Mie hazy atmosphere
(linear plot)

(f) Mie murky atmosphere
(linear plot)
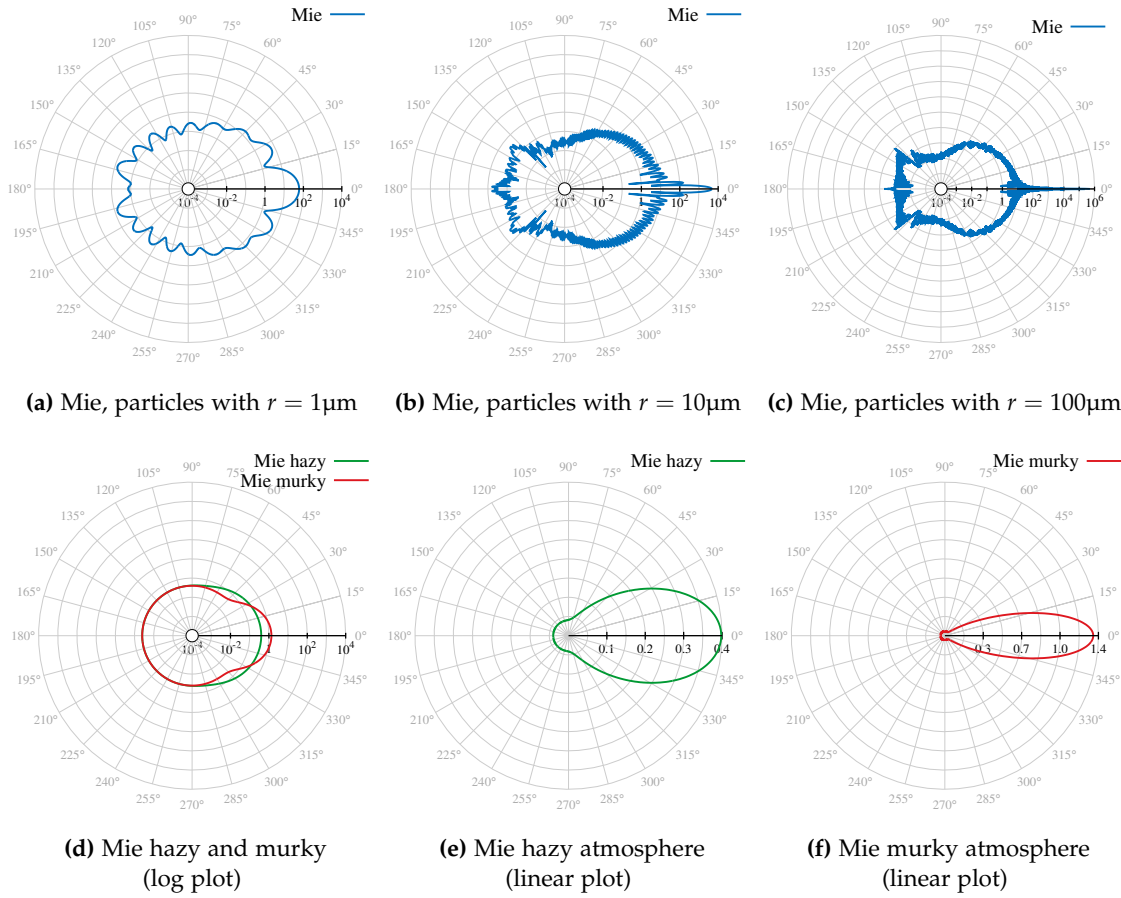
Figure 2.7: Phase functions of Mie scattering: (a), (b), and (c) show logarithmic polar plots of phase functions for Mie scattering in water particles with different radii r. The data was generate using MiePlot by Philip Laven. (d) demonstrates two approximations for "hazy" and "murky" atmospheres and (e) and (f) plot these approximations in linear scale to relate them better to phase functions from Figures 2.5 and 2.6.

atmospheres:

$$f_{p\text{MieHazy}}(\theta) \;=\; \frac{1}{4\pi}\left(\frac{1}{2} + \frac{9}{2} + \left(\frac{1+\cos\theta}{2}\right)^8\right), \tag{2.37}$$

$$f_{p\text{MieMurky}}(\theta) \;=\; \frac{1}{4\pi}\left(\frac{1}{2} + \frac{33}{2} + \left(\frac{1+\cos\theta}{2}\right)^{32}\right). \tag{2.38}$$

Lorenz-Mie scattering occurs mostly in gases and fluids that contain sufficiently large particles, e.g. water droplets in clouds or fat globules in milk. The solution can be also used in a reverse process to determine the size of the scattering particles [Graßmann and Peters 2004], or to compute the scattering properties of various forms of participating media [Frisvad et al. 2007].

Both Rayleigh and Mie are important when rendering realistic skies. While the first is mostly responsible for the aerial perspective and the color of a clear sky, the latter is necessary when simulating light transport in clouds. For a comprehensive explanation we refer the interested reader to Born and Wolf [1999] and to Bouthors [2008] for rendering skies in real-time.

2.6.6   Classification of Media.

We can classify participating media with respect to spatial and angular variation of the previously mentioned volumetric parameters.

Homogeneity and Heterogeneity.   If the parameters of a medium (e.g. the particle density or the cross-sections) are spatially invariant, the medium is *homogeneous*. As we will show in Section 2.6.7, certain components of the radiative transfer in homogeneous media can be expressed in a closed form. If the parameters change with position, the medium is said to be *inhomogeneous* or *heterogeneous*. There are two traditional ways of modeling the heterogeneity, one is based on procedural description and the other on storing the density in a discretized form using e.g. voxel grids.

Order of Scattering.   The density of the medium also affects the number of interactions that a photon will undergo on average. If we have a good *a-priori* intuition about the density and interaction coefficients, we can introduce relevant approximations and thereby significantly speed-up the light transport computation. As an example, consider a thin medium such as fog, haze, or champaign. Before reaching the camera, the light is not likely to scatter more than few times (if at all) and we can thus restrict the computation to *single-scattering* only. The simulation then reduces to direct illumination of the volume, which is far simpler than integrating the contribution of paths of all possible lengths. In the opposite extreme, when photons undergo hundreds and more interactions, and these are sufficiently localized, we can approximate the *multiple-scattering* component using the diffusion process. We detail the individual approaches in Section 2.8.

(An)Isotropy.   Media can be also be classified with regards to the phase function. If the phase function is independent of the position or the direction of incidence, light propagates "equally" in all directions and the medium is said to be *isotropic*. Note that the phase function itself can still be anisotropic, but its shape must be the same everywhere in the medium and for all incident directions. In some liquids and solids, e.g. those with crystalline structure, the phase function is truly four-dimensional depending also on the incident direction. Such media are classified as *anisotropic* or sometimes also referred to as *oriented*.

2.6.7   Transmittance

Now that we outlined the different types of interactions and participating media, we can focus on how light propagates through the volume. In vacuum, photons travel along straight lines unobstructed and radiance remains constant until the light reaches a surface. In media, on the contrary, the radiance changes due to absorption and scattering of photons.

Homogeneous Media

Consider a differential beam of light, shown in Figure 2.8, that propagates through a homogeneous medium along direction ω. We will denote the differential flux of photons entering the beam through a differential area $\mathrm{d}\mathcal{A}(\mathbf{x})$ along directions confined to $\mathrm{d}\sigma(\omega)$ as L_0. Let us further assume the medium to remain stationary and unchanged for the time of studying the light

$$\text{transmittance } T = L_d / L_0 = e^{-\kappa_t d}$$
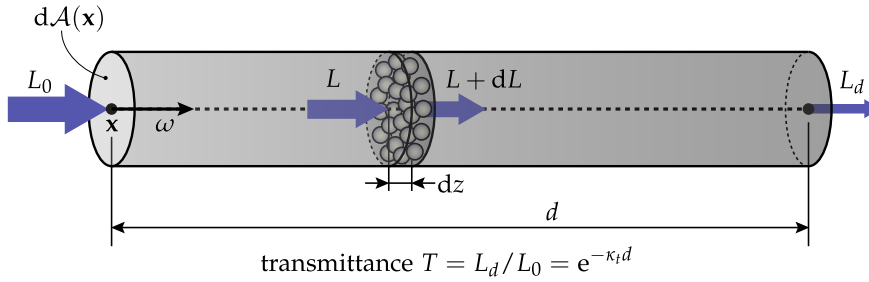
Figure 2.8: Illustration of the Beer-Lambert law that derives transmittance through a homogeneous slab with extinction coefficient κ_t.

transport within the beam. We shall now estimate L_d, i.e. the fraction of L_0 that reaches the end of the beam without interacting with the medium. As we move further away from **x**, some of the original photons will be absorbed or out-scattered. This process of extinction is quantified by the extinction coefficient κ_t, which expresses the fraction of light that is lost per unit distance. Consider an infinitesimal segment of the beam with length dz. Denoting the radiance entering the segment L, we can write the radiance exiting the segment as $L + dL$, where dL is the change due to interactions with the medium. From the definition of κ_t, the fraction of the incident light dL/L that makes it through the segment can be written as:

$$\frac{dL}{L} = -\kappa_t dz. \tag{2.39}$$

The reason for the negative sign on the right stems from the fact that κ_t expresses only the magnitude of photons that are lost, not that they should be subtracted from L; this needs to be accounted for explicitly using the minus sign. Integrating the differential equation yields:

$$\ln(L) = -\kappa_t z + c, \tag{2.40}$$

where the constant c vanishes when integrating over a finite length d of the beam:

$$\ln(L_d) - \ln(L_0) = -\kappa_t d, \tag{2.41}$$

$$\ln\left(\frac{L_d}{L_0}\right) = -\kappa_t d. \tag{2.42}$$

The last equation is referred to as the Beer-Lambert law,[6] which expresses the logarithmic dependence of *transmittance* (or *transmission*) $T = L_d / L_0$ on the product of the extinction coefficient and distance the light travels through the medium. The transmittance represents the fraction of the original radiance that travels along the entire length of the beam without interacting with the medium, and can be obtained by exponentiating Equation (2.42):

$$T = e^{-\kappa_t d}. \tag{2.43}$$

Equation (2.43) provides means to evaluate the transmittance in homogeneous media along a segment of length d. For a beam with zero length the transmittance is 1; as the length approaches infinity the transmittance drops to 0. The product $\kappa_t d$ is called *optical thickness* τ and measures

---

[6]In most literature the Beer-Lambert law is concerned only with absorption. Since we are interested in all light that is absorbed or out-scattered, we use the κ_t in our derivation.

the transparency of a slab of the medium with thickness d.

Since transmittance is multiplicative:

$$T(\mathbf{x}_0 \leftrightarrow \ldots \leftrightarrow \mathbf{x}_N) = \prod_{i=0}^{N-1} T(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}), \tag{2.44}$$

we can evaluate the total transmittance along a path consisting of N segments by first accumulating the optical thickness and then using a single instance of the (possibly expensive) exponential function:

$$T(\mathbf{x}_0 \leftrightarrow \ldots \leftrightarrow \mathbf{x}_N) = e^{-\kappa_t \sum_{i=0}^{N-1} d_i}. \tag{2.45}$$

Heterogeneous Media

In the case of a heterogeneous medium, the optical thickness is a function of two arbitrary points requiring the extinction coefficient to be integrated along the straight line connecting them:

$$\tau(\mathbf{x} \leftrightarrow \mathbf{y}) = \int_0^{\|\mathbf{x}-\mathbf{y}\|} \kappa_t(\mathbf{x} + s\omega)\mathrm{d}s, \tag{2.46}$$

where ω is the direction pointing from $\mathbf{x}$ towards $\mathbf{y}$. Naturally, transmittance becomes also a function of $\mathbf{x}$ and $\mathbf{y}$:

$$T(\mathbf{x} \leftrightarrow \mathbf{y}) = e^{\tau(\mathbf{x} \leftrightarrow \mathbf{y})}. \tag{2.47}$$

Generally, the optical thickness between points $\mathbf{x}$ and $\mathbf{y}$ cannot be expressed in a closed form and we need to evaluate the integral numerically, e.g. by one of the quadrature rules, which are in this context referred to as *ray marching* [Jensen and Christensen 1998, Perlin and Hoffert 1989]. The idea is to step along the ray connecting $\mathbf{x}$ and $\mathbf{y}$ breaking up the integration domain into a set of disjoint segments. The integral is thus replaced by a sum of sub-integrals that are approximated using one of the quadrature rules (e.g. rectangle, trapezoid, or Simpson's rule). Since the sum has to be finite in practice, there will be a certain error. If the marching is equidistant, the error may show up as aliasing (e.g. banding artifacts). When the length of segments is randomized (in the spirit of Monte Carlo integration), the aliasing is replaced by noise. Pauly et al. [2000] elaborate more on different ray marching schemes and propose to randomly jitter a sequence of equidistantly spaced marching steps to strike a good balance between banding and noise.

The biggest drawback of using quadratures to evaluate transmittance is that the results are biased. This is because:

$$E[e^{\tau(\mathbf{x} \leftrightarrow \mathbf{y})}] \neq e^{E[\tau(\mathbf{x} \leftrightarrow \mathbf{y})]}, \tag{2.48}$$

where $E[X]$ is the expected value of X. In other words, the stochastic error of estimating the transmittance will not average to zero and the estimation does not converge to the correct value.

In order to estimate transmittance in an unbiased way, we need to employ a form of rejection sampling called Woodcock tracking. This technique was introduced for sampling the free path, which we elaborate on first in the next section and defer the complete description of unbiased transmittance computation to the end of Section 2.6.8.

2.6.8 Free Path Sampling

The trajectory of a photon traveling through a medium consists of several segments connecting the individual interaction events. If the medium has a constant index of refraction, the segments are linear, but in general, they can be curved if the refractive index changes continuously. The length of each segment, which is commonly denoted the *free path*, depends on the local particle density and extinction cross-section of the medium.

An alternative to characterizing media using an extinction coefficient is to specify the *mean free path*. In a homogeneous unbounded medium, the mean free path d_m can be found analytically by computing the expected value of a probability density function (PDF) p, which is based on the transmittance along a semi-infite beam (see Equation (2.51) below):

$$
\begin{aligned}
d_m &= E\left[p(d)\right] \\
&= \int_0^\infty \kappa_t e^{-\kappa_t t} t \, \mathrm{d}t \\
&= \frac{1}{\kappa_t}.
\end{aligned}
\tag{2.49}
$$

The fact that d_m is the reciprocal of κ_t should be no surprise. While the extinction coefficient defines the fraction of energy that becomes extinct per unit length, the mean free path defines the length traveled by a "unit" of flux (e.g. a photon) before extinction.

In the following paragraphs, we detail a few techniques developed over the years to sample the length of the free path d of a photon traveling from $\mathbf{x}$ in direction ω. Although we can possibly sample the distance from an arbitrary distribution and still obtain unbiased results, we will limit the description to techniques that sample the free path from a probability density function that is proportional to transmittance, i.e.:

$$
p(\mathbf{x}, \omega, d) \propto T(\mathbf{x} \leftrightarrow \mathbf{x} + d\omega).
\tag{2.50}
$$

Homogeneous Media

In homogeneous media, the extinction coefficient is constant and the probability density function for sampling the free path can be expressed in a closed form. For an infinite medium, the free path is unbounded and the PDF equals to the transmittance normalized to integrate to unity:

$$
\begin{aligned}
p(d) &= \frac{T(d)}{\int_0^\infty T(t)\,\mathrm{d}t} \\
&= \frac{e^{-\kappa_t d}}{\int_0^\infty e^{-\kappa_t t}\,\mathrm{d}t} \\
&= \kappa_t e^{-\kappa_t d}.
\end{aligned}
\tag{2.51}
$$

Integrating $p(d)$ yields the cumulative distribution function (CDF) $P(d)$:

$$
\begin{aligned}
P(d) &= \int_0^d p(t)\,\mathrm{d}t \\
&= 1 - e^{\kappa_t d}.
\end{aligned}
\tag{2.52}
$$

Sampling the length of the free path amounts to inversion sampling of $P(d)$, i.e. choosing a uniform random number $\xi \in \langle 0, 1 \rangle$ and solving the equation $P(d) = \xi$ for d:

$$
\begin{aligned}
\xi &= 1 - e^{\kappa_t d} \\
d &= -\frac{\ln(1 - \xi)}{\kappa_t}.
\end{aligned}
\tag{2.53}
$$

For ξ close to 1, the length of the free path will approach infinity.

It may happen that the sampled length of the free path is behind the nearest surface at distance d_{max}. In that case, the photon will first interact with the surface and we thus need to clamp d to d_{max} and set the PDF of the sample to $1 - P(d_{max})$.

In some situations, it is desired to restrict the free path sampling only to distances that are shorter than d_{max}, e.g. when sampling the in-scattered light along a given finite camera ray. This can be achieved by changing the normalization factor in $p(d)$ to an integral with the upper bound set to the maximum distance d_{max}:

$$
\begin{aligned}
p(d) &= \frac{e^{-\kappa_t d}}{\int_0^{d_{max}} e^{-\kappa_t t} dt} \\
&= \frac{\kappa_t e^{-\kappa_t d}}{1 - e^{-\kappa_t d_{max}}}.
\end{aligned}
\tag{2.54}
$$

The CDF evaluates to:

$$
P(d) = \frac{e^{\kappa_t d_{max}} - e^{\kappa_t(d_{max} - d)}}{e^{\kappa_t d_{max}} - 1},
\tag{2.55}
$$

and can be readily inverted to sample d using a random number ξ analytically:

$$
d = d_{max} - \frac{\ln(\xi - (\xi - 1)e^{\kappa_t d_{max}})}{\kappa_t}.
\tag{2.56}
$$

Originally, we introduced the aforementioned equations in the context of sampling the free path in homogeneous media. However, closer inspection reveals that only the extinction coefficient needs to be constant to obtain the above closed form solutions. Indeed, the absorption and scattering coefficients can vary spatially but as long as they add up to the same value, we can still sample the free path analytically. Nevertheless, as the set of media with spatially varying albedo but constant extinction coefficient is rather theoretical, the practical applicability of the aforementioned analytic distributions reduces to homogeneous media.

Heterogeneous Media

The length of the free path in heterogeneous media can be, at the cost of some bias, resolved by ray marching. Except for the termination criterion, the technique is similar to a ray marching estimate of transmittance. We first draw a random number ξ, and then march along the ray $(\mathbf{x}, \omega)$ accumulating the optical thickness until the transmittance reaches $1 - \xi$ (or simply ξ, since ξ is uniformly distributed), or the ray hits the nearest surface. As stated previously, using quadrature methods to estimate exponentiated integrals does not converge to the correct result. In the next paragraph, we describe a technique that is unbiased and can be used for both, sampling the free path as well as evaluating the transmittance.

(a) Heterogeneous medium   **(b)** Woodcock tracking   **(c)** Rejection sampling
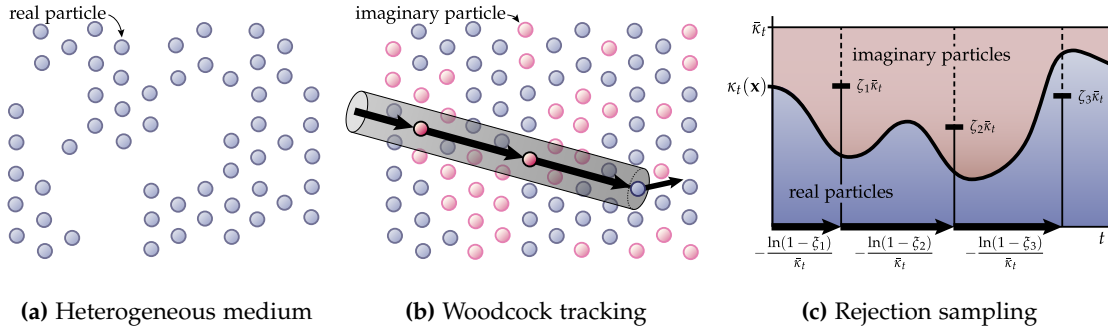
Figure 2.9: The idea behind Woodcock tracking is to fill the heterogeneous volume (a) with imaginary particles, so that the combined majorant extinction coefficient $\bar{\kappa}_t$ is constant throughout the volume. The algorithm then samples tentative distances (b) and probabilistically decides whether the collision occurred with a real or an imaginary particle, based on the relative extinction coefficient of real particles (c). In this example, we show three iterations where the beam of light first collides with two imaginary particles, and then bounces off a real particle. The plot in (c) shows the real extinction coefficient $\kappa_t(\mathbf{x})$ along the beam.

Woodcock Tracking.   Another iterative approach for sampling the free path is called *Woodcock tracking* [Woodcock et al. 1965], sometimes also known as *delta-tracking*, *pseudo scattering*, or more generally *distance sampling*. The technique, originally developed for neutron simulation in reactors with arbitrary shape, was introduced to the field of computer graphics by Raab et al. [2008]. Woodcock tracking is a form of rejection sampling, where samples are discarded when the interaction does not occur with "real" particles. The idea is to fill the volume with "imaginary" particles, so that the combined particle density, denoted $\bar{\rho}$, is the same everywhere; generally set to the maximum density of real particles. Formally, the imaginary particles have the following properties (marked with ′):

$$\sigma'_a = 0, \tag{2.57}$$

$$\sigma'_t = \sigma'_s = \sigma_t, \tag{2.58}$$

$$f'_p(\theta) = \delta(\theta), \tag{2.59}$$

$$\rho'(\mathbf{x}) = \bar{\rho} - \rho(\mathbf{x}). \tag{2.60}$$

In words, all imaginary particles have albedo equal to 1 and a perfectly forward-scattering phase function. All light interacting with imaginary particles is scattered in the forward direction; therefore, imaginary particles have no impact on the light transport. The *majorant extinction coefficient* $\bar{\kappa}_t$ of the combined medium reads:

$$\begin{aligned}
\bar{\kappa}_t &= \sigma_t \rho(\mathbf{x}) + \sigma'_t \rho'(\mathbf{x}) \\
&= \sigma_t \left(\rho(\mathbf{x}) + \bar{\rho} - \rho(\mathbf{x}) \right) \\
&= \sigma_t \bar{\rho} \tag{2.61}
\end{aligned}$$

and is constant throughout the volume.

We can thus analytically sample a *tentative* free path length d_t:

$$d_t = -\frac{\ln(1 - \xi)}{\bar{\kappa}_t}, \tag{2.62}$$

WoodcockTracking(x, w, tMax)

```
 1  extMax ← majorantExtinction()
 2  t ← 0
 3  do :
 4      t ← t − log(1 − rand())/extMax
 5      p ← x + t * w
 6      if t >= tMax :
 7          break
 8      ext ← extinctionAt(p)
 9  while rand() > ext/extMax
10  return t
```

Figure 2.10: Pseudocode of Woodcock tracking.

and then probabilistically decide whether the interaction at $\mathbf{x}_t = \mathbf{x} + d_t\omega$ occurred with a real or an imaginary particle. For this, we draw a random number ζ. If ζ is smaller or equal to the relative extinction coefficient of real particles, i.e. $\zeta \leq \kappa_t(\mathbf{x}_t)/\bar{\kappa}_t$, the collision involves a real particle and is accepted. In the opposite case, the light interacts with an imaginary particle, its flux and direction remain unchanged, and we repeat the process of sampling tentative events until an interaction with a real particle is found, or the neareast surface is hit. Figure 2.9 shows an example of the tracking with two imaginary and one real interactions. Pseudocode of Woodcock tracing is provided in Figure 2.10.

Imaginary particles have no impact on the light transport; they are used just to reason about the majorant extinction coefficient and to provide an intuition for rejecting interactions and continuing the tracking further. As for any kind of rejection sampling, the number of rejected samples depends on how closely the envelope, here the majorant extinction coefficient, matches the sampled distribution. In media with large differences in density, the relative concentration of real particles will be locally low, and the tracking is likely to take many steps requiring a lot of random numbers. This can easily become the bottleneck of the simulation. Some techniques thus divide the volume into regions with independently computed majorant extinction coefficients [Szirmay-Kalos et al. 2011, Yue et al. 2010]. This decreases the overall number of imaginary particles and avoids excessive rejection of samples.

Woodcock Multi-Tracking.   In order to evaluate transmittance in an unbiased manner, Jarosz et al. [2011b] designed an estimator that uses multiple instances of Woodcock tracking. The transmittance along a finite ray is then estimated by counting the relative number of instances that interact with the first real particle behind the end-point of the ray. The authors show that the expected value of such estimator equals to the transmittance and the estimation is unbiased.

One disadvantage of Woodcock multi-tracking is that estimating the transmittance using N free-path samples requires asymptotically $N \times M$ random numbers, where M is inversely proportional to the *collision sampling efficiency* [Leppänen 2010] of the majorant. This requirement can be lifted by correlating the tracking of individual free paths using the same sequence of random numbers ξ, and by using the same random ζ in the termination criterion of a single tracking.

2.7   Radiative Transfer Equation

With the various local parameters covered in the previous sections, we can now formulate a more complete model of light propagation in the presence of participating media. Let us consider an infinitesimal cylindrical volume $\mathrm{d}\mathcal{V} = \mathrm{d}\mathcal{A}\mathrm{d}z$, where $\mathrm{d}\mathcal{A}$ and $\mathrm{d}z$ are the differential cross-section and the differential length of the cylinder, respectively. The change in flux flowing between the two sides of the cylinder along directions confined to the differential solid angle $\mathrm{d}\omega$, where ω is the axis of the cylinder, will be subject to four physical processes.

Absorption and Out-Scattering.   The change in radiance due to light being absorbed (see Figure 2.11.a) and transformed into other form of energy can be expressed as:

$$\mathrm{d}L(\mathbf{x}\rightarrow\omega) = -\kappa_a(\mathbf{x})L(\mathbf{x}\rightarrow\omega)\mathrm{d}z. \tag{2.63}$$

The other loss comes from light being scattered out (see Figure 2.11.b) by the medium:

$$\mathrm{d}L(\mathbf{x}\rightarrow\omega) = -\kappa_s(\mathbf{x})L(\mathbf{x}\rightarrow\omega)\mathrm{d}z. \tag{2.64}$$

Adding these two together yields the total loss of radiance per $\mathrm{d}z$:

$$\mathrm{d}L(\mathbf{x}\rightarrow\omega) = -\kappa_t(\mathbf{x})L(\mathbf{x}\rightarrow\omega)\mathrm{d}z. \tag{2.65}$$

Emission and In-Scattering.   Radiance gained due to emission within the differential cylinder (see Figure 2.11.c) can be expressed as:

$$\mathrm{d}L(\mathbf{x}\rightarrow\omega) = \kappa_a(\mathbf{x})L_e(\mathbf{x}\rightarrow\omega)\mathrm{d}z. \tag{2.66}$$

The opposite process of out-scattering is when light from all incident directions scatters within the differential cylinder along direction ω (see Figure 2.11.d). The radiance gain due to in-scattering reads:

$$\mathrm{d}L(\mathbf{x}\rightarrow\omega) = \kappa_s(\mathbf{x})L_i(\mathbf{x}\rightarrow\omega)\mathrm{d}z, \tag{2.67}$$

where $L_i(\mathbf{x}\rightarrow\omega)$ is given by a product integral of the incident radiance and the phase function $f_p(\omega\leftarrow-\omega')$ over all directions:

$$L_i(\mathbf{x}\rightarrow\omega) = \int_{\mathcal{S}^2} f_p(\omega\leftarrow-\omega')L(\mathbf{x}\leftarrow\omega')\mathrm{d}\omega'. \tag{2.68}$$

The self-emission and the in-scattered light are often expressed as a single *source term* $J(\mathbf{x}\rightarrow\omega)$:

$$
\begin{aligned}
J(\mathbf{x}\rightarrow\omega) &= (1-\alpha(\mathbf{x}))L_e(\mathbf{x}\rightarrow\omega) + \alpha(\mathbf{x})L_i(\mathbf{x}\rightarrow\omega) \\
&= (1-\alpha(\mathbf{x}))L_e(\mathbf{x}\rightarrow\omega) + \alpha(\mathbf{x})\int_{\mathcal{S}^2} f_p(\omega\leftarrow-\omega')L(\mathbf{x}\leftarrow\omega')\mathrm{d}\omega',
\end{aligned} \tag{2.69}
$$

where $\alpha(\mathbf{x})$ is the albedo at $\mathbf{x}$.

In 1960, Chandrasekhar [1960] defined and concatenated all the previous terms in the funda-
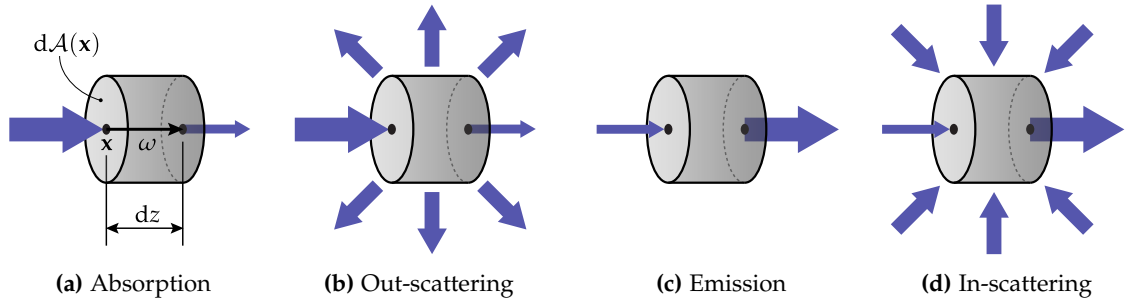
Figure 2.11: Four processes that define the change in radiance between two sides of a differential cylinder.

mental *radiative transfer equation* (RTE) that governs the variation of radiance in a medium:[7]

$$dL(\mathbf{x} \to \omega) = \underbrace{\kappa_t(\mathbf{x}) J(\mathbf{x} \to \omega) dz}_{\text{gains}}$$

$$- \underbrace{\kappa_t(\mathbf{x}) L(\mathbf{x} \to \omega) dz}_{\text{losses}}, \tag{2.70}$$

which can be further split to emphasize the individual components:

$$dL(\mathbf{x} \to \omega) = \underbrace{\kappa_a(\mathbf{x}) L_e(\mathbf{x} \to \omega) dz}_{\text{emission}}$$

$$+ \underbrace{\kappa_s(\mathbf{x}) L_i(\mathbf{x} \to \omega) dz}_{\text{in-scattering}}$$

$$- \underbrace{\kappa_a(\mathbf{x}) L(\mathbf{x} \to \omega) dz}_{\text{absorption}}$$

$$- \underbrace{\kappa_s(\mathbf{x}) L(\mathbf{x} \to \omega) dz}_{\text{out-scattering}}. \tag{2.71}$$

The above differential equation can be written in several alternative forms, e.g.:

$$(\omega \cdot \nabla) L(\mathbf{x} \to \omega) = \kappa_a(\mathbf{x}) L_e(\mathbf{x} \to \omega)$$

$$+ \kappa_s(\mathbf{x}) L_i(\mathbf{x} \to \omega)$$

$$- \kappa_a(\mathbf{x}) L(\mathbf{x} \to \omega)$$

$$- \kappa_s(\mathbf{x}) L(\mathbf{x} \to \omega), \tag{2.72}$$

or in an integral form by integrating both sides of Equation (2.71) along a semi-infinite ray $(\mathbf{x}, \omega)$, i.e. expressing the gains that are subject to extinction as:

$$L(\mathbf{x} \leftarrow \omega) = \int_0^\infty T(\mathbf{x} \leftrightarrow \mathbf{x}_t) \kappa_t(\mathbf{x}_t) J(\mathbf{x}_t \to -\omega) dt$$

$$= \int_0^\infty T(\mathbf{x} \leftrightarrow \mathbf{x}_t) \big[\kappa_a(\mathbf{x}_t) L_e(\mathbf{x}_t \to -\omega) + \kappa_s(\mathbf{x}_t) L_i(\mathbf{x}_t \to -\omega) \big] dt, \tag{2.73}$$

---

[7]In the original book, Chandrasekhar uses the term *specific intensity* for what is today commonly denoted *spectral radiance*.

where $\mathbf{x}_t = \mathbf{x} + t\omega$. This integral form of the RTE can be writen concisely as:

$$L(\mathbf{x} \leftarrow \omega) = \int_0^\infty T(\mathbf{x} \leftrightarrow \mathbf{x}_t) L(\mathbf{x} \rightarrow -\omega) \mathrm{d}t. \tag{2.74}$$

2.7.1 Boundary Conditions

Since most of the scenes that we deal with in computer graphics contain surfaces, we need to express the radiance along a *finite* ray and with respect to boundary conditions. The boundary condition for Equation (2.74) is given by Equation (2.20), i.e. for a ray $(\mathbf{x}, \omega)$ hitting a surface point $\mathbf{x}_b$, the boundary condition is equal to the radiance scattered from $\mathbf{x}_b$ in direction $-\omega$:

$$L(\mathbf{x}_b \rightarrow -\omega) = L_e(\mathbf{x}_b \rightarrow -\omega) + \int_{\mathcal{S}^2} f_s(-\omega \leftarrow \mathbf{x}_b \leftarrow \omega') \, L(\mathbf{x}_b \leftarrow \omega') \, (n(\mathbf{x}_b) \cdot \omega') \, \mathrm{d}\omega'. \tag{2.75}$$

Adding the boundary condition to Equation (2.74) expresses the total radiance reaching $\mathbf{x}$ from direction ω as:

$$L(\mathbf{x} \leftarrow \omega) = \int_0^b T(\mathbf{x} \leftrightarrow \mathbf{x}_t) L(\mathbf{x}_t \rightarrow -\omega) \mathrm{d}t \; + \; T(\mathbf{x} \leftrightarrow \mathbf{x}_b) L(\mathbf{x}_b \rightarrow -\omega), \tag{2.76}$$

see Figure 2.12 for an illustration of some of the terms. By expanding the exitant radiance functions and writing them in terms of emmission and incident radiance, we can emphasize the recursive nature of the RTE:

$$
\begin{aligned}
L(\mathbf{x} \leftarrow \omega) = \; & \underbrace{\int_0^b T(\mathbf{x} \leftrightarrow \mathbf{x}_t) \kappa_a(\mathbf{x}_t) L_e(\mathbf{x}_t \rightarrow -\omega) \mathrm{d}t}_{\text{accumulated volume emission}} \\
& + \underbrace{\int_0^b T(\mathbf{x} \leftrightarrow \mathbf{x}_t) \kappa_s(\mathbf{x}_t) \int_{\mathcal{S}^2} f_p(-\omega \leftarrow -\omega') L(\mathbf{x}_t \leftarrow \omega') \mathrm{d}\omega' \mathrm{d}t}_{\text{accumulated volume in-scattering}} \\
& + \underbrace{T(\mathbf{x} \leftrightarrow \mathbf{x}_b) L_e(\mathbf{x}_b \rightarrow -\omega)}_{\text{attenuated surface emission}} \\
& + \underbrace{T(\mathbf{x} \leftrightarrow \mathbf{x}_b) \int_{\mathcal{S}^2} f_s(-\omega \leftarrow \mathbf{x}_b \leftarrow \omega') \, L(\mathbf{x}_b \leftarrow \omega') \, (n(\mathbf{x}_b) \cdot \omega') \, \mathrm{d}\omega'}_{\text{attenuated surface scattering}}
\end{aligned}
\tag{2.77}
$$

2.7.2 Sensors

When rendering a scene, we do not necessarily need to find the equilibrium radiance at all points of the scene, but rather in a small subset that is visible to the camera. The camera can be represented by a two-dimensional sensor that measures the radiance incident from a specific cone of directions. Our goal is to integrate the incident radiance over the sensor's pixel area $\mathcal{A}_p$ and over the aperture Ω of the camera. The measurement I can be written as:

$$I = \int_{\mathcal{A}_p} \int_{\Omega} W(\mathbf{x} \leftarrow \omega) L(\mathbf{x} \leftarrow \omega) \mathrm{d}\omega \mathrm{d}\mathbf{x}, \tag{2.78}$$
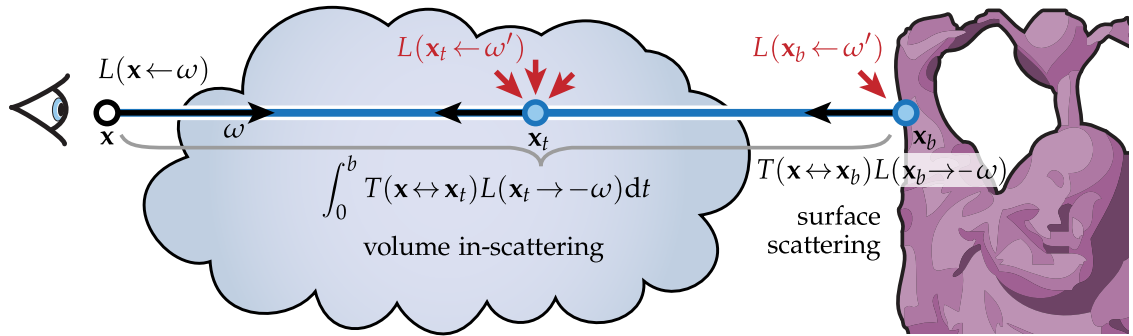
Figure 2.12: Illustration of some of the terms from Equations (2.76) and (2.77).

where $W(\mathbf{x} \leftarrow \omega)$ is the *sensor response function*. At this point, we can simply plug in the recursive formulation of equilibrium radiance from Equation (2.77). One disadvantage of this formulation is the recursive "explosion" of the integrals. In the next section, we introduce an alternative, yet equivalent formulation that avoids this problem.

2.7.3 Path Integral Formulation

Path integrals were pioneered by Feynman and Hibbs [1965]. Veach [1997] adapted the path integral framework to light transport between surfaces and Pauly et al. [2000] extended this work to participating media. The main idea is to hide the recursion and express the light transport as an integral over all possible paths.

The first step is to extend the three-point formulation of the LTE (cf. Equation (2.24)) to account for volumes. Alternatively, one can also derive it directly from the RTE (cf. Equation (2.76)). We skip the derivation here for brevity and refer interested readers to Appendix A.1. The resulting three-point formulation of the RTE is a Fredholm integral equation of the second kind:

$$L(\mathbf{x} \to \omega) = \hat{L}_e(\mathbf{x} \to \omega) + \int_{\mathbb{R}^3} f(\omega \leftarrow \mathbf{x} \leftarrow \omega') \hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}) T(\mathbf{x} \leftrightarrow \mathbf{y}) V(\mathbf{x} \leftrightarrow \mathbf{y}) L(\mathbf{y} \to \mathbf{x}) \mathrm{d}\mu(\mathbf{y}). \quad (2.79)$$

The above equation unifies the notation across surface and volumetric points by using several generalized terms; the *generalized emission* $\hat{L}_e$:

$$\hat{L}_e(\mathbf{x} \to \omega) = \begin{cases} L_e(\mathbf{x} \to \omega) & \text{if } \mathbf{x} \in \partial \mathbb{V} \\ \kappa_a(\mathbf{x}) L_e(\mathbf{x} \to \omega) & \text{if } \mathbf{x} \in \mathbb{V}, \end{cases} \quad (2.80)$$

the *generalized scattering function* f:

$$f(\mathbf{x} \leftarrow \mathbf{y} \leftarrow \mathbf{z}) = \begin{cases} f_s(\mathbf{x} \leftarrow \mathbf{y} \leftarrow \mathbf{z}) & \text{if } \mathbf{y} \in \partial \mathbb{V} \\ f_p(\mathbf{x} \leftarrow \mathbf{y} \leftarrow \mathbf{z}) \kappa_s(\mathbf{y}) & \text{if } \mathbf{y} \in \mathbb{V}, \end{cases} \quad (2.81)$$

the *generalized geometry term* $\hat{G}$:

$$\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}) \quad = \quad \frac{D_{\mathbf{x}}(\mathbf{y}) D_{\mathbf{y}}(\mathbf{x})}{\|\mathbf{x} - \mathbf{y}\|^2}, \tag{2.82}$$

$$D_{\mathbf{x}}(\mathbf{y}) \quad = \quad \begin{cases} n(\mathbf{x}) \cdot \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{x} - \mathbf{y}\|} & \text{if } \mathbf{x} \in \partial \mathbb{V} \\ 1 & \text{if } \mathbf{x} \in \mathbb{V}, \end{cases} \tag{2.83}$$

and the *generalized differential measure* $\mathrm{d}\mu(\mathbf{x})$:

$$\mathrm{d}\mu(\mathbf{x}) = \begin{cases} \mathrm{d}\mathcal{A}(\mathbf{x}) & \text{if } \mathbf{x} \in \partial \mathbb{V} \\ \mathrm{d}\mathcal{V}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathbb{V}. \end{cases} \tag{2.84}$$

Path Space.   Let $\mathcal{P}_k$ be the set of all paths of length k:

$$\mathcal{P}_k = \{\bar{x} = \mathbf{x}_0 \mathbf{x}_1 \ldots \mathbf{x}_k; \ \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^3\}, \tag{2.85}$$

where $1 \le k < \infty$ and $\bar{x}$ is a single path of length k. For any subset $D \subset \mathcal{P}_k$ we define the *product measure* as:

$$\mu_k(D) = \int_D \mathrm{d}\mu(\mathbf{x}_0) \cdots \mathrm{d}\mu(\mathbf{x}_k), \tag{2.86}$$

and the *differential path measure* as:

$$\mathrm{d}\mu(\bar{x}) = \mathrm{d}\mu(\mathbf{x}_0) \cdots \mathrm{d}\mu(\mathbf{x}_k). \tag{2.87}$$

Given the definition of $\mathcal{P}_k$, we can define the *path space* $\mathcal{P}$, i.e. the space of all paths of all possible lengths, as:

$$\mathcal{P} = \bigcup_{k=1}^{\infty} \mathcal{P}_k. \tag{2.88}$$

The measurement I from Equation (2.78) can now be expressed as a single integral over the path space:

$$I = \int_{\mathcal{P}} f_j(\bar{x}) \mathrm{d}\mu(\bar{x}), \tag{2.89}$$

where the integrand f_j is known as the *measurement contribution function*.

We shall now define f_j. By combining Equation (2.76) and Equation (2.78), and then recursively expanding the exitant radiance (see the full derivation in Appendix A.2), we can identify the terms whose product is known as the *throughput* $\mathcal{T}$ of the path. The throughput consists of the generalized scattering functions (one for each intermediate vertex of the path) and the generalized geometry, transmittance, and visibility terms (one for each path segment). See Figure 2.13 for an illustration. For a path connecting vertices $\mathbf{x}_j$ and $\mathbf{x}_k$, the throughput reads:

$$\mathcal{T}(\mathbf{x}_j \ldots \mathbf{x}_k) = \prod_{i=j+1}^{k-1} \left[f(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) \hat{G}(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) V(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) \right]$$
$$\hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k) T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k) V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k). \tag{2.90}$$
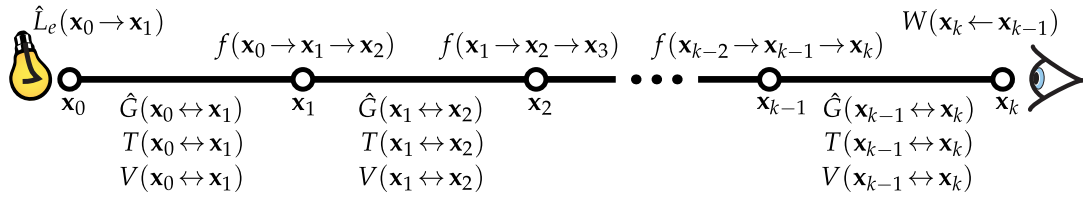
Figure 2.13: Illustration of terms defining the measurement contribution function from Equation (2.91). The product of the f, $\hat{G}$, T, and V terms is commonly referred to as the *path throughput*.

With $\mathbf{x}_0$ and $\mathbf{x}_k$ being points on a light source and on the sensor, respectively, the measurement contribution function can be succinctly written as:

$$I_j(\bar{x}) = \hat{L}_e(\mathbf{x}_0 \to \mathbf{x}_1)\mathcal{T}(\mathbf{x}_0 \ldots \mathbf{x}_k)W(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1}), \tag{2.91}$$

2.7.4 Operator Notation

The transport of light, after it is emitted and until it gets absorbed, consists of two alternating processes: scattering on surfaces and in media, and propagation (along straight lines) between individual interactions. It is often convenient to express these steps using an operator notation. The three operators defined in the following paragraphs are linear relations, that act on input functions producing new instances of these functions. We adopt the surface operators introduced by Arvo et al. [1994] and extend them to account for interactions with participating media. Although the operators can be applied to arbitrary functions, which are defined on ray space, we skip the formalism and define them directly on the radiance function.

Local Scattering Operator. The *local scattering operator*[8] **K** takes the *incident* radiance $L(\mathbf{x} \leftarrow \omega')$ at a point $\mathbf{x}$ and transforms it into the *exitant* radiance $L(\mathbf{x} \to \omega)$:

$$(\mathbf{K}L)(\mathbf{x} \to \omega) = \int_{\mathcal{S}^2} f(\omega \leftarrow \mathbf{x} \leftarrow \omega')L(\mathbf{x} \leftarrow \omega')\mathrm{d}\omega'^{\perp}. \tag{2.92}$$

Note that for $\mathbf{x} \in \partial\mathbb{V}$, the projected solid angle $\mathrm{d}\omega'^{\perp} = (n(\mathbf{x}) \cdot \omega')\,\mathrm{d}\omega'$ accounts for the orientation of the surface w.r.t the direction of incidence ω'. For $\mathbf{x} \in \mathbb{V}$, the dot product vanishes, i.e. $\mathrm{d}\omega'^{\perp} = \mathrm{d}\omega'$. **K** is a *local* operator in the sense that the relation of the exitant to the incident radiance is defined for each point $\mathbf{x} \in (\mathbb{V} \cup \partial\mathbb{V})$ in isolation, independent of all other points.

Propagation Operator. The transport of radiance between two interactions is described by the *propagation operator*[9] **G**. Given a point $\mathbf{x}$ and a direction ω, the propagation operator expresses the *incident* radiance $L(\mathbf{x} \leftarrow \omega)$ as the integral of the *exitant* (emitted or in-scattered) radiance along the ray $(\mathbf{x}, \omega)$, plus the attenuated *exitant* radiance emitted or scattered from the nearest

---

[8]While Arvo et al. use the term *local reflection operator*, we prefer the more general term *local scattering operator* as it better represents media scattering and surface transmission.

[9]Similarly to Veach [1997], we prefer the name *propagation operator* instead of the term *field radiance operator* used by Arvo et al. [1994]

surface point $\mathbf{x}_b$:

$$(\mathbf{G}L)(\mathbf{x}\leftarrow\omega) = \int_0^b T(\mathbf{x}\leftrightarrow\mathbf{x}_t)L(\mathbf{x}_t\rightarrow\omega)\mathrm{d}t + T(\mathbf{x}\leftrightarrow\mathbf{x}_b)L(\mathbf{x}_b\rightarrow\omega), \qquad (2.93)$$

where $\mathbf{x}_t = \mathbf{x} + t\omega$, $\mathbf{x}_b = \mathbf{x} + b\omega$, and b is the distance to the nearest surface visible along ω. More informally, $\mathbf{G}$ takes all radiance exitant from volumes and surfaces and transports it to $\mathbf{x}$ attenuating it by the corresponding transmittance.

Transport Operator.   The *transport operator* $\mathbf{T}$ combines $\mathbf{G}$ and $\mathbf{K}$ transforming one exitant radiance function $L(\mathbf{x}\rightarrow\omega)$ into another exitant radiance $(\mathbf{T}L)(\mathbf{x}\rightarrow\omega)$ as a result of a single light bounce:

$$(\mathbf{T}L)(\mathbf{x}\rightarrow\omega) = (\mathbf{K}\mathbf{G}L)(\mathbf{x}\rightarrow\omega) \qquad (2.94)$$

Whenever it does not introduce ambiguity, we will drop the functional parameters and denote the exitant radiance as L.

2.7.5   Neumann Series

With the help of transport operators we can express Equation (2.79) in terms of exitant radiance and write it succinctly as:

$$L = L_e + \mathbf{T}L, \qquad (2.95)$$

stating that the exitant equilibrium radiance is a sum of the emitted and transported radiance. Equation (2.95) clearly reveals the recursive nature of solving the radiative transfer. By recursively expanding the equation:

$$L = L_e + \mathbf{T}L_e + \mathbf{T}^2 L_e + \dots, \qquad (2.96)$$

we obtain the Neumann series:

$$L = \sum_{k=0}^{\infty} \mathbf{T}^k L_e. \qquad (2.97)$$

In order to assess the convergence of the Neumann series, it is necessary to define the *operator norm*, which for a linear operator $\mathbf{S}$ reads:

$$\|\mathbf{S}\| = \sup_{\|h\|\leq 1} \|\mathbf{S}h\|. \qquad (2.98)$$

For scenes containing only reflective surfaces, Arvo [1995] proves that $\|\mathbf{T}\| \leq 1$ if all BRDFs are symetrical and energy conserving. Additionally, if there is at least one BRDF with albedo < 1, then $\|\mathbf{T}\| < 1$, which is a sufficient condition for the Neumann series to converge to a finite value. This formalizes our discussion from Section 2.5.1 on the need for energy conserving distribution functions to ensure convergence of global illumination algorithms.

In his thesis, Veach points out that if the scene contains refractive surfaces, then $\|\mathbf{K}\|$, and thus

also $\|\mathbf{T}\|$, are no longer bounded by 1. Precisely:

$$\|\mathbf{K}\| < \frac{\eta_{max}^2}{\eta_{min}^2}, \tag{2.99}$$

where η_{max}^2 and η_{min}^2 are the maximum and minimum refractive indices present in the scene, respectively. This stems from the fact that radiance can increase when refracted into an optically thicker medium. Fortunately, the condition $\|\mathbf{T}\| < 1$ is not strictly necessary to achieve convergence. As shown in [Veach 1997], a weaker condition $\|\mathbf{T}^k\| < 1$ for some $k \geq 1$ is sufficient, i.e. as long as the radiance function decreases after a certain number of bounces, the Neumann series converges.

2.8 Evaluation of the Radiative Transfer Equation

In order to compute a realistic image of a virtual scene, we need to find the equilibrium radiance at points that are visible to the camera. Optimally, we would like to obtain a closed-form functional representation defined over all such points; however, this is possible only in very simple and mostly uninteresting scenes. We thus need to resort to numerical recipes. Since the number of (infinitesimal) points that are visible to the camera is infinite, Dutré et al. [2006] formulate the problem as finding the average equilibrium radiance over a number of point sets. The computation thus simplifies to finding the radiance only for a few representatives whose combined contribution defines the radiance for all points in the set. The exact definition of the point sets is dependent on the rendering algorithm. In the following, we briefly outline some of the techniques developed for solving the light transport and the radiative transfer equation in particular. For an overview of other rendering algorithms please refer to Dutré et al. [2006] and Pharr and Humphreys [2010].

2.8.1 Analytic Integration

The lack of computational power in the early years of computer graphics forced researchers to derive analytic solutions to the integro-differential transport equations. Many of those were adopted from the heat transfer literature, which provides more than 300 *form-factors*[10] that describe the transport of energy between surfaces of various geometric primitives, ranging from infinitesimal elements to shapes such cones or cylinders. A collection of these form-factors can be found in Howel et al. [2010]. We will now focus on analytic techniques that take into account participating medium.

In certain situations, some terms in the RTE (cf. Equation (2.77)) can be expressed in a closed form. For instance, both emission terms yield analytic expressions in homogeneous media. In some cases, we can integrate even the in-scattered radiance analytically. This term of the RTE is often referred to as the *airlight integral*.

Lecocq et al. [2000] proposed an angular reformulation of the airlight integral for point light sources. Instead of integrating along the length of the ray, they integrate along the angle between the two lines that connect the point light to the start and end points of the ray. By expanding the

---

[10]In the heat transfer literature, these are often called *configuration factors* or *geometric factors*.

reformulated integral into a Taylor series they obtain an analytic approximation of the airlight integral. Sun et al. [2005] further simplified the angular formulation and presented a semi-analytic model, which does not require expansion into a Taylor series, but relies on precomputing and tabulating the integrand. Both of the previous approaches are limited to homogeneous media with isotropic phase functions and point lights. Pegoraro and Parker [2009] derived the first fully analytic solution to the airlight integral, and further extended the technique to handle anisotropic phase functions and light sources with axially symmetric emission profiles [Pegoraro et al. 2009]. The same authors then extended the technique to support arbitrary phase functions and emission profiles [Pegoraro et al. 2010; 2011]. A common drawback to these approaches is that the phase function and the emission profile have to be expanded into a Taylor series, which makes the evaluation of the anti-derivative fairly expensive.

All of the aforementioned analytic techniques assume full visibility between the point light and the camera ray. To overcome this major limitation, Biri et al. [2006] employ volumetric shadows to identify segments of the ray that are fully visible, and break the airlight integral into a sum over these segments. Another option is to employ the analytic solution only as a control variate and sample the airlight integral numerically. The resulting images suffer from noise only in regions where the camera rays are partially occluded.

2.8.2 Monte Carlo Integration

Finding an anti-derivative can be often complicated or even impossible. In such cases, we have to employ numerical integration, i.e. we sample the value of the integral over a finite set of points and average the obtained values. Quadrature and cubature rules distribute these points uniformly using regular grids, and are known to be most efficient on low-dimensional integrals. They can be extended to multiple dimensions by recursively expanding the one-dimensional integration; however, as the number of dimensions increases, the number of evaluations of the integral grows exponentially. To overcome the *curse of dimensionality*, we can instead evaluate the integral on a set of multi-dimensional points. When these points are defined using a random or quasi-random sequence, we talk about Monte Carlo (MC) or quasi-Monte Carlo (QMC) integration, respectively.

The most important and distinctive property of Monte Carlo methods stems from the *central limit theorem*. Given a sequence of N independent and identically distributed random variables $X_1, X_2, ..., X_N$ with common mean μ and standard deviation σ, the average of these variables:

$$\bar{X}_N = \frac{1}{N} \sum_{i=1}^{N} X_i, \tag{2.100}$$

has approximately normal distribution $(\mu, \sigma/\sqrt{N})$. An important observation here is that X_i does not have to be normally distributed; the averaging works for any sequence of random variables as long as they have a well defined μ and σ. Informally, the central limit theorem says that by increasing N, we narrow the normal distribution of the average. In the limit, i.e. for $N \to \infty$, $\bar{X}_N = \mu$ and $\sigma = 0$.

Consider a multi-dimensional integral F of a function f defined over a domain $\mathcal{D}$:

$$F = \int_{\mathcal{D}} f(x)\mathrm{d}x. \tag{2.101}$$

In order to estimate the value of F, we define an estimator $\langle F^N \rangle$ that approximates F by averaging the value of f on N randomly chosen points from $\mathcal{D}$:

$$F \approx \langle F^N \rangle = \frac{V}{N} \sum_{i=1}^{N} f(X_i),$$ (2.102)

where V is the size of $\mathcal{D}$:

$$V = \int_{\mathcal{D}} \mathrm{d}x.$$ (2.103)

The estimator $\langle F^N \rangle$ itself is a random variable, whose value depends on the number of sampled points and their distribution. For a set of uniformly chosen points (i.e. $p(X_i) = 1/V$) it can be easily shown that the expected value of $\langle F^N \rangle$ equals to F:

$$
\begin{aligned}
E\left[\langle F^N \rangle\right] &= E\left[\frac{V}{N} \sum_{i=1}^{N} f(X_i)\right] \\
&= \frac{V}{N} \sum_{i=1}^{N} E\left[f(X_i)\right] \\
&= \frac{V}{N} \sum_{i=1}^{N} \int_{\mathcal{D}} f(x)p(x)\mathrm{d}x \\
&= \frac{1}{N} \sum_{i=1}^{N} \int_{\mathcal{D}} \frac{f(x)p(x)}{p(x)}\mathrm{d}x \\
&= \int_{\mathcal{D}} f(x)\mathrm{d}x \\
&= F.
\end{aligned}
$$ (2.104)

We can also express the variance:

$$
\begin{aligned}
\sigma^2\left[\langle F^N \rangle\right] &= \sigma^2\left[\frac{V}{N} \sum_{i=1}^{N} f(X_i)\right] \\
&= \frac{1}{N^2} \sum_{i=1}^{N} \sigma^2\left[\frac{f(X_i)}{p(X_i)}\right] \\
&= \frac{1}{N}\sigma^2\left[\frac{f(X)}{p(X)}\right],
\end{aligned}
$$ (2.105)

and the standard deviation of the estimator:

$$\sigma\left[\langle F^N \rangle\right] = \frac{1}{\sqrt{N}}\sigma\left[\frac{f(X)}{p(X)}\right].$$ (2.106)

The above equations highlight two important properties of Monte Carlo estimators: first, the estimators converge at the rate of $O(N^{-1/2})$ (i.e. to reduce the error by a factor of 2 we need to draw 4× more samples), second, the variance of the estimator depends on the probability $p(X)$, which is used to draw the samples. In many cases, we have some *a-priori* information about the integrand, which can be used to determine a suitable distribution for choosing the samples.

A Brief History of Monte Carlo Methods.

Monte Carlo methods were originally developed in 1940s in the Los Alamos National Laboratory to numerically verify and improve designs of thermonuclear weapons. The code name "Monte Carlo" was proposed by Ulam and von Neumann to emphasize that these simulations involve some sort of random decisions, as do the games in the famous casino of Monte Carlo. The fundamental paper describing the Monte Carlo method was published by Metropolis and Ulam [1949] in 1949; however, this was not the first time when random sampling was used in calculations. In 1777, Comte de Buffon proposed an experiment that involved dropping a needle many times on parallel lines to probabilistically estimate the value of π. In order to generate random numbers, Lord Kelvin drew slips of paper out of a glass jar when studying kinetic energy of gases. The initial application of Monte Carlo sampling to transport theory is sometimes attributed to Fermi, who applied a similar approach to simulations of neutron transport in 1930s, but did not publish anything on the topic. More information about the early applications and history of MC sampling can be found in [Hammersley and Handscomb 1964, Kalos and Whitlock 1986].

Monte Carlo methods play an important role in transport and diffusion theory. Some particles (e.g. neutrons) penetrate deep into solid objects undergoing many interactions with the medium inside. In order to simulate these, MC approaches create random walks that mimic the transport of neutrons throughout the domain of interest. A survey of early MC techniques for neutron transport was assembled by Spanier and Gelbard [1969]. The authors categorize the approaches as collision (discrete) estimators and track length (continuous) estimators. Most of these estimators construct random walks analogously to the underlying physical processes, i.e. the random walks are terminated when the neutron is absorbed. A different family of estimators can be derived from these *analog* processes by forbidding absorption [Gelbard et al. 1966, Spanier 1966]. Such *non-analog* random walks require to be terminated using some artificial termination rule (e.g. Russian roulette), which can in some cases lead to a lower variance than with analog estimators. Coleman [1968] verifies some of these estimators mathematically and Lux and Koblinger [1991] provide a more up-to-date comparison of the different analog and non-analog estimators.

Simulating the transport of photons in participating media shares many similarties to neutron transport. One interesting example is the class of *next event estimators* (NEE) developed for computing the energy at a given point [Kalos 1963]. These estimators compute the energy transported to the point detector explicitly after each scattering event. Such approaches however suffer from a severe problem: since the detector is approximated by a point, the estimator contains a $(1/d^2)$-singularity causing a theoretically infinite variance. Steinberg and Kalos [1971] proposed to incorporate the $(1/d^2)$-singularity into the PDF for sampling the scattering event, which effectively cancels the original singularity and results in an unbiased estimator with finite variance. An analogous sampling scheme, called the *equi-angular* sampling, has recently been introduced to the field of computer graphics by Kulla and Fajardo [2012] to minimize the variance when estimating inscattered light from a point light along a ray. To reduce the degree of the singularity, Kalos [1963] also proposed the *once-more collided flux estimator*, which adds an additional scattering event and reduces the singularity to $1/d$. Georgiev et al. [2013] propose an efficient sampling strategy for this kind of light transport.

In the early years of computer graphics, Monte Carlo concepts were leveraged by Appel [1968] who shot random light rays from light sources to estimate shadows and spatial intensity of light on solid objects. Whitted [1980] took an opposite approach (sometimes referred to as back-

ward ray tracing[11]) by tracing rays from the camera and possibly extending them recursively into paths if they hit specular surfaces. Cook et al. [1984] extended Whitted's ray tracing to support distribution effects, such as depth of field or motion blur, by stochastically integrating over time and lens aperture. In 1986, Arvo [1986] proposed to trace rays recursively from the light sources to synthesize caustics. In the same year, Kajiya [1986] formulated light transport using a single integral equation (in principle Equation (2.20)) and proposed to solve it using Monte Carlo sampling of paths. His *path tracing* became a *de facto* standard for computing reference solutions inspiring many of the follow-up publications. Lafortune and Willems [1993] and Veach and Guibas [1994] independently proposed to combine path tracing and light tracing in a unified bidirectional tracing framework, for which Veach and Guibas [1994] devised a number of strategies to properly weight the different estimators. The same authors then applied Metropolis-Hastings algorithm to light transport [Veach and Guibas 1997] to amortize the cost of constructing paths by perturbing those with high contribution to construct new ones. Several authors extended this technique to further improve the convergence in specific situations [Cline et al. 2005, Jakob and Marschner 2012, Kaplanyan and Dachsbacher 2013b, Lehtinen et al. 2013].

Unbiased vs. Biased Estimators

The variance of Monte Carlo estimators can be further reduced by correlating the estimates and/or caching and reusing results over several queries. Examples of such algorithms include *irradiance caching* [Ward et al. 1988], *volumetric radiance caching* [Jarosz et al. 2008a], *density estimation* approaches [Jensen 1996, Shirley et al. 1995], or *instant radiosity* [Keller 1997]. These approaches are better in suppressing noise; however, they generally introduce approximations that bias the estimator. While unbiased estimators yield correct results on average, biased estimators systematically alter the result. If the estimator is *consistent*, the systematic error can be made arbitrarily small by taking more samples; however, the convergence rate of consistent rendering algorithms is often worse than in the case of MC approaches (e.g. progressive photon mapping [Hachisuka et al. 2008b, Knaus and Zwicker 2011] converges at the rate of $O(N^{-1/3})$ [Kaplanyan and Dachsbacher 2013a] while path tracing converges at $O(N^{-1/2})$).

In his thesis [Veach 1997], Veach argues that unbiased estimators are preferred over biased ones, simply because the error of the algorithm is guaranteed to manifestate itself as random variation of the estimator, and thus easy to quantify: we just need to compute the variance of the sample. In contrast, the error of biased algorithms is generally hard to assess and the only robust approach is to compare the results to those of an unbiased algorithm. There is no doubt that unbiased algorithms have a certain advantage over biased ones; nevertheless, there are still situations when employing biased estimators makes sense, or is even the only viable option. Consider for instance a scene consisting of a swimming pool with a wavy water surfaces illuminated by a point light. When the scene is rendered using a pinhole camera, computing the illumination at the bottom of the pool is generally impossible with unbiased path sampling techniques. This is because the delta functions, i.e. the point light, the pinhole camera, and the specular water surface, make the problem overconstrained for stochastic sampling. In contrast, consistent methods can easily discover the corresponding paths and render an image that contains all light transport, be it at the cost of a small error.

---

[11]The terms *forward* and *backward* ray tracing are sometimes interchanged. Arvo [Arvo 1986] used the term backward ray tracing for rays that are shot from the light source. In 1995, the author added an addendum to the paper regretting the chosen terminology as this caused a lot of confusion. We shall thus use the term "forward" when talking about rays shot from the light source, and "backward" for rays shot from the camera.

Another example when biasing the computation is desirable are many-light algorithms. If formulated without any systematic error, the estimator suffers from unbounded variance [Kollig and Keller 2006]. Introducing a certain amount of bias, either by removing or spatially blurring the energy, removes distracting artifacts and produces images that are visually more pleasing than those obtained with the original unbiased formulation.

2.8.3   Finite Element Methods

The rendering algorithms described in the previous section focus on estimating the equilibrium radiance over points that are visible to camera, and they are thus view-dependent. We shall also mention algorithms that try to compute the illumination in *world space* in a view-independent manner. The solution can be subsequently used for several camera positions or entire animations. These algorithms first discretize the scene by tessellating the geometry into a finite number of patches and then seeks for the equilibrium diffuse energy, which is defined by a set of linear equations describing the exchange of light between individual patches. Since outgoing diffuse illumination is well modeled by radiosity, finite element methods used for rendering are often called *radiosity algorithms* [Cohen and Greenberg 1985, Goral et al. 1984, Nishita and Nakamae 1985].

The fact that the solution needs to be view-independent restricts the application to diffuse scenes only. Many publications try to overcome this limitation by computing the view-dependent transport separately [Smits et al. 1992], or by supporting more complex reflections [Aupperle and Hanrahan 1993, Immel et al. 1986, Sillion et al. 1991, Wallace et al. 1987]. Researchers also tried to adjust the tesselation to the structure of the light transport by importance-driven refinement [Smits et al. 1992], discontinuity meshing [Lischinski et al. 1992], or clustering [Smits et al. 1994]. Rushmeier and Torrance [1987] extended radiosity methods to handle participating media. There are also several approaches that combine finite element methods with MC sampling, e.g. *Monte Carlo radiosity* estimates the coefficients for the linear equations by tracing light particles [Pattanaik and Mudur 1993, Shirley 1990].

Despite many of these improvements, the cost of evaluating the light transport remains tied to the geometric complexity of the scene. Applications of radiosity algorithms to high quality rendering are thus today rare.

Many-Light Methods

*Education is man's going forward from cocksure
ignorance to thoughtful uncertainty.*

— KENNETH G. JOHNSON (1922–2002)

O ver the last few decades, we have been witnessing an increasing demand for visualizing virtual scenes across many industries. The demand sparked numerous research activities resulting in rendering algorithms, which allow rasterizing virtual geometry into a set of colored pixels. In this dissertation, we focus on synthesizing images that look virtually indistinguishable from photographs. There are different perspectives that one can take on this problem, but in general, the goal is to solve the radiative transfer equation in either its full complexity or in one of the simplified forms.

At the end of the previous chapter, we mentioned some of the Monte Carlo techniques that evaluate the integro-differential equation by recursive sampling. Thanks to the central limit theorem, these approaches yield correct results, on average, but the results often suffer from high amount of noise. Several algorithms strive to overcome this and accelerate rendering by reusing computation and/or correlating estimates. These techniques split the simulation of transport between emitters and cameras into two phases. In the first phase, the algorithm distributes information about (multi-bounce) illumination coming from emitters, and stores it in a form of e.g. irradiance samples [Ward et al. 1988], a photon map [Jensen 1996], or a collection of virtual point lights [Keller 1997]. The second phase is then responsible for connecting these samples to the camera e.g. by shooting primary rays and estimating the density of photons. While the scheme may seem convoluted, the great advantage of these algorithms is that they can reuse results of the first part across multiple pixel queries, thereby reducing the variance.

Many-light methods, see Figure 3.1 for illustrations of the two phases, fall into this category. The advantage of many-light algorithms is their simple, unified and adaptable solution to many difficult rendering problems. The core insight is that the general light transport can be approximated by the simpler task of calculating direct illumination from *many* virtual light sources. This gives many-light algorithms two distinct advantages. First, it provides a unified and straightforward mathematical framework for calculating global illumination. Second, it makes many-light algorithms very adaptable: the same algorithm can be adjusted to meet a wide range of quality and performance goals. For instance, by using fewer virtual sources, many-light methods can produce biased, but artifact free images in a fraction of a second; this makes them attractive for real-time rendering applications. On the other hand, by using sufficiently many virtual sources with a highly scalable evaluation algorithm, any bias from the virtual source approximation
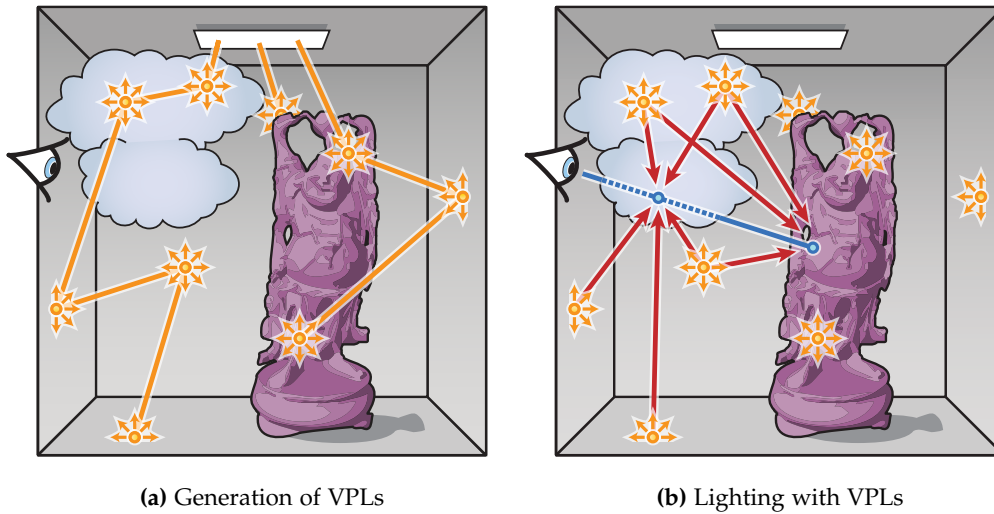
| **(a)** Generation of VPLs | **(b)** Lighting with VPLs |

Figure 3.1: Many-light algorithms operate in two passes: first, they distribute a number virtual lights, e.g. virtual point lights (VPLs) (a), and then use them to illuminate the scene and by this approximate indirect illumination (b).

can be reduced below the perceptible level. The algorithm then produces results comparable to unbiased methods in less time and becomes appealing even for high-fidelity applications.

In this chapter, we provide a coherent summary of many-light rendering. The structure and the content of the chapter is based on the state-of-the-art report by Dachsbacher et al. [2014]. First, we explain the fundamental concept of many-light algorithms in Section 3.1. Then we describe the two important components: how to generate virtual lights and how to use them to approximate global illumination, in Sections 3.2 and 3.3, respectively. Finally, we review extensions addressing scalability and performance of many-light algorithms in offline and real-time rendering in Sections 3.4 and 3.5, respectively.

3.1   Algorithm Overview

The basis for all many-light algorithms was established by Alexander Keller in 1997 in a paper called *instant radiosity* [Keller 1997]. The algorithm makes a key observation that complex global illumination can be approximated by direct illumination from a set of specifically distributed virtual point lights (VPLs). Similarly to the original work, we will introduce the algorithm by describing the method for surfaces only, and later generalize to include participating media as a part of the precise mathematical formulation in Sections 3.2 and 3.3.

In principle, all global illumination MC methods evaluate light transport by constructing *light transport paths*, along which light travels from light sources to camera sensors. Path tracing [Kajiya 1986], for example, constructs paths by tracing rays starting from the camera. In order to handle complex light transport more robustly, bidirectional path tracing [Lafortune and Willems 1993, Veach and Guibas 1994] traces sub-paths from camera as well as from light sources and then (deterministically) connects them to form full paths.

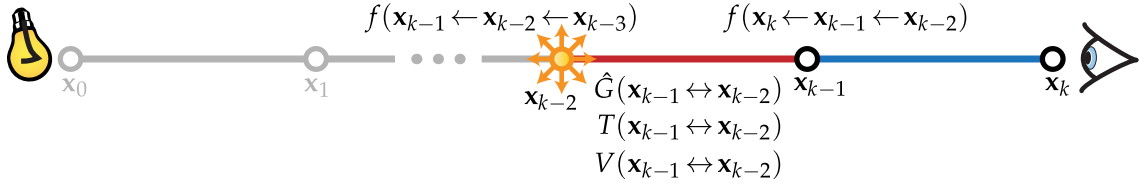Instant radiosity (IR) is a variant of bidirectional path tracing that constructs the two sets of sub-

Figure 3.2: Illustration of terms involved in computing the contribution of a single VPL to a shading point.

paths in a specific manner. Recall the measurement contribution function from Equation (2.91), for which we now define a Monte Carlo estimator:

$$\langle I_j(\bar{x}) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{\hat{L}_e(\mathbf{x}_0 \to \mathbf{x}_1) \mathcal{T}(\mathbf{x}_0 \ldots \mathbf{x}_k) W(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1})}{p(\mathbf{x}_0 \ldots \mathbf{x}_k)}, \tag{3.1}$$

where $p(\mathbf{x}_0 \ldots \mathbf{x}_k)$ is the joint probability distribution for constructing the path. Keller's key observation here is that we can split the evaluation of the estimator into arbitrarily long light paths $\mathbf{x}_0 \ldots \mathbf{x}_{k-2}$ and rather short[1] camera paths $\mathbf{x}_{k-1}, \mathbf{x}_k$. A single light path can be then reused to calculate illumination of many different points $\mathbf{x}_{k-1}$ seen by the camera.

Specifically, instant radiosity precomputes a number of path prefixes of the form $\mathbf{x}_0 \ldots \mathbf{x}_{k-2}$ and stores their end vertices $\mathbf{x}_{k-2}$ as virtual point lights. With each VPL i the algorithm also stores the partially evaluated estimator from Equation (3.1), which is usually referred to as the "flux" of the VPL:

$$\Phi_i = \frac{1}{N} \frac{\hat{L}_e(\mathbf{x}_0 \to \mathbf{x}_1) \mathcal{T}(\mathbf{x}_0 \ldots \mathbf{x}_{k-2})}{p(\mathbf{x}_0 \ldots \mathbf{x}_{k-2})}. \tag{3.2}$$

Additionally, we also store information necessary to use the VPL to "illuminate" or to "connect to" a given point $\mathbf{x}_{k-1}$, i.e. to evaluate the following terms (see Figure 3.2):

$$f(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}) V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) \hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) f(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2} \leftarrow \mathbf{x}_{k-3}). \tag{3.3}$$

This includes a reference to the scattering function at the VPL location $\mathbf{x}_{k-2}$, the direction towards $\mathbf{x}_{k-3}$, and the local tangent frame if $\mathbf{x}_2$ resides on a surface. In practice, it is often assumed that the scattering function at the VPL location is view independent (i.e. Lambertian if $\mathbf{x}_{k-2} \in \partial \mathbb{V}$ and isotropic if $\mathbf{x}_{k-2} \in \mathbb{V}$). In this case, we do not need to store the incident direction and we can premultiply the VPL "flux" [W] by the value of the scattering function [sr^{-1}], to obtain the VPL "intensity" [W.sr^{-1}]. It is often simpler and less error-prone to think about VPLs in terms of partial evaluations of the estimator in Equation (3.1) rather than in terms of flux or intensity.

In general, many-light algorithms consist of two phases (see Figure 3.1 for illustration):

Phase 1: Generation of VPLs

First, a large set of light sub-paths with arbitrary length is generated and stored. For each vertex of these sub-paths, the local geometric and material information and the current "flux" (i.e. emitted radiance from the light source multiplied by the path throughput to the vertex, divided by the probability density of constructing the path to that point) is

---

[1]In case of the primary ray hitting a specular or highly glossy surface, some variants of IR extend the camera path until it reaches a diffuse or moderately glossy surface. The camera path can thus consist of more than two vertices.

recorded. The intention is that the data stored for each vertex suffice to compute the outgoing illumination scattered from this vertex into any direction. If this is true, we can discard the notion of the original path and instead model the vertex as an unusual type of point light source. Since these do not correspond to any physical light sources in the scene, we call them *virtual point lights*.

Phase 2: Lighting with VPLs

In order to complete the IR algorithm, camera sub-paths are constructed for each pixel in the second phase. Since the light sub-paths were arbitrarily long, it is sufficient to consider only length-one camera paths. Then, like other bidirectional algorithms, IR connects vertices of these camera sub-paths to vertices of the light paths to form full paths. This step amounts to computing direct illumination of directly seen surfaces due to the VPLs.

The two phase IR algorithm is often more efficient than general bidirection methods for two reasons. First, since each VPL is used to illuminate surface points seen through all (or many) image pixels, the effort invested into generating the VPLs is well amortized. Additionally, the use of a single set of VPLs to illuminate all surface points produces correlated pixel values. This property is extremely efficient in visually suppressing noise, which is typical for traditional Monte Carlo approaches that build *independent* paths for each pixel. Note that this latter advantage has little to do with reducing numerical error – it is purely of perceptual nature.

The original instant radiosity method [Keller 1997] was not the end-all solution to the global illumination problem. Its specific strategy for constructing transport paths has advantages as well as drawbacks. Common to most variants is that they are relatively simple to implement and quickly yield visually pleasing results at predictable rendering costs. On the other hand, IR methods are prone to splotchy artifacts (imagine the entire light energy in a scene contracted to few VPLs), and have difficulties with high frequency global illumination, e.g. in scenes with highly glossy surfaces. We elaborate on these problems later, point to publications that try to overcome them, and describe our original solutions.

3.2   Generation of Virtual Point Lights

In this section, we detail the first phase of the algorithm, and describe a common random walk procedure used to distribute VPLs. We also discuss improvements that were developed to direct the VPLs into visually important parts of the scene.

3.2.1   Random Walk VPL Distribution

The most common approach to generate VPLs is to distribute them using several random walks. Indeed, we can use the same particle tracing procedure as for distributing photons in photon mapping [Jensen 1996]. We start by tracing N light paths, which originate at light sources, creating M VPLs; one at each bounce (vertex) of the light path. We proceed in the following steps:

1. **Sample the first vertex.** The first vertex $\mathbf{x}_0$ we create on a light source. Usually, $\mathbf{x}_0$ is sampled from a PDF $p(\mathbf{x}_0)$ that is proportional to the radiant exitance of individual emitters. We also initialize the vertex index j to 0.
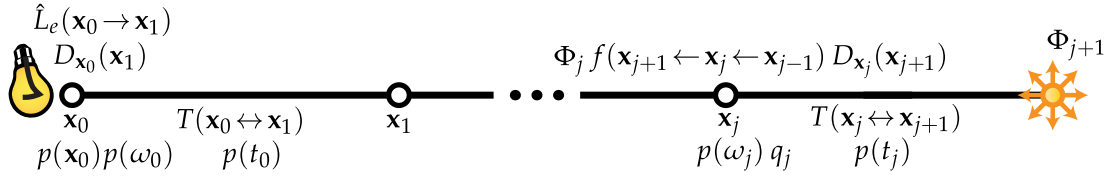
Figure 3.3: This illustration shows the terms that are required to compute the "flux" of a VPL during the construction of random walks.

2. **Sample the next path vertex.** Then we sample direction ω_j from a PDF $p(\omega_j)$ proportional to the directional emission distribution of the light source (for the first path vertex, $j = 0$) or to the generalized scattering function (for other vertices, $j > 0$). In scenes without participating media, this direction uniquely determines the next path vertex. To account for media, we need to sample the free path t_j along the ray $\mathbf{r}_j(t) = \mathbf{x}_j + t\omega_j$, i.e. how far the particle travels before interacting with the medium. Please refer to Section 2.6.8 for details on how to sample the free path. If the sampled distance extends beyond the nearest surface along ω_j, the next path vertex $\mathbf{x}_{j+1}$ will be the corresponding surface intersection point. Otherwise, the next vertex resides in the medium.

3. **Create a VPL.** A virtual point light is stored at the position of the generated path vertex $\mathbf{x}_{j+1}$. The "flux" of the VPL is calculated as:

$$\Phi_{j+1} = \begin{cases} \dfrac{\hat{L}_e(\mathbf{x}_0 \to \mathbf{x}_1) \, D_{\mathbf{x}_0}(\mathbf{x}_1) \, T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)}{p(\mathbf{x}_0) \, p(\omega_0) \, p(t_0)} & \text{if } j = 0 \\[2ex] \Phi_j \dfrac{f(\mathbf{x}_{j+1} \leftarrow \mathbf{x}_j \leftarrow \mathbf{x}_{j-1}) \, D_{\mathbf{x}_j}(\mathbf{x}_{j+1}) \, T(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1})}{p(\omega_j) \, p(t_j) \, q_j} & \text{if } j > 0. \end{cases} \tag{3.4}$$

As mentioned before, for every VPL we also store a reference to the scattering function f at the VPL location $\mathbf{x}_{j+1}$, the incident direction ω_j, and the local tangent frame if $\mathbf{x}_{j+1} \in \partial \mathbb{V}$.

4. **Terminate or continue the path.** Use Russian roulette to terminate the random walk with probability $(1 - q_{j+1})$, where the survival probability q_{j+1} is usually proportional to the albedo of the surface or volumetric point $\mathbf{x}_{j+1}$, or the throughput of the path. If the random walk survives the Russian roulette, set $j := j + 1$ and go to step 2.

After finishing all random walks we divide the "flux" of each VPL by the total number of generated light paths N. Note that the "flux" of the VPLs created by the above procedure exactly corresponds to all the terms in Equation (3.2); see Figure 3.3 for illustration.

When the scene does not contain any participating media, we can simplify the random walk procedure by omitting the free path sampling. The generated path vertex $\mathbf{x}_{j+1}$ is always on the nearest surface and the transmittance $T(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1})$ and the PDF $p(t_j)$ equal to 1 and cancel out.

3.2.2 Improved VPL Generation

The random walk procedure described previously does not take into account the position of the camera. It may thus generate many VPLs with only a small contribution to points in the view frustum. The problem becomes even more prominent in large environments where the camera

looks at a small part of the scene. Additionally, the density of VPLs generated by the basic VPL generation algorithm along concave geometry is usually insufficient to faithfully render local interreflections. In this section, we briefly discuss various approaches developed for generating VPLs where they are most needed for a given camera view.

Rejection of Unimportant VPLs.

A straightforward approach to avoid VPLs with negligible contribution is to probabilistically reject the unimportant ones. The algorithm, as proposed by Georgiev and Slusallek [2010], generates many candidate VPLs using the previously described random walk procedure, but keeps only those with significant contribution to the entire image. The authors first estimate the average VPL contribution Φ_μ by creating a number of pilot VPLs and rendering a low resolution image. Then they generate candidate VPLs and for each candidate they estimate its contribution Φ_i by calculating the light it delivers to a few pixels of the image. The candidate is accepted with probability:

$$p_i = \min \left\{ \frac{\Phi_i}{\Phi_\mu} + \epsilon, 1 \right\}, \tag{3.5}$$

which is proportional to the ratio of the actual VPL contribution to the average contribution. If accepted, the VPL flux is divided by p_i to ensure unbiasedness. This approach is essentially a Russian roulette driven by the relative expected contribution, which produces a set of VPLs with approximately the same contribution to the image.

Metropolis Instant Radiosity.

The rejection sampling approach described above is simple but suffers from an important disadvantage: many candidate VPLs may need to be generated before one VPL is accepted. To create VPLs that are relevant to the camera straight from the beginning, Segovia et al. [2007] propose to replace the standard VPL tracing by a Metropolis-Hastings sampler. Consider a path that connects a light source to the camera. As discussed in Section 3.1, the second vertex from the camera can be interpreted as a VPL. We can now use the Metropolis-Hastings procedure, as in the Metropolis light transport algorithm [Veach and Guibas 1997], to explore the space of all possible light paths by proposing and probabilistically accepting path mutations. Every time a path is mutated, the second vertex from the camera of the mutated path yields a new VPL. Segovia et al. [2007] shows that all VPLs created in this way contribute the exact same total flux to the image.

Though very different, both the VPL rejection algorithm [Georgiev and Slusallek 2010] and Metropolis Instant Radiosity [Segovia et al. 2007] generate VPL sets where each VPL has at least roughly the same contribution to the image. From this, we can expect that the VPL sets generated by both algorithms will be of similar quality. For complex scenes, where light needs to bounce many times to reach the camera, the rejection algorithm may perform poorly because it will reject many VPLs. On the other hand, while the VPL rejection approach is trivial, Metropolis Instant Radiosity requires a substantial implementation effort. Finally, as the VPL distribution is driven by the contribution of VPLs to the *entire* image, none of the two algorithms addresses glossy interreflections.

Creating VPLs from the Camera.

The density of VPLs distributed by the generation algorithms discussed so far is usually insufficient to faithfully render local interreflections in geometric cavities. To deal with this problem, it may be more appropriate to distribute the VPLs by tracing paths from the camera instead of starting at the light sources. This approach is likely to produce VPLs in locations important for the image. The idea of generating VPLs by tracing paths from the camera appeared first in [Segovia et al. 2006a] under the name Bidirectional Instant Radiosity. It was later also used by Davidovič et al. [2010], who refer to the VPLs generated from the camera as local VPLs (as opposed to global VPLs, generated by tracing paths form the light sources).

3.3 Lighting with Virtual Point Lights

Once the VPLs are generated, many-light algorithms use them to illuminate the scene, i.e. to calculate the outgoing radiance $L(\mathbf{x}_{k-1} \to \mathbf{x}_k)$ scattered from points $\mathbf{x}_{k-1}$ in the view frustum towards points $\mathbf{x}_k$ in the camera. This amounts to summing over all M VPLs and adding together their "flux" Φ_i weighted by the terms from Expression (3.3). The estimator of $L(\mathbf{x}_{k-1} \to \mathbf{x}_k)$ reads:

$$
\begin{aligned}
\langle L(\mathbf{x}_{k-1} \to \mathbf{x}_k) \rangle \ = \ & \sum_{i=1}^{M} f(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}^i) V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}^i) T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}^i) \hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}^i) \\
& f(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}^i \leftarrow \mathbf{x}_{k-3}) \Phi_i,
\end{aligned}
\tag{3.6}
$$

where $\mathbf{x}_{k-2}^i$ is the position of i-th VPL.

Although the equation above—the core part of many-light methods—is fairly simple and can be evaluated efficiently, closer inspection reveals the major problem of many-light algorithms. The geometry term $\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y})$ contains a $(1/d^2)$-singularity with d representing the distance between the shading point and the VPL (see Equation (2.82) for definition). Since the distance can be arbitrarily small, the contribution of a VPL to a shading point, and thus the variance of the estimator, is possibly unbounded. This is in a sense the same problem as in a *naive* bidirectional path tracing that does not employ a proper weighting of multiple estimators. The closer the two vertices being connected are, the higher the value of the sample. As a result, some pixels in the resulting path-traced image are very bright.

With many-light algorithms, the $(1/d^2)$-singularity stands out even more clearly than in *naive* bidirectional path tracing. This is because we correlate the estimates by connecting all shading points to the same set of VPLs. Note that the correlation itself does not compromise the mathematical correctness of the algorithm, i.e. the estimation is still unbiased. However, we ultimately trade noise for structured artifacts, which, thanks to the $(1/d^2)$-singularity, show up very clearly as high-intensity splotches, and are often more distracting than stochastic noise (see Figure 3.4.a). Since the splotches heavily degrade the quality of rendered images, we devote the next few sections to existing techniques that try to avoid them.
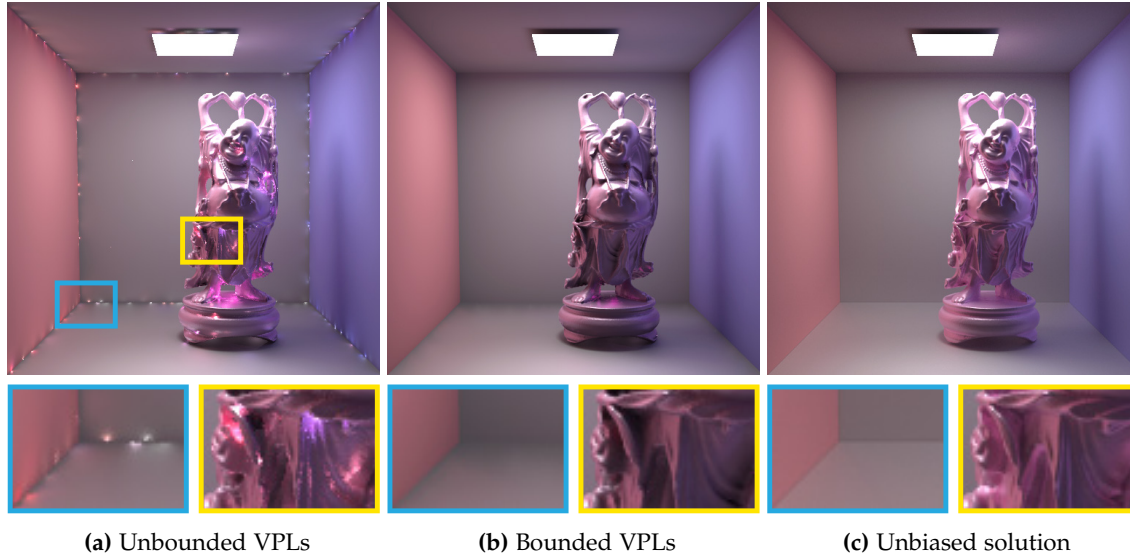
(a) Unbounded VPLs   **(b)** Bounded VPLs   **(c)** Unbiased solution

Figure 3.4: Adding the full contribution of each VPL leads to high-intensity splotches (a). When the contribution is bounded (b) these artifacts disappear; however, the rendered image becomes locally darker than the ground-truth (c).

3.3.1 Bounded Estimation

A straightforward and popular approach to suppress the bright splotches is to bound the geometry term to a user-defined maximum b. By *bounding*[2] we guarantee that the geometry term does not exceed b, no matter how close the shading point to a VPL is. This technique was used already in the original instant radiosity method [Keller 1997], where the bounding occurred as a consequence of implementing the method in hardware, which at that time clamped all color values to $\langle 0, 1 \rangle$. Let us now define a *bounded geometry term* $\hat{G}_b(\mathbf{x} \leftrightarrow \mathbf{y})$:

$$\hat{G}_b(\mathbf{x} \leftrightarrow \mathbf{y}) \quad = \quad \min(\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}), b). \tag{3.7}$$

By substituting $\hat{G}_b(\mathbf{x} \leftrightarrow \mathbf{y})$ for $\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y})$ in the estimator from Equation (3.6):

$$\langle L(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k) \rangle \quad = \quad \sum_{i=1}^{M} f(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}^i) V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}^i) T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}^i) \hat{G}_b(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}^i)$$
$$f(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}^i \leftarrow \mathbf{x}_{k-3}) \Phi_i, \tag{3.8}$$

we avoid the structured artifacts; however, we partly suppress short distance light transport, and thus obtain a solution that is biased. Images rendered with the bounded geometry term suffer from artificial darkening in regions with locally concave geometry, such as corners and cavities (see Figure 3.4.b for examples). Furthermore, selectively suppressing light transport can have a severe impact on material appearance [Křivánek et al. 2010], and should be in such cases either avoided or compensated for.

---

[2]Sometimes also called *clamping*.

3.3.2 Bias Compensation via Final Gathering

One way to avoid changes in material appearance and to ensure unbiased renderings is to add a correction term that recovers the missing light transport. Kollig and Keller [2006] express the energy removed due to bounding, denoted $B(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k)$, as the difference between the transport obtained with the original and the bounded geometry term:

$$B(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k) = \int_{\mathbb{R}^3} f(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}) V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2})$$
$$\hat{G}_r(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) L(\mathbf{x}_{k-2} \rightarrow \mathbf{x}_{k-1}) \mathrm{d}\mu(\mathbf{x}_{k-2}), \tag{3.9}$$

where $\hat{G}_r(\mathbf{x} \leftrightarrow \mathbf{y})$ is the *residual geometry term* transporting only the missing light:

$$\begin{aligned} \hat{G}_r(\mathbf{x} \leftrightarrow \mathbf{y}) &= \hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}) - \hat{G}_b(\mathbf{x} \leftrightarrow \mathbf{y}) \\ &= \hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}) - \min(\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}), b) \\ &= \max(\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y}) - b, 0). \end{aligned} \tag{3.10}$$

In order to remove the singularity from Equation (3.9), Kollig and Keller propose to change the domain of integration to the unit sphere $\mathcal{S}^2$. Raab et al. [2008] extend the formulation to support participating media by adding an additional integral to account for points along the sampled direction. Incorporating both of these changes yields:

$$\begin{aligned} B(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k) &= \int_{\mathbb{R}^3} f(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} \leftarrow \mathbf{x}_{k-2}) V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) \\ & \quad \frac{\hat{G}_r(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2})}{\hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2})} \hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_{k-2}) L(\mathbf{x}_{k-2} \rightarrow \mathbf{x}_{k-1}) \mathrm{d}\mu(\mathbf{x}_{k-2}) \\ &= \int_{\mathcal{S}^2} f(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} \leftarrow \omega) \, D_{\mathbf{x}_{k-1}}(\omega) \int_0^d T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{y}) \frac{\hat{G}_r(\mathbf{x}_{k-1} \leftrightarrow \mathbf{y})}{\hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{y})} L(\mathbf{y} \rightarrow \mathbf{x}_{k-1}) \mathrm{d}t \mathrm{d}\omega, \end{aligned}$$
$$\tag{3.11}$$

where $\mathbf{y}$ is a point at distance t on a ray with origin $\mathbf{x}_{k-1}$ and direction ω; d is the distance to the nearest surface seen along the ray, and $D_{\mathbf{x}}(\omega)$ is defined as:

$$D_{\mathbf{x}}(\omega) = \begin{cases} n(\mathbf{x}) \cdot \omega & \text{if } \mathbf{x} \in \partial \mathbb{V} \\ 1 & \text{if } \mathbf{x} \in \mathbb{V}. \end{cases} \tag{3.12}$$

In order to obtain unbiased results, Kollig and Keller add the bias compensation term $B(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k)$ to the bounded solution from Equation (3.8). As suggested in Equation (3.11), B can be estimated via localized final gathering, i.e. by shooting rays towards nearby surfaces, calculating the outgoing radiance therefrom, and transporting back only such amount of energy that corresponds to the removed VPL lighting.

Although the compensation is localized, the proposed implementation is fairly costly since $L(\mathbf{y} \rightarrow \mathbf{x}_{k-1})$ is estimated using all light sources and *all* VPLs. As bounding occurs during the compensation, the technique is recursive and quickly degenerates to path tracing (see Figure 3.5). Another pitfall is that $\hat{G}_r$ is often zero during the MC integration and $B(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k)$ thus suffers from high variance. As a result, the cost of obtaining unbiased results with acceptable amount of noise can exceed the computation time of the bounded transport by orders of magnitude, which makes the algorithm less attractive for practical applications.
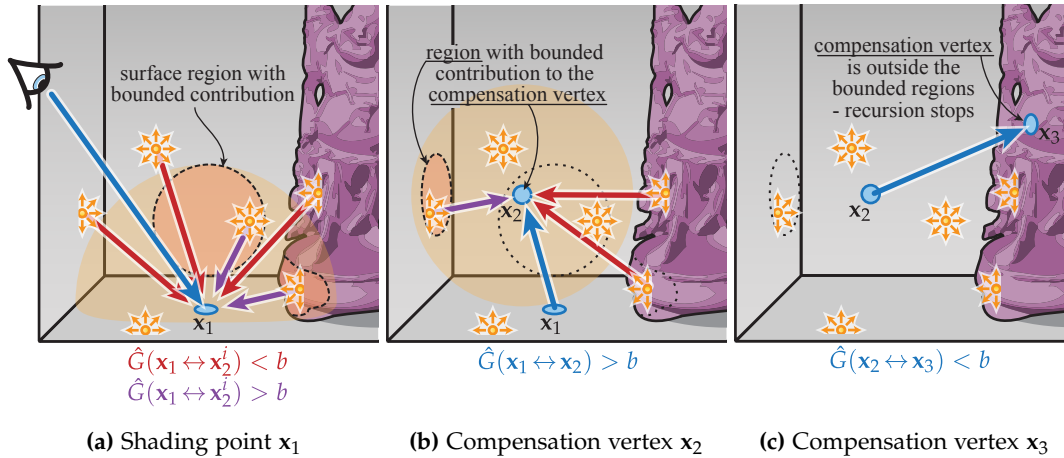
$$\hat{G}(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2^i) < b$$
$$\hat{G}(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2^i) > b$$

(a) Shading point $\mathbf{x}_1$

$$\hat{G}(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) > b$$

(b) Compensation vertex $\mathbf{x}_2$

$$\hat{G}(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3) < b$$

(c) Compensation vertex $\mathbf{x}_3$

Figure 3.5: When computing the illumination of a shading point $\mathbf{x}_1$, the contribution of some VPLs (purple arrows) is greater than a user-defined threshold b and thus bounded. To compensate for the energy loss, Kollig and Keller [2006] propose shoot a ray to create a compensation vertex $\mathbf{x}_2$. If $\mathbf{x}_2$ is inside the original bounding region (marked with orange in (a)) they compute the illumination of $\mathbf{x}_2$ (b) and transport only the amount corresponding to the removed energy using the residual operator $\hat{G}_r$. As bounding occurs also at $\mathbf{x}_2$, the technique is recursive and the path is terminated only when the new vertex is outside the region where bounding occurred (c).

3.3.3 Bias Compensation using Local Lights

Similarly to the previously mentioned technique, Davidovič et al. [2010] separate light transport into the bounded, global component and local, residual component. The global component accounts for long-distance light transport, while the local component corresponds to short-range interreflections and indirect glossy highlights. The authors take advantage of the specific structure that the individual components exhibit, and design a solution tailored for each of them independently. Specifically, they handle as much energy as possible in the global component leaving only the local interreflections for the local component. This approach turns out to be more efficient than a general global illumination solution.

The local component is handled by "local" VPLs, which are distributed by tracing paths from the camera. Since the local VPLs are designed to calculate localized transport in regions with concave geometry, each local VPL contributes only to a small tile of pixels around the pixel, through which the path generating the local VPL was traced. In addition, the authors avoid occlusion tests assuming full visibility between shading points and local VPLs. This approximation, which is the main reason for the high efficiency in glossy scenes, is made possible by capturing most of the energy (and indirect shadows) in the global component, and leaving just the localized, but visually still important transport to the local component. The complete global illumination solution is obtained by summing both components.

3.3.4 Avoiding the Singularity: Virtual Spherical Lights

Rather than compensating for the bias due to the bounded light transport, Hašan et al. [2009] try to avoid the singularity in the first place. They notice that the intensity of bright splotches can be exacerbated by glossy BRDFs, and the compensation can thus become arbitrarily expensive.

This happens when the shading point and the VPL are aligned such that either of the two BRDF terms (or both) have a large value in that direction. It is worth noting that while the artifacts due to the geometry term are generally localized (occurring mostly in corners), the latter can happen across large distances if the materials in the scene are sufficiently glossy.

The authors first introduce the concept of a photon light: a point light that distributes its energy over surfaces within a surrounding spherical region of a certain radius. The name was inspired by the connection to photon mapping, where each photon contributes its energy to nearby surfaces. In order to efficiently evaluate the contribution of a photon light, they use the visibility of the original VPL for all points inside the photon light and also assume that surfaces around the VPL are planar. This yields a new lighting primitive called the *virtual spherical light* (VSL). The advantage of VSLs is that they replace the point-to-point evaluation, which is the source of the $(1/d^2)$-singularity, by an integration over the solid angle subtended by the spherical light. Additionally, the integration averages the product of the two BRDFs over the solid angle, and thus effectively avoids high intensity splotches due to sharp BRDFs.

The concept of a photon light introduces bias, which is similar to the systematic error of photon mapping with final gathering. However, unlike the straightforward bounding, it preserves the energy by redistributing it spatially over nearby surfaces. VSLs can also be combined with scalable many-light methods, e.g. Lightcuts [Walter et al. 2005] or Matrix Row-Column Sampling [Hašan et al. 2007] that are described in the following section.

3.4   Scalability

In the preceding sections, we have discussed how VPLs are generated, represented, and evaluated. In this section, we elaborate more on the last aspect in the context of truly many VPLs. This is often critical as the accuracy of many light methods strongly depends on the number of VPLs used. With only few VPLs, the illumination can be reconstructed only coarsely. While this may be sufficient for small scenes or in real-time applications, generating high-quality renderings in complex scenes requires capturing many detailed, highly localized, indirect illumination effects such as glossy highlights, indirect shadows, and proximity color bleeding. Accurately simulating these effects may require thousands or millions of VPLs. Since the effects of individual VPLs would often be imperceptible, a linear, brute force evaluation of Equation (3.6) for millions of VPLs would be prohibitively expensive and inefficient. Instead, one would prefer an accurate but approximate evaluation that requires far less computation. We will call algorithms *scalable* if their cost increases slowly, or sub-linearly, with the number of VPLs used.

Alternatively, increasing scalability can be viewed as variance reduction. If an algorithm handles a million VPLs while only spending the resources for few hundreds, this is equivalent to decreasing the noise in the estimate while holding the amount of computation fixed. The main Monte Carlo variance reduction techniques are stratification, adaptive sampling, and importance sampling. The scalable algorithms discussed below are based on carefully combining stratification and adaptivity, or caching of importance.
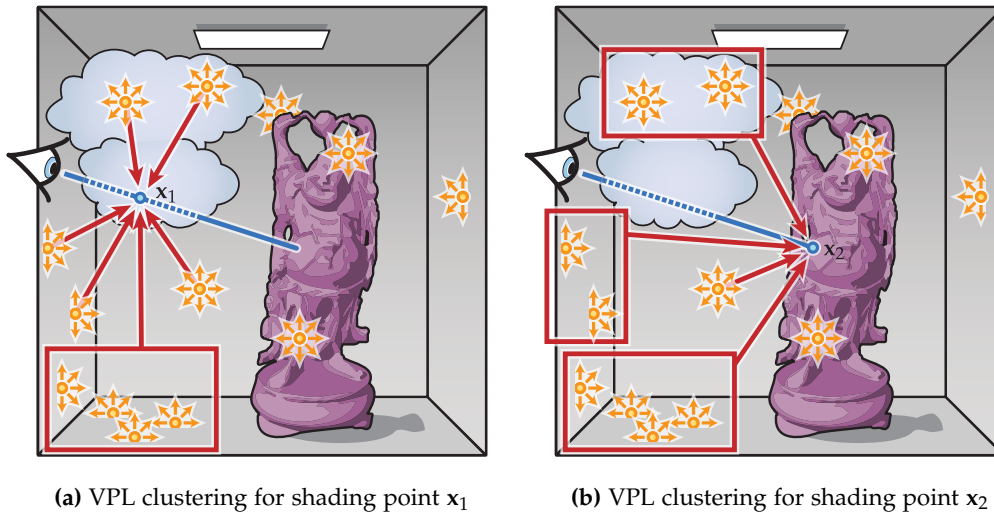
(a) VPL clustering for shading point x_1 (b) VPL clustering for shading point x_2

Figure 3.6: Scalable many-light algorithms can achieve sub-linear cost when evaluating VPL lighting. They cluster VPLs adaptively to adjust the accuracy of the many-light evaluation. For example, a group of distant VPLs might be replaced with a single brighter representative.

3.4.1   VPL Clustering

The methods discussed in this section exploit a common insight: within a large set of VPLs, each VPL does not contribute equally. Many VPLs have low importance because they contribute very little to a region of interest; for example if the VPLs are far away or occluded. However, typically a small number of VPLs are very important, such as VPLs that contribute to a glossy highlight, and these must be handled accurately. A general, scalable algorithm tries to exploit this non-uniform VPL importance to reduce computation. It seeks to identify and evaluate all of the most important VPLs while only sparsely evaluating the unimportant ones.

While the algorithms discussed below differ in how they estimate importance and select a set of VPLs to evaluate, they all use the same framework. Each algorithm clusters similar VPLs together. They choose a clustering that places unimportant VPLs in large clusters and important VPLs in smaller clusters, see Figure 3.6 for an illustration. It is then assumed that VPLs within a cluster are sufficiently similar that their aggregate effect can be approximated by evaluating just a single, brighter, representative VPL. If the representative is chosen randomly from the VPLs within the cluster and its power scaled appropriately, the sum of these representative approximations is equivalent to a stratified, Monte Carlo evaluation of the path integral, where only a single sample is drawn from the domain of each cluster. If these algorithms can find a set of clusters—typically called a *cut*—that is much smaller than the number of all VPLs, the cut approximation becomes a scalable alternative to brute force evaluation.

We will discuss several techniques based on constructing these cuts [Hašan et al. 2007; 2008, Ou and Pellacini 2011, Walter et al. 2006; 2005; 2012] and one more method [Georgiev et al. 2012] that chooses the most relevant VPLs for a shading point rather than clustering them.

Lightcuts

Lightcuts [Walter et al. 2005] was the first practical, scalable many-light method. At each receiver point where illumination needs to be computed, lightcuts generates a customized cut based on analytic per-cluster error bounds and a perceptual metric. As a first pass the VPLs are first organized into a binary tree based on spatial and directional similarity. To select a cut for a receiver, we start with a trivial, coarse clustering, such as putting all the lights in a single cluster. This corresponds to a cut consisting of only the root node of the light tree. Then we iteratively select the cluster in the current cut with the highest error bound and refine it replacing it by its children in the light tree. This process is repeated until the error bounds for all clusters in the cut are below a perceptual-based threshold. The size of the cut is typically only weakly dependent on the number of VPLs, resulting in very large speed-ups compared to a full evaluation as the number of VPLs grows. The use of analytic error bounds also guarantees that the most important VPLs are always found and evaluated, making the estimation robust. Arbree et al. [2008] extend the lightcuts method for sub-surface scattering. Davidovič et al. [2012] describe a progressive, GPU-friendly variant of lightcuts.

Multidimensional lightcuts [Walter et al. 2006] extends the domain of clusters and cuts to include receiving points as well as lights, to achieve scalable performance across a much wider range of effects. When computing a pixel, we really want the average illumination over a region rather than its value at individual receiving points. For example, the region may extend over an image space for anti-aliasing, over the aperture for depth of field, over time for motion blur, and spatially for effects such a volume rendering. When rendering, these regions are typically converted to many receiver points using sampling, but separately evaluating a cut for each point is inefficient as they often have very different sensitivities to the lights and illumination accuracy requirements. Instead, multidimensional lightcuts builds a hierarchy over all light-receiver point pairs for a pixel and its cut is a partition of this much larger point-pair space. To make this feasible, the point pair hierarchy, called the product graph, is implicitly represented as the Cartesian product of a light tree and a receiver tree, which is also called the gather tree. The cut selection process is similar to that in the lightcuts method in that it uses analytic per cluster error bounds and refines the cut until a perceptual threshold is met. One major difference is that cluster refinement can now choose between light or receiver refinement at each step. Overall this technique greatly reduces the rendering cost when computing effects that require both many receiver points per pixel and large numbers of VPLs.

Bidirectional lightcuts [Walter et al. 2012] extends the previous approach to handle even more effects including glossy reflections, subsurface scattering, and short-range indirect illumination. While its goals are similar to the bias compensation methods mentioned previously, it functions by adding additional receiver points per pixel.

Matrix Row-Column Sampling

In contrast to lightcuts, which generates a cut per receiver point, *matrix row-column sampling* (MRCS) [Hašan et al. 2007] computes a single, global cut for the whole image. Because the cut generation is amortized over the whole frame, this approach has two advantages compared to lightcuts.

1. As a measure of VPL importance, it replaces the error bounds, which may be expensive

or unknown for general materials, with sparsely sampled direct estimates of each VPL's image contribution. This allows MRCS to easily add material and light types (including VSLs), and to include visibility information in the cut selection algorithm.

2. The VPL importance and the VPL contributions can both be estimated by using shadow maps. By using graphics hardware to accelerate their calculation, MRCS can achieve very fast, low-noise rendering.

Of course, there are some disadvantages: because the cut is computed once from sparsely sampled data, MRCS is less adaptive than lightcuts, and may sometimes miss small features of the VPL illumination, such as glossy highlights that affect only few pixels.

In order to compute the global cut, MRCS models the VPL evaluation problem as a large matrix **M**. The rows of **M** represent the surface points visible through each pixel and the columns of **M** represent VPLs. Each entry $\mathbf{M}(i, j)$ represents the fractional energy of the j-th VPL that reaches the camera through the i-th pixel. The key insight of MRCS is that **M** is usually a highly structured, often low-rank, matrix and can be well approximated by the reduced matrix **R** that subsamples the elements of **M**. The MRCS consists of computing **R**, using it to compute a cut and then using that cut to approximate the original matrix **M**.

The MRCS approach has also been extended to render full animations [Hašan et al. 2008], concatenating the matrices of the animation frames into a 3D array (tensor), and exploiting temporal coherence to drive the number of required cluster representatives even lower. Davidovič et al. [2010] propose a modification of MRCS called visibility clustering.

LightSlice

The *lightslice* [Ou and Pellacini 2011] algorithm combines the idea of locally adapted cuts from lightcuts with the global optimization advantages of MRCS. The authors of lightslice noticed that, while lightcuts can capture detailed illumination effects by recomputing a cut at each receiver point, MRCS demonstrated that many of these cut calculations waste effort recomputing a shared set of global VPL clusters. The lightslice algorithm improves performance by identifying these shared global clusters once and then reusing them as a starting point for local per-slice cluster refinements. The lightslice algorithm works as follows:

1. Generate all the receiver points for an image and cluster them based on their geometric proximity into groups called *slices*.

2. Select a representative receiver point from each slice and then run MRCS on the set of all slice representatives. This forms both an initial, global cut for all slices and a reduced matrix **R** describing the light transport of the slice representatives.

3. For each slice i, restrict **R** to a smaller matrix $\mathbf{R}^i$ that contains only the row for slice i and the rows from other nearby slices. Iteratively refine the global clustering for i by using $\mathbf{R}^i$ to identify and split high-cost, local clusters to generated localized cuts for each slice.

By using localized per-slice cuts, lightslice is able to reduce the average cut size needed compared to a single global cut while also reducing the chance that locally important VPLs will be missed due to the sparse sampling. Also, by reusing an initial global cut as the local starting point, it significantly reduces the cost of cut selection.

3.4.2 VPL Importance Sampling

Georgiev et al. [2012] propose a different approach to improve scalability: they choose the most relevant VPLs for any given position in the scene based on cached importance. In a preprocess, the contribution of all VPLs is computed at a number of locations in the scene, and cached. When calculating VPL contributions during rendering, the cached contributions at a few nearby locations are used as a discrete probability distribution from which the most relevant VPLs are sampled randomly.

All the aforementioned scalable approaches have greatly increased the effective number of VPLs that can be used in many-light methods, and thus increased the achievable accuracy and image quality. In the next section, we review techniques from the other end of the application spectrum that puts more emphasis on interactivity and real-time use.

3.5 Interactive and Real-Time Applications

The obvious challenge in interactive and real-time rendering, compared to offline methods, is the tight time budget. This expectedly restricts the number of virtual lights that can be created and used (typically several hundreds to thousands), and also the types of materials that can be faithfully rendered under such constraints. Conceptually, the main difference resides in the computation of visibility, where rasterization is typically used instead of ray casting for VPL generation and shadowing. The following sections address these aspects and also briefly touch on temporal stability, which is crucial for plausible, interactive animations.

3.5.1 Rapid Generation of VPLs

In typical real-time scenarios, the render times are dominated by shading and shadowing costs, and less by VPL generation. Since the VPL generation is relatively cheap and only a small number of VPLs can be handled, any spatial indexing structure—even if unoptimized or suboptimally built—is sufficient for tracing the few light paths, and should be used if available.

VPLs can also be created using rasterization. Similar to shadow maps, *reflective shadow maps* [Dachsbacher and Stamminger 2005] (RSMs) render the scene from a primary light source to capture directly lit surfaces. In addition to depth, RSMs store also the position, normal, and reflected flux. Every pixel, or a random subset of pixels, can be thus interpreted as a small light source and serve as a VPL. A similar idea has previously been used to create virtual dipole light sources for rendering translucent objects [Dachsbacher and Stamminger 2003]. In the original RSM paper, the contribution of VPLs was accumulated for a low-resolution image and refined at edges during upsampling. Dachsbacher and Stamminger [2006] later proposed to accumulate the lights' contributions using deferred shading and with bounded regions of influence; they also introduced importance sampling of the underlying RSM.

Ritschel et al. [2011] propose to choose VPLs from RSMs by estimating their contribution to the image, and thus obtaining a probability factor to importance sample the RSM. In both cases, longer light paths can be obtained by recursively creating further RSMs computed for the previously generated VPLs (used in [Ritschel et al. 2008]). Note that various approaches for casting rays using rasterization or based on voxelization exist and can also be used to trace light paths.

3.5.2   VPL Lighting for Interactive Applications

Although programmable graphics hardware is able to shade from virtually arbitrary many light sources, many-light rendering is almost exclusively accompanied by deferred shading techniques [Deering et al. 1988, Saito and Takahashi 1990]. Bounding the regions of influence of VPLs (as in [Dachsbacher and Stamminger 2006]) to speed up shading is often not the preferred solution as it removes light transported over larger distances. Instead, the shading signal is usually subsampled and subsequently interpolated. To this end, most of the techniques employ interleaved sampling [Keller and Heidrich 2001], which has been first used for many-light rendering by Wald et al. [2002]: shading of a single pixel is not performed using all VPLs, but instead each pixel is lit only by a disjoint subset of VPLs. The reasoning is that neighboring pixels often represent nearby and similarly oriented geometry and their shading would thus be similar as well. The interleaved shading is then transferred across neighboring pixels in a post-processing step using an edge-aware image filter. Segovia et al. [2006b] describe a GPU-friendly implementation of interleaved sampling where the deferred shading buffers are reorganized such that pixels lit by the same subset of VPLs are stored in the same tiled sub-buffer. Interleaved sampling greatly speeds up the rendering; however, it is prone to aliasing artifacts with highly detailed geometry and normal mapping, and problematic with glossy BRDFs.

Nichols and Wyman [2009, 2010] propose a hierarchical shading technique based on the observation that indirect illumination in regions with smooth surfaces varies slowly, while geometric detail requires more shading evaluation. Their technique makes use of a min-max mipmap of the depth buffer to detect discontinuities. Indirect illumination is then computed in multi-resolution deferred shading buffers where smooth regions (little variation in the min-max mipmaps) are shaded in low-resolution buffers, and detailed regions in high-resolution buffers. The resolution pyramid is finally combined to the final image using an adapted, bilinear interpolation technique. Note that interleaved sampling approaches are orthogonal to shadow computation from VPLs, while multi-resolution splatting assumes a smooth shading signal (i.e., it does not detect shadow boundaries and thus blurs across them).

Further acceleration of the shading computation with a large number of light sources can be achieved by *tiled shading*: the image plane is subdivided into tiles, and for each tile a list of light sources potentially affecting the visible surfaces is built. Each tile can then be processed independently without evaluating all light sources [Olsson and Assarsson 2011]. Olsson et al. [2012] extended this idea and cluster light sources by their tile and depth.

3.5.3   Visibility Computation in Interactive Rendering

Whenever we compute shading from VPLs, we also have to determine whether the shading point is actually lit or occluded. To this end, almost all interactive methods compute the (hemi-)spherical visibility per VPL before shading. The original instant radiosity [Keller 1997] employed a variant of shadow volumes, but almost all later implementations rely on shadow mapping as this technique is robust, flexible, and its main disadvantage – jagged shadow edges – is not crucial when the contribution of hundreds of VPLs is accumulated.

Computing the per-VPL visibility, or shadow map, is often the most time-consuming part of interactive many-light implementations. One way to speed up this step is to avoid repetitive computation and exploit temporal coherence. Laine et al. [2007] propose to keep VPLs for as

long as they contribute to the image, and invalidate and reposition only few of them per frame. Their method maintains VPLs for single-bounce indirect illumination only (a limitation which could be relaxed when computing VPLs with ray casting), and it is restricted to static scenes, or indirect light from static on dynamic geometry only.

A different strategy is to compute visibility less accurately, which has been shown to be sufficient for indirect illumination in many cases [Yu et al. 2009]. Ritschel et al. [2008] compute low-resolution, low-quality *imperfect shadow maps* (ISM): hundreds to thousands of shadow maps are rendered in parallel from a point representation of the scene geometry. Since only a few thousand point samples are used per each shadow map, the resulting maps contain holes and need to be filled using an image-space heuristic. The point representation is precomputed, but can be deformed with the scene to support animations. In a follow-up work, Ritschel et al. [2011] describe how these point sets can be chosen in a view-adaptive manner, improving the quality of the shadow maps. Holländer et al. [2011] also address the many-view rendering problem and present an incremental and GPU-friendly LoD algorithm, which can be used to compute shadow maps. *Micro-rendering* [Ritschel et al. 2009a] handles visibility computation using a point hierarchy and massively-parallel hybrid rasterization-raycasting technique to render water-tight hemispherical images. It also supports warping of the hemisphere, as it was originally designed for fast final gathering for both diffuse and glossy surfaces.

Recently, another approximate representation gained much interest: any scene geometry can be voxelized allowing for simple ray marching to compute visibility. Voxelization itself is a large research field, see Crassin's PhD thesis [Crassin 2011] for an exhaustive overview.

3.5.4   Improving Quality and Temporal Stability

A crucial factor determining the quality of rendered images is the number of used virtual lights: while a low number might be sufficient for (mostly) diffuse scenes, significantly more is necessary for scenes with glossy surfaces [Křivánek et al. 2010]. Moreover, VPLs are often generated using random walks and even if the same random numbers are used, the generated VPLs can have different locations or contributions if the scene changes. This leads to distracting temporal flickering. Obviously the number of VPLs cannot be increased arbitrarily for interactive scenarios and thus special techniques had to be developed.

One possible solution, orthogonal to incrementally updating VPLs [Laine et al. 2007], is to consider a VPL not as a point light, but instead as an area light that represents indirect illumination from a certain surface area. A straightforward approach to estimate the represented area is to initially create more VPLs and cluster them before shading. The size of the cluster serves as an estimate of the surface area, and this yields, together with the accumulated contribution of the VPLs, a *virtual area light* [Dong et al. 2009]. Visibility from area lights can efficiently be evaluated using soft shadow methods, e.g. [Annen et al. 2008]. When using RSMs, clustering can also be computed directly in image space of the RSM [Prutkin et al. 2012].

Approximate Bias Compensation for Rendering Scenes with Media

Truth is much too complicated to allow anything but approximations.

— JOHN VON NEUMANN (1903–1957)

Participating media cause a wide variety of scattering effects that immensely contribute to the realism of virtual scenes. In landscape sceneries, the appearance of haze, clouds, or smoke is caused by light that is scattered multiple times due to interactions with small particles in the air. Since scattering occurs "everywhere" and continues until the light is absorbed or leaves the medium, simulating such transport is very costly. A common simplification is to consider transport that involves a single scattering event only. This already creates impressive volumetric effects, such as light shafts piercing through tree canopies, but the simplification is only valid for participating media with low albedo (i.e. relatively high absorption). In contrast, multiple scattering is crucial for the appearance of highly scattering media, such as clouds.

In this chapter, we present a novel method for high-quality, many-light rendering of scenes with participating media. Our technique builds upon the algorithm proposed by Raab et al. [2008]. The authors take the original instant radiosity [Keller 1997] extended to compensate for bias due to bounding [Kollig and Keller 2006] and demonstrate that such algorithm is capable of handling scenes with participating media. The major difference here is that VPLs are generated also in volumes and, in addition to illuminating surfaces, they illuminate the medium and approximate thus effects of multiple scattering.

As described in Chapter 3, many-light algorithms feature several advantages: (1) they require hardly any pre-processing; distributing VPLs by the means of random walks is a matter of milliseconds, even for thousands of VPLs. (2) Rendering with VPLs is highly scalable and covers wide application spectrum from interactive previews to high quality rendering with millions of VPLs. (3) VPLs can rely on (deep) shadow mapping to quickly resolve visibility without the need for additional, complicated structures (e.g. photon maps). All of these advantages hold when extending the algorithm to account for participating media. The extra ingredient, added in [Raab et al. 2008] on top of the original technique, is the *combined single scattering* from primary light sources and VPLs, which adds the multiple scattering component.

As in the case of surfaces, illuminating media with point lights is prone to a singularity that creates bright splotches in rendered images (see Figure 4.1.a). On surfaces, the product of the two cosine terms in the numerator of $\hat{G}$ reduces the impact of the squared distance in the denominator. The product accounts for the mutual orientation of the two surface points. In media, this product is no longer present and $\hat{G}$ thus reduces to the inverse squared distance only. As a

(a) Unbounded VPLs
49 minutes

(b) Bounded VPLs
49 minutes

(c) Bias compensation
8 hours

(d) Raab et al [2008]
8 hours 49 minutes
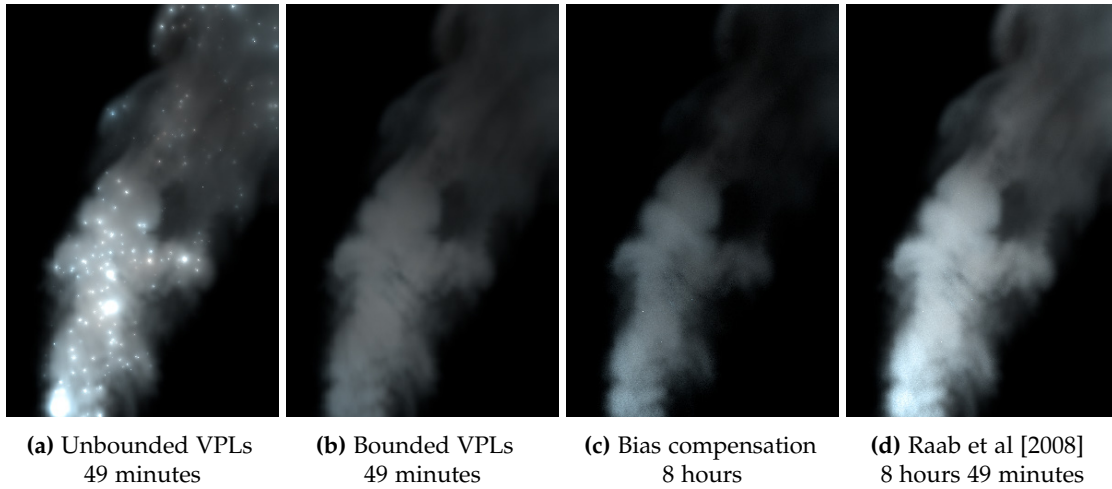
Figure 4.1: Rendering of a rising smoke with (a) unbounded contribution (i.e. with $\hat{G}$), and (b) bounded contribution (i.e. with $\hat{G}_b$). In (c) we show the bias compensation as proposed by Kollig and Keller [2006] and extended by Raab et al. [2008] to participating media. Adding the compensation term (c) to the bounded solution (b) yields unbiased results (d); however, at a significant additional computation cost.

result, the artifacts due to the singularity can be even more prominent and we need to bound the geometry term more aggressively to remove the artifacts (see Figure 4.1.b). The bias compensation by Raab et al. [2008] recovers all the missing energy (see Figure 4.1.c), unfortunately, at a significant computational cost and thus heavily decreasing the performance of the algorithm.

In this chapter, we study bias compensation in the presence of participating media and propose the *approximate bias compensation* (ABC) [Engelhardt et al. 2012], which, unlike the compensation by Raab et al. [2008], increases the entire rendering cost by only a fraction of the bounded solution. Our goal is to create a fast, high quality renderer. We seek an approach that can be easily accelerated using GPUs taking advantage of their strong capabilities, such as fast creation of shadow maps, to efficiently resolve visibility between shading points and VPLs. In the following, we present:

- an analysis of bias compensation, which quantifies the amount of energy recovered with each additional bounce of the residual transport;

- a comparison of several sampling strategies for estimating the residual transport;

- two approximations that make the integration more efficient and GPU friendly; and,

- a progressive, GPU-based renderer for synthesizing images with heterogeneous media.

The rest of the chapter is structured as follows: we review the relevant previous work concerned with rendering participating media in the next section. In Section 4.2, we investigate and analyze different integration strategies whose implementation is described in Section 4.3. Then we present results that can be obtained with our technique in Section 4.4, and finally, we discuss the limitations and possible improvements in Section 4.5.

4.1   Previous Work

Since the previous work related to many-light rendering was discussed already in Chapter 3, we devote this section to algorithms that are primarily targeting scenes with participating media. We also omit references to techniques that are used as building blocks (e.g. for sampling the free path or transmittance) and refer the reader back to Sections 2.6.8 and 2.6.7.

Path Tracing Approaches.   Early rendering methods aiming at the radiative transfer in participating media [Chandrasekhar 1960] were based on stochastic techniques (e.g. [Kajiya and Von Herzen 1984]) and/or finite element methods (e.g. [Rushmeier and Torrance 1987]); see [Rushmeier 1995] for an overview of the pioneering work. Many of the follow-up publications were then inspired by the path integral framework [Feynman and Hibbs 1965] and sampling strategies developed for neutron transport simulations [Gelbard et al. 1966, Kalos 1963, Spanier 1966, Spanier and Gelbard 1969, Steinberg and Kalos 1971]. Such examples include bidirectional [Lafortune and Willems 1996] and Metropolis [Pauly et al. 2000] sampling, as well as techniques building on the path integral formulation [Premože et al. 2003; 2004] for rendering in the presence of participating media. These approaches are general, compute unbiased solutions, and can easily handle arbitrary phase functions. The major drawback is the slow convergence, especially in heterogeneous media, due to the costly sampling of free paths. Some algorithms thus divide the integration domain into smaller regions with similar properties [Szirmay-Kalos et al. 2011, Yue et al. 2010] allowing the free path sampling to proceed more efficiently.

Caching and Density Estimation Approaches.   Several methods address the high cost of pure Monte Carlo approaches by reusing or correlating the estimates. This can be achieved by caching irradiance [Ward et al. 1988] or radiance samples [Křivánek et al. 2005, Scherzer et al. 2012], which has also been successfully extended to participating media [Jarosz et al. 2008a]. Similarly, photon mapping [Jensen 1996] also samples and stores the distribution of light in the scene (in the form of a photon map), but instead of interpolating the samples, it performs density estimation to reconstruct the local flux density. Jensen and Christensen [1998] generalized photon mapping to participating media. Jarosz et al. [2008b] improved their approach by finding all photons along the length of each camera ray in one *beam query*, and later applied similar concept to light paths turning entire path segments into *photon beams* [Jarosz et al. 2011a;b]. The benefits of using lines (instead of points) to represent and to query the underlying quantity was concurrently explored by Sun et al. [2010], who simulate single scattering and caustics by finding nearly intersecting camera and light paths. Boudet et al. [2005] calculate the density estimation in volumes via photon splatting and propose a GPU friendly implementation.

Several extensions seek to decrease the error caused by density estimation [Havran et al. 2005, Herzog et al. 2007, Lastra et al. 2002, Moon and Marschner 2006], but ultimately, some amount of bias remains. Progressive photon mapping [Hachisuka and Jensen 2009, Hachisuka et al. 2008b, Knaus and Zwicker 2011] and progressive photon beams [Jarosz et al. 2011b] eliminate this error gradually by shooting and discarding photons in batches, while progressively refining radiance statistics to converge to the correct solution in the limit. In spite of all the aforementioned improvements, density estimation approaches are still not very suitable for interactive rendering, mainly due to splotchy artifacts on surfaces and in media, which go away only with large numbers of photon points or photon beams.

Single Scattering.   In media with low density or low albedo, most of the visible effects are due to single light bounce. In such cases, one can simplify the light transport and tailor the rendering algorithm specifically to single scattering only. The integration can be then carried out (semi-) analytically [Pegoraro and Parker 2009, Pegoraro et al. 2009; 2010; 2011, Sun et al. 2005]. Although closed form solutions are generally preferred over numerical recipes, the necessity to handle the visibility independently [Biri et al. 2006], precompute a set of tables, or expand the scattering function using Taylor series complicates the implementation and decreases the efficiency of these techniques. Quadrature methods, such as ray marching, perform well in scenes with limited depth especially when restricted to regions where shadowing occurs [Wyman and Ramsey 2008]. Engelhardt and Dachsbacher [2010] observe that the amount of inscattered light changes smoothly along epipolar lines. They detect shadow discontinuities along these lines and place samples more intelligently to reduce the integration error. Baran et al. [2010] further improve the quality via hierarchical integration. Monte Carlo integration of single scattering with a suitable importance sampling [Kulla and Fajardo 2012] can also yield results with low variance. When ported on graphics hardware, these techniques are capable of real-time performance.

Multiple Scattering.   When the medium has high density and albedo, we can presume that each photon gets scattered many times before reaching the camera. In such cases, one can efficiently model the process of scattering using the diffusion theory [Fick 1855, Stam 1995]. This approach is used intensively for rendering optically thick materials, such as marble, wax, or skin. Jensen et al. [2001] introduced the *diffusion dipole* model, which analytically calculates the amount of light transmitted between two points on a planar semi-infinite medium. To further accelerate the computation, Jensen and Buhler [2002] proposed a hierarchical integration scheme. In the case of human skin, the response of the diffusion dipole can be well approximated with a set of normal distributions, which can provide real-time performance when implemented on the GPU [d'Eon et al. 2007, Jimenez et al. 2009]. Donner and Jensen [2005] extended the model to a *multipole*, and d'Eon and Irving [2011] suggested to quantize the diffusion, both obtaining better quality than the original dipole. The major drawback of most diffusion-based approaches are the strong assumptions about the geometry (e.g. planarity, semi-infiniteness, etc.). Recently, Habel et al. [2013] proposed to combine diffusion with Monte Carlo sampling to better account for the geometry, while retaining the benefits of the fast diffusion approximation.

A comprehensive overview of literature devoted to rendering scenes with participating media can be found in [Cerezo et al. 2005].

4.2   Residual Light Transport

In this section, we study the energy recovered using the bias compensation technique by Raab et al. [2008] (described in Section 3.3.2). Their approach is relatively costly as it requires recursive ray tracing and access to all VPLs at every compensation vertex. We first analyze the underlying Equation (3.11) and then derive several optimizations, sampling strategies, and simplifying assumptions; all leading to a more efficient, approximate bias compensation technique that yields results visually nearly indistinguishable from the ground truth.

(a) Scanline   (b) Pixel intensity   (c) Bias compensation
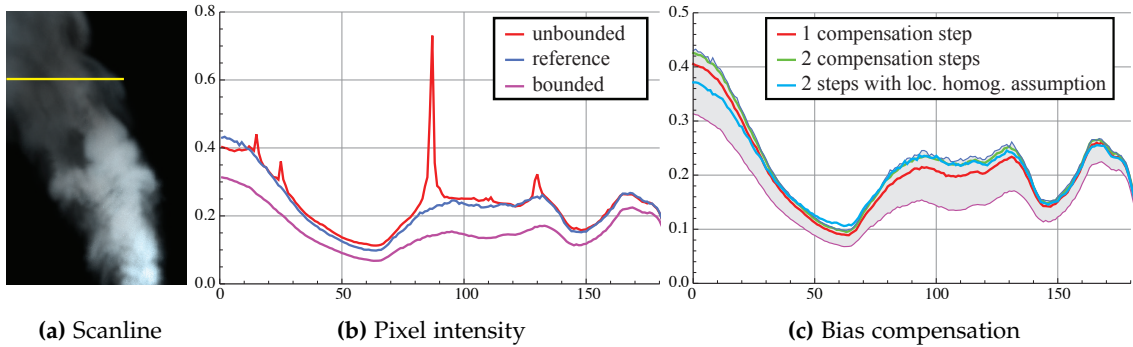
Figure 4.2: For one line (a) from renderings in Figure 4.1 we plot pixel intensity of the unbounded, bounded, and reference solution (b). In (c), we show the amount of energy recovered by one and two compensation steps (the top and the bottom boundary of the band correspond to the reference and the bounded solution, respectively). The blue curve shows energy recovered assuming that the medium is locally homogeneous (see Section 4.2.2).

4.2.1 Limiting Recursion Depth

When computing the residual transpot, Raab et al. [2008] create "compensation" vertices by ray casting. At each compensation vertex, they estimate the illumination using the *bounded* estimator, which gathers light from the original light sources and VPLs. As such, the bias compensation is a recursive process: bounding occurs not only at the shading point, but also at the compensation vertex. However, as the gathered radiance gets convolved with the scattering function and attenuated by transmittance, the amount of light drops exponentially with every bounce of the residual transport. Figure 4.2 shows that one compensation step (i.e. one bounce of residual transport), and then bounding the VPLs' contribution to the compensation vertices, recovers most of the energy and already mimics the behavior of the ground truth curve quite well. It can also be seen that two steps (and then bounding) is visually almost indistinguishable from the ground truth solution.

4.2.2 Assuming Locally Homogeneous Media

In order to create a new "compensation" vertex $\mathbf{y}$ for a shading point $\mathbf{x}$, Raab et al. [2008] choose a random direction ω along which they sample free path length using Woodcock tracking [Woodcock et al. 1965] (see Section 2.6.8). If $\mathbf{y}$ happens to be outside the bounding region, $\hat{G}_r$ will be zero and the compensation vertex will have no contribution. This approach is unbiased, but suffers from high variance. The variance can be reduced by creating more compensation paths; however, as there is no possibility in *heterogeneous* media to limit the importance sampling of transmittance (i.e. the Woodcock tracking) to a maximum distance, a lot of free path samples will be rejected before generating one within the bounding region. We avoid this issue by introducing the assumption that the medium is *locally homogeneous* around $\mathbf{x}$.

One key ingredient of our ABC is that we generate compensation vertices always inside the bounding region around $\mathbf{x}$. The radius d of the spherical bounding region can be computed from the bound b as $d = 1/\sqrt{b}$. Assuming the medium inside the bounding region is homogeneous with extinction coefficient $\overline{\kappa_t}$, the probability density function for sampling a distance t within

| **(a)** Accurate transmittance | **(b)** Locally homogeneous | **(c)** 16× difference |
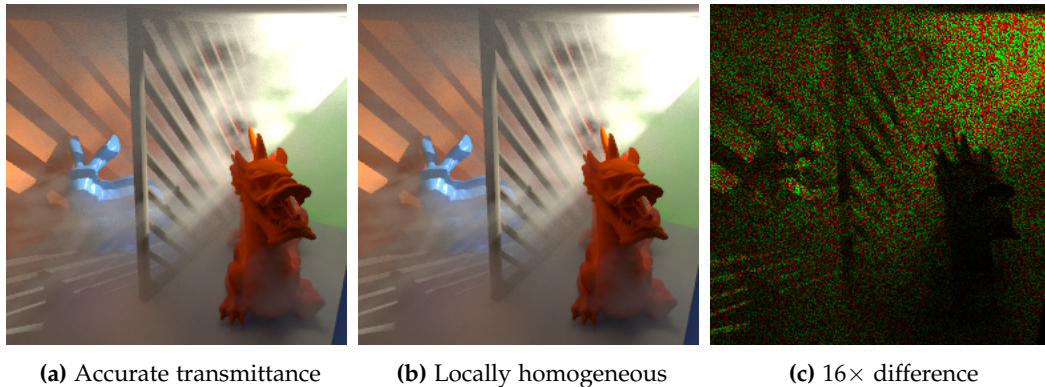
Figure 4.3: Bias compensation with accurately evaluated transmittance (a) and assuming the medium to be locally homogeneous (b). (c) shows a 16× scaled difference (a) - (b); green and red signify positive and negative values, respectively.

the region reads:

$$p(t) = \frac{\overline{\kappa_t} e^{-\overline{\kappa_t} t}}{1 - e^{-\overline{\kappa_t} d}}. \tag{4.1}$$

Inverting the corresponding CDF and solving for the distance yields:

$$t = -\frac{\ln(1 - \xi(1 - e^{-\overline{\kappa_t} d}))}{\overline{\kappa_t}}, \tag{4.2}$$

where $\xi \in \langle 0, 1 \rangle$ is a uniform random number. For the value of $\overline{\kappa_t}$, we use the average extinction coefficient within the bounding region, which can be efficiently obtained from a downsampled version of the medium, if stored e.g. as a 3D texture. Note that if there is a surface intersection along ω occurring closer to $\mathbf{x}$ than t, this intersection becomes the new compensation vertex.

The assumption of local homogeneity does not compromise the results. In fact, it only affects the placement of compensation vertices, and the computation remains unbiased, as long as we correctly evaluate the transmittance $T(\mathbf{x} \leftrightarrow \mathbf{y})$. Nevertheless, we take the assumption one step further, and assume the medium to be homogeneous not only for sampling the compensation vertex, but also for evaluating the transmittance:

$$T(\mathbf{x} \leftrightarrow \mathbf{y}) \approx \overline{T}(\mathbf{x} \leftrightarrow \mathbf{y}) = e^{-\overline{\kappa_t} \|\mathbf{x} - \mathbf{y}\|}. \tag{4.3}$$

Avoiding an exact evaluation of the transmittance (e.g. using costly ray marching or Woodcock tracking) proves useful especially for GPU implementations as all other transmittances can be precomputed before rendering (e.g. using deep shadow maps for the camera and the VPLs). As such, no transmittance evaluations are required during rendering, which often enables higher frame rates.

Although the assumption of local homogeneity can theoretically fail at locations with strongly varying extinction, it yielded results visually indistinguishable from the ground truth (see Figure 4.3) in all our test scenes. Note that these locations could be easily detected if necessary using the gradient of $\kappa_t(\mathbf{x})$.

4.2.3   Integration Strategies

In this section, we analyze different sampling strategies for estimating the residual light transport. Our goal is to find a strategy that strikes a good balance between the number of eye ray samples and compensation vertices. In Figure 4.4, we compare the results at roughly equal rendering times; Figure 4.5 then shows RMSE plots for four strategies that we describe next.

1-to-N Strategy.   Raab et al. [2008] propose to create *one* compensation vertex using free path sampling and illuminate it using *all N* VPLs. This has two implications. First, because of the free path sampling, the vertex may be placed outside the bounding region and the efforts of creating the vertex thus wasted. Furthermore, this leads to high variance since the compensation integral from Equation (3.11) is severely under-sampled (see Figure 4.4). Second, each compensation step requires access to all VPLs and their shadow maps, making the approach heavy on memory and difficult to integrate into progressive renderers.
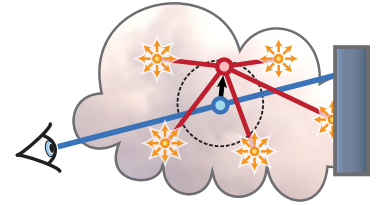
N-to-1 Strategy.   We found that the variance can be significantly reduced by creating more compensation vertices, but connecting each of them to less VPLs. More precisely, we iterate over all VPLs and for each we create a new compensation vertex. Each of the N compensation vertices thus requires access to *one* VPL only. This strategy increases the number of rays to be traced, but despite the slightly longer run-times, it still exhibits lower variance than [Raab et al. 2008] as shown in the equal-time comparison in Figure 4.4.
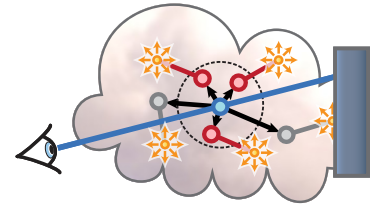
1-to-1 Strategy.   A slightly different approach is to generate a different location along the eye ray for each VPL, for instance w.r.t the expected contribution of each VPL, which further reduces variance. Creating only a single compensation vertex that is connected to the VPL seems to be sufficient and reduces the cost of the compensation. This approach is also GPU friendly, since VPLs can be processed independently: we can generate one VPL, create a shadow map for it, and compute its contribution to all pixels (including the residual transport). The VPL is then discarded and the image computed progressively. At any moment during the rendering, we thus need to store only a single (deep) shadow map in the memory.
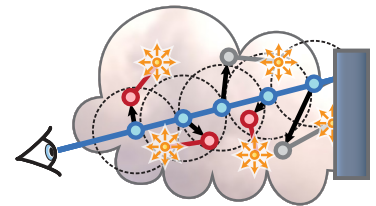
1-to-1 Strategy Assuming Locally Homogeneous Medium. Because of the free path sampling, all the previous strategies generate many compensation vertices outside the bounding region and thus with zero contribution. To avoid this, we would like to create all vertices inside the bounding region. This can be achieved by assuming the medium to be locally homogeneous. Furthermore, it ensures that parallel execution paths do not diverge, which is favorable for GPU implementation.

(a) 1-to-N　　　　**(b)** N-to-1　　　　**(c)** 1-to-1　　　　**(d)** 1-to-1 loc. homog.
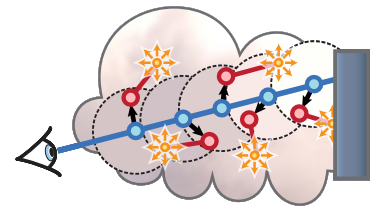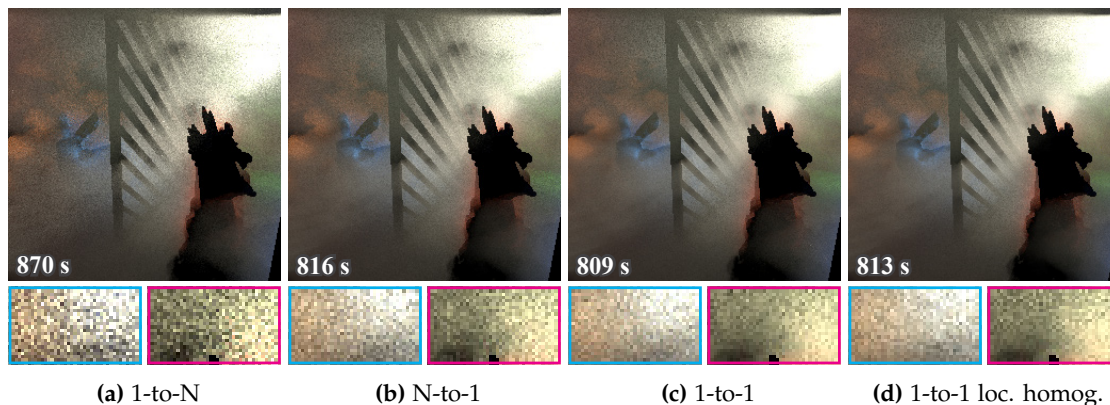
Figure 4.4: Comparison of four different sampling strategies for estimating the residual transport from Figure 4.3 (CPU implementations): (1-to-N) [Raab et al. 2008] creates one compensation vertex for each shading point and connects it to all VPLs. (N-to-1) achieves lower variance by generating more vertices while connecting each of them to one VPL only. Similar, but more GPU-friendly approach is to generate only one vertex connected to a single VPL (1-to-1). By assuming a locally homogeneous medium, we avoid the expensive evaluation of transmittance and ensure that vertices are always created within the bounding region. To achieve roughly equal rendering times, we adjust the number of samples along the eye ray: 3, 1, 115, and 78 for 1-to-N, N-to-1, 1-to-1, and 1-to-1 locally homogeneous, respectively.
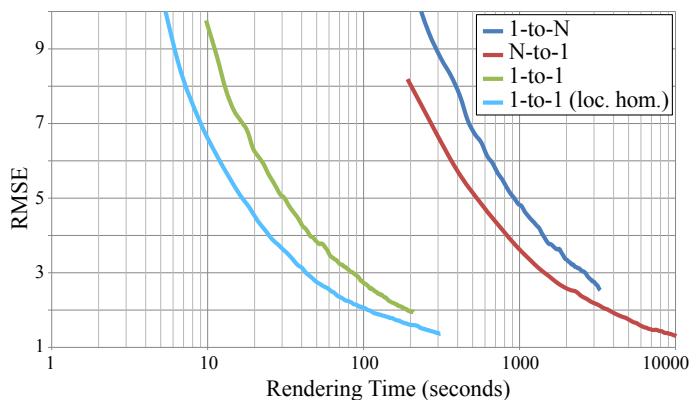
Figure 4.5: RMSE plot for different sampling strategies used in Figure 4.4. The 1-to-1 strategy clearly outperforms other strategies.

4.2.4　Omitting Local Visibility

Computing visibility between point **x** and the compensation vertex **y** is often the performance bottleneck. Clearly, the transport can only be occluded when **x** is close to a surface. Figure 4.6.a depicts a situation when omitting the visibility test can cause artifacts. To asses how often these artifacts appear, and their influence on the resulting image, we set up a series of experiments. Interestingly, it was not easy to produce visible artifacts at all. This can be explained by considering the circumstances that have to coincide to cause them: (1) **x** must be close to a thin opaque object, and (2) the medium must not be too dense otherwise sampling a distance through the opaque object is unlikely. Figure 4.6.b shows one of our test scenes; artifacts become visible only after scaling the brightness by at least two orders of magnitude. Note that a somewhat similar assumption (ignoring visibility on short distances) has also been used for global illumination on surfaces [Arikan et al. 2005, Davidovič et al. 2010].

(a) Illustration   **(b)** Rendering   **(c)** Scaled insets
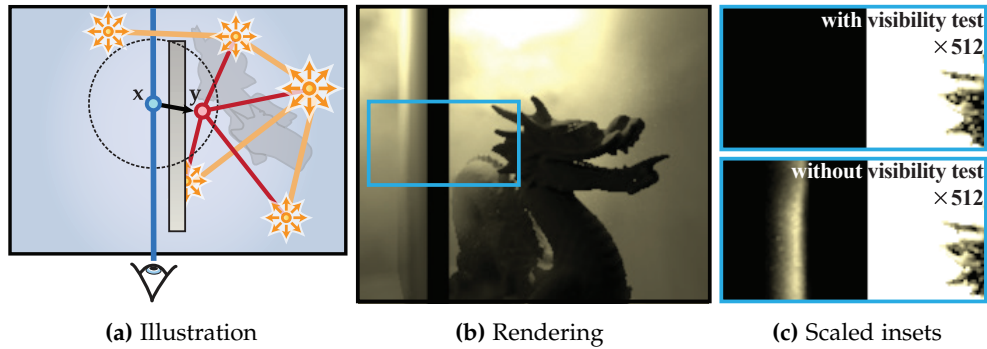
Figure 4.6: A test scene for evaluating the absence of the visibility test. (a) illustrates the transport of interest: bleeding of the residual transport through the wall, (b) shows the rendering. In (c) we provide scaled insets computed with accurate visibility test (top) and without testing the visibility (bottom). Artifacts can be revealed only by tremendous scaling ($\times 512$).
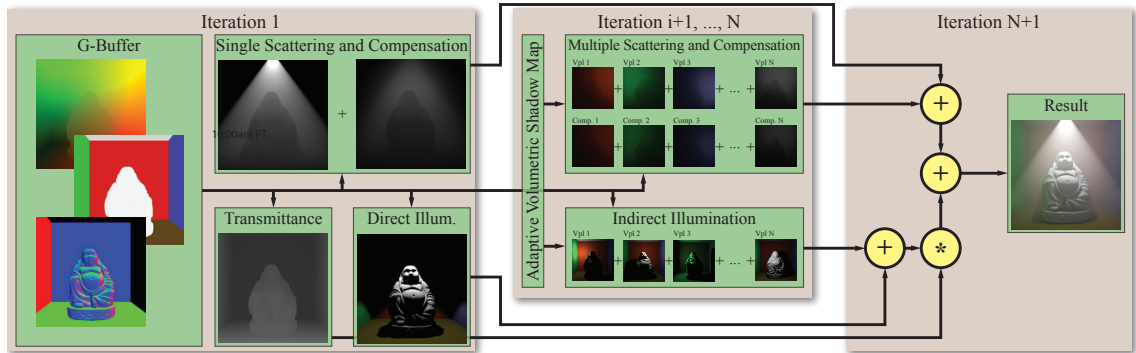
Figure 4.7: The data flow in our progressive GPU renderer. In the first iteration we compute the geometry buffer, single scattering with bias compensation from primary light sources, transmittance, and direct illumination. In each subsequent iteration we compute multiple scattering, indirect illumination, and bias compensation due to a single VPL, and add it to the final result.

4.3 Implementation Details

We integrated our approximate bias compensation technique into a custom offline renderer to evaluate our assumptions (Figures 4.2, 4.3, 4.4, and 4.6) For further acceleration we implemented a progressive GPU renderer using Direct3D 11 (Figures 4.7, 4.8, 4.10, 4.11, and 4.12). Random walks for creating VPLs are always carried out on the CPU using ray tracing. For acceleration, we use a kD-tree built with the surface area heuristic.

In this section, we restrict ourselves to the peculiarities of the GPU implementation, which is outlined in Figure 4.7. We split the computation into evaluating contributions from primary light sources and from VPLs. First, we render a geometry buffer filled with all relevant information such as BRDFs, positions, and normals. Afterwards, we evaluate single scattering and direct illumination with visibility computed using shadow maps (with resolutions of 512^2 up to 4096^2). Transmittance towards light sources is evaluated analytically in case of homogeneous media and numerically in heterogeneous media using ray marching (the offline renderer uses slower but unbiased Woodcock tracking).

VPL Lighting.   After the contribution from primary light sources has been computed, our renderer iterates over all VPLs and accumulates their contribution, one per iteration. For each VPL, we first construct a variant of the adaptive volumetric shadow map [Salvi et al. 2010], and then compute its contribution to all surface and volumetric points seen by the camera. The latter is evaluated using an adaptive ray marching along the camera the rays.

Bias compensation.   We split the compensation integral into two terms: one for compensating from the primary light sources, and the second gathering compensation energy from VPLs. This allows us to evaluate compensation from the primary lights directly in the single scattering shader, which is executed only once in the first iteration. For that we generate a buffer that contains a set of random directions and additional random numbers used to sample the distance along the compensation ray. Computing bias compensation due to VPLs is integrated into a shader that evaluates multiple scattering. Our assumption of locally homogeneous medium and neglecting visibility allow us to create the compensation vertices *always* within the bounding region. This, in contrast to the original bias compensation [Raab et al. 2008], avoids branching and divergent execution paths, substantially accelerating the GPU implementation.

4.4   Results

We evaluated our method using several test scenes with homogeneous and heterogeneous participating media. All timings were recorded using an Intel Core i7 6-core system with 3.2GHz and a GeForce GTX 580 GPU.

Our algorithm can be used for rendering moderately complex scenes with image-based lighting, such as those in Figure 4.8. Both the CRYTEK SPONZA scene (262k triangles, 118k VPLs) and the CITY scene (823k triangles, 108k VPLs) were rendered with a 2-step ABC (i.e. two bounces of residual transport). The shading cost depends on the geometric complexity of the scene and the resolution of the 3D texture storing the heterogeneous participating medium. A detailed analysis of the per-VPL GPU shading cost is shown in Figure 4.9.

Figure 4.10 shows a visual comparison of our approximate bias compensation technique to a reference image computed with unbiased bias compensation Raab et al. [2008] in the participating medium. In both cases, we use 6800 VPLs in the entire scene. The approximate bias compensation recovers most of the lost energy already after the first compensation step.

In Figure 4.11, we compare our algorithm to photon mapping with beam radiance estimate (BRE) by Jarosz et al. [2008b]. In contrast to the BRE, instant radiosity with our compensation technique can be trivially parallelized, thus we used the GPU implementation in the comparison. All images are rendered at 1024^2 pixel resolution. Shooting photons and building the search data structure for BRE took 25 seconds, rendering using beam queries additional 110 seconds (135 seconds in total). The instant radiosity with our 2-step ABC took 125 seconds. Photon mapping still shows the typical artifacts that arise from an insufficient number of photons in the volume, while ABC is nearly indistinguishable from the reference solution, which was computed using the method of Raab et al. [2008] in 31 hours.

As shown in Figure 4.12, our algorithm also supports media with anisotropic phase functions. We rendered the scene with 4320 VPLs and a 2-step ABC varying the *g* parameter of the Henyey-Greenstein phase function. The average shading time per VPL is 16 ms.

(a) CRYTEK SPONZA **(b)** CITY

Figure 4.8: Our approximate bias compensation can be used in complex environments to recover the energy loss due to clamping the contribution of VPLs. In the CRYTEK SPONZA the clamped volumetric and surface illumination was rendered in 39 minutes (using 118k VPLs), while the missing energy was recovered using a two-bounce ABC in only 13 minutes.
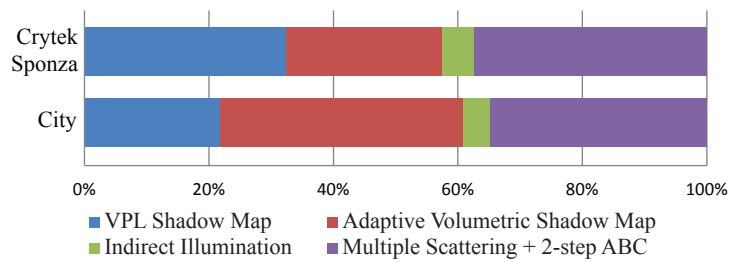
Figure 4.9: Shading time per VPL for the CRYTEK SPONZA and CITY scenes: most of the shading time is used for constructing the shadow maps (about 60%), while indirect (surface) illumination is relatively fast, and multiple scattering with 2-step ABC requires only $35 - 38\%$ of the entire shading cost.

(a) Reference **(b)** Bounded **(c)** 1-step ABC **(d)** 2-step ABC
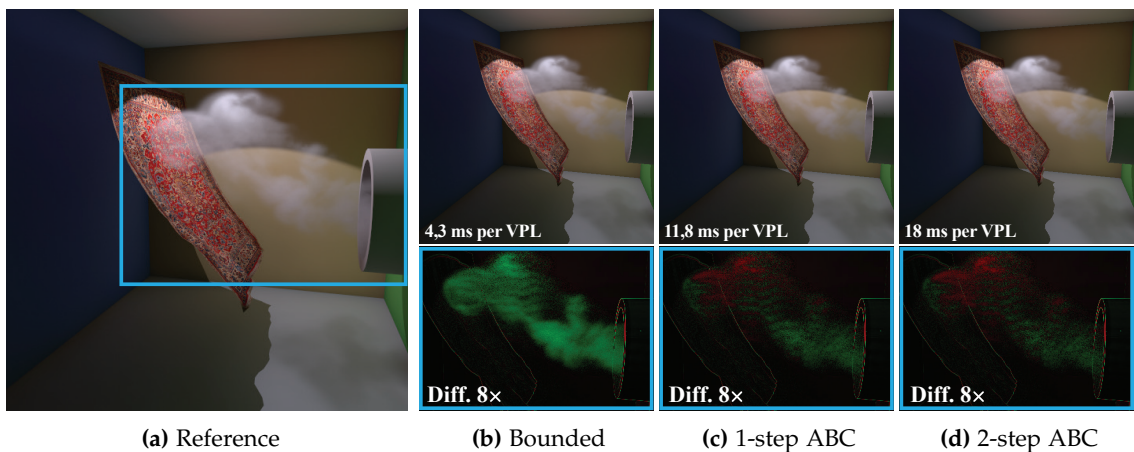
Figure 4.10: A room with a heterogeneous smoke ($\kappa_s = 0.9$, $\kappa_a = 0.001$) rendered with 6800 VPLs. Bounding (b) removes a remarkable amount of energy, which is almost completely recovered using just one ABC step (c). The insets visualize lost energy (green), and overcompensation (red). Differences on the edges are due to different sampling of primary visibility on the CPU and GPU.

(a) BRE [Jarosz et al. 2008b]   (b) Our ABC   (c) Reference
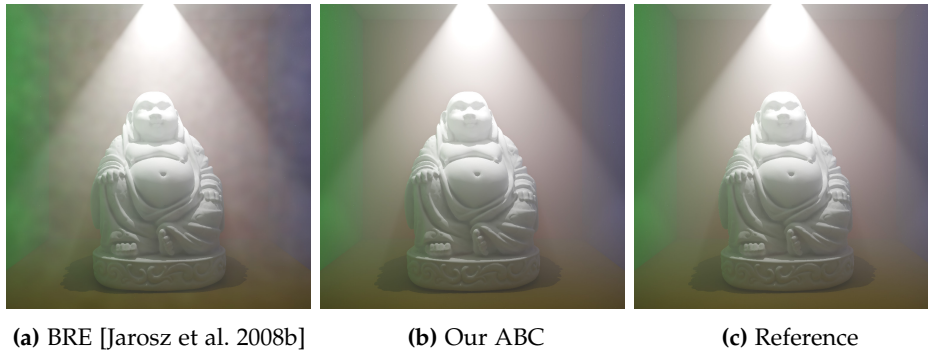
Figure 4.11: An approximate equal-time comparison of beam radiance estimate [Jarosz et al. 2008a] (a) with 1 million volumetric photons and our GPU-accelerated 2-step ABC (b) with 7887 VPLs.

(a) Backward scattering   (b) Isotropic   (c) Forward scattering
$g = -0.7$ $g = 0$ $g = 0.7$
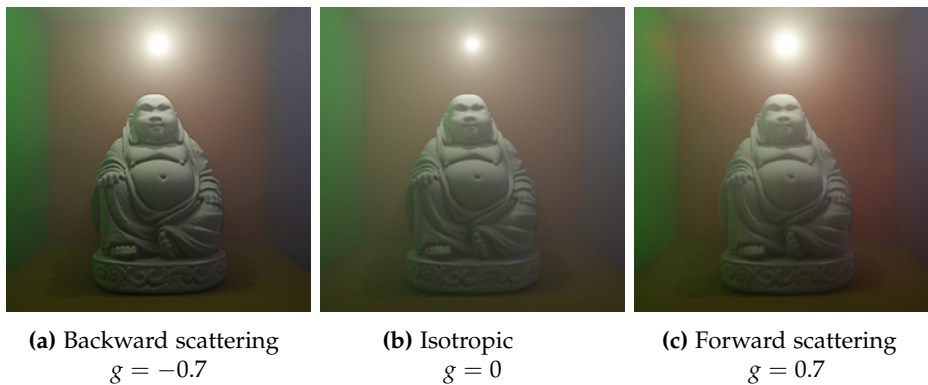
Figure 4.12: The BUDDHA in a homogeneous medium ($\kappa_s = 0.075$, $\kappa_a = 0.001$) with varying g parameter of the Henyey-Greenstein phase function. The scene is lit by a single point light.

4.5   Discussion and Possible Improvements

In this section, we describe our findings from experimenting with our technique, which we believe are important to assess its strengths and limitations, and use in complex scenes.

Scattering Functions.   Similarly to many-light techniques tailored for lighting surfaces, our approach *sub-samples the path space*. The resulting images are typically close to ground truth when scattering functions are isotropic or moderately anisotropic. Our method supports even highly anisotropic phase functions; however, strong forward or backward scattering leads to similar problems that arise when VPL techniques face highly glossy materials. This is because sub-sampling the path space assumes smooth illumination, which is only valid for isotropic and moderately anisotropic scattering. To achieve artifact-free renderings of highly anisotropic media, a large number of VPLs is required. We also observe that more VPLs are necessary for dense and heterogeneous media. Thin media can be rendered with fewer VPLs. For surfaces, the link between the scene geometry and materials and the number of required VPLs has been extensively studied by Křivánek et al. [2010]. Deriving similar dependency on the parameters of the medium would be interesting future work.

Complex Scenes. Scenes with large extent benefit from bi-directional VPL generation (see Section 3.2.2 for an overview) to create enough VPLs that contribute to the image significantly. In our implementation, we use a variant of Georgiev and Slusallek's [2010] method: we create a larger number of paths and keep only those VPLs that contribute the most to the image. This is evaluated by simply considering whether they are close to the camera. This approach is surprisingly well-suited for scenes with participating media, such as those in Figure 4.8.

Animations. Temporal changes to primary light sources or scene geometry inherently change the distribution of VPLs in each frame. This becomes noticeable as flickering if an insufficient number of VPLs is used. Since our method directly depends on the VPL distribution, this effect is carried over. To reduce the temporal artifacts, a sufficient number of VPLs is necessary (several thousand for small scenes). To further improve temporal coherence, the VPLs can be distributed using the same sequence of random numbers in every frame.

Transport between Media and Surfaces. Our approximate bias compensation is primarily tailored for estimating the residual transport in media. While the basic instant radiosity method easily handles transport between surfaces and media, omitting the visibility test, as proposed in Section 4.2.4, restricts our compensation technique to media-to-media, or surface-to-media transport only. As a result, volumetric illumination lacks residual transport from nearby surfaces. This can be avoided by tracing an actual ray and placing the compensation vertex at the intersected surface point, shall there be any before reaching the sampled distance d.

Unbiased Rendering. Some of our observations can also be used to accelerate the original bias compensation technique [Raab et al. 2008]. The 1-to-1 integration strategy proved to be superior over the 1-to-N and more GPU friendly. Assuming the medium to be locally homogeneous only for sampling and evaluating the actual transmittance accurately can also improve the result, since each of the compensation vertices is inside the bounding region and thus likely to deliver residual energy.

4.6 Conclusion

In this chapter, we presented a fast, instant radiosity-based method for rendering global illumination, including multiple scattering, in heterogeneous media. Key to our method is the approximate bias compensation technique that enables rendering images close to ground truth in a fraction of time required by the bias compensation from [Raab et al. 2008]. While the visual impact of our approximations is mostly indistinguishable from the original technique, our approach is more efficient and amenable to GPU acceleration.

Screen-Space Bias Compensation for Surface Illumination

There are two possible outcomes: If the result confirms the hypothesis, then you've made a measurement. If the result is contrary to the hypothesis, then you've made a discovery.

— ENRICO FERMI (1901–1954)

I n the previous chapter, we focused on efficiently solving the radiative transfer equation using many-light algorithms. Most of the speed-up was achieved by cleverly altering the original technique [Kollig and Keller 2006, Raab et al. 2008], while keeping in mind all aspects of the radiative transfer. In this chapter, we revisit the problem but in a simplified scenario considering surface illumination only.

Our goal is to achieve high-quality global illumination at interactive frame rates. Motivated by the steadily increasing computational power of GPUs and by the improvements of the original instant radiosity method, such as imperfect shadow maps [Ritschel et al. 2008], we again build on the many-light approach using virtual point lights. As shown in Chapter 3, these techniques are GPU friendly and capable of achieving high performance. The common downside is however the somewhat poor visual quality of the result, shall it be obtained in real-time. This is due to the restricted number of VPLs, leading to undersampling of the local light transport, and due to a significant amount of bounding that is required to suppress splotchy artifacts. We propose to recover the missing energy in a GPU-friendly manner in screen space: as the compensation essentially recomputes the missing light transport over short distances, we observe that most of the required information about nearby surfaces can be captured in screen space.

Our approach is based on a solid theory obtained from a reformulation of the rendering equation, which is presented in Section 5.2. To further accelerate the compensation, we use a hierarchical approach that reduces the number of arithmetic and memory operations, and also discuss strategies to suppress artifacts due to sparse or missing sampling of surfaces in screen space. We show several examples demonstrating that our method achieves comparable results to offline rendering algorithms while achieving interactive speed on contemporary GPUs.

In this chapter, we present the following:

- a new formulation of the rendering equation that enables an efficient integration of the residual transport;

- a screen-space bias compensation amenable for GPU acceleration; and,

- a hierarchical integration scheme to further speed-up the rendering.

5.1 Screen-Space Techniques for Global Illumination

Most of the previous work that is relevant to this chapter was already mentioned in Chapter 3 and Section 4.1, thus we avoid repeating it. Here, we focus on literature that makes expensive, global illumination operations more tractable by moving them from world space to screen space.

These techniques are typically used when high performance needs outweigh the quality. Many of them are based on reflective shadow maps (RSMs) [Dachsbacher and Stamminger 2005]. Similar to a standard shadow map, an RSM captures directly lit surfaces, but stores additional information, such as position, normal, and material properties, that are required to compute one-bounce indirect illumination. Multi-resolution splatting [Nichols and Wyman 2009] is also based on RSMs computing indirect lighting (without visibility) at lower resolution for smooth surfaces, and more accurately in regions with geometric discontinuities. Image-space radiosity [Nichols et al. 2009] uses a hierarchy for the RSM, as well as for the image space. Computing ambient occlusion in image-space (e.g. see [Bavoil et al. 2008]) is widely used nowadays, and has been extended by Ritschel et al. [2009b] to account for directional lighting with colored shadows and indirect illumination from nearby surfaces. These techniques capture short range one-bounce light transport and typically require filtering to reduce sampling artifacts.

5.2 New Form of the Rendering Equation

Computing a solution to the rendering equation, e.g. using path tracing, is typically costly due to the inherent recursive nature of the light transport. Instant radiosity accelerates the computation of global illumination by replacing all but the first two terms[1] of the Neumann series in Equation (2.97) with lighting due to VPLs:

$$
\begin{aligned}
L &= \sum_{k=0}^{\infty} \mathbf{T}^k L_e \\
&\approx \underbrace{L_e}_{\text{emission}} + \underbrace{\mathbf{T}L_e}_{\text{direct illum.}} + \underbrace{\mathbf{T}\hat{L}}_{\text{indirect illum.}},
\end{aligned}
\tag{5.1}
$$

where $\hat{L}$ represents the radiance function due to a set of VPLs. For a single VPL with power Φ, and a given direction ω, $\hat{L}(\mathbf{x} \to \omega) = f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega')\Phi$, where $\mathbf{x}$ and ω' are the position and direction of incidence of the VPL, respectively. Since we deal with many-light methods and disregard effects due to participating media in this chapter, it will be convenient to express the transport operator $\mathbf{T}$ from Section 2.7.4 concisely as an integral over surfaces:

$$
(\mathbf{T}L)(\mathbf{x} \to \mathbf{z}) = \int_{\mathcal{A}} f_s(\mathbf{z} \leftarrow \mathbf{x} \leftarrow \mathbf{y}) V(\mathbf{x} \leftrightarrow \mathbf{y}) G(\mathbf{x} \leftrightarrow \mathbf{y}) L(\mathbf{y} \to \mathbf{x}) d\mathbf{y}.
\tag{5.2}
$$

We will now introduce two variants of $\mathbf{T}$. The first one, called the *bounded transport operator* $\mathbf{T}_b$, replaces the standard geometry term G with its bounded version G_b. We will denote the difference between $\mathbf{T}$ and $\mathbf{T}_b$ as the *residual transport operator* $\mathbf{T}_r$, which replaces G in Equation (5.2) be

---

[1] The original instant radiosity [Keller 1997] used VPLs even for the second term, i.e. the direct illumination.

the residual geometry term G_r. As such, Equation (5.1) can be written as:

$$
\begin{aligned}
L &\approx L_e + \mathbf{T}L_e + (\mathbf{T}_b + \mathbf{T}_r)\hat{L} \\
&\approx L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L} + \mathbf{T}_r\hat{L}.
\end{aligned}
\tag{5.3}
$$

In the equation above, the VPL lighting is split into the bounded and the residual transport, where the latter causes the splotchy artifacts. Most many-light techniques thus omit the residual transport, which biases the integration. To preserve unbiased results, Kollig and Keller [2006] propose to use hemispherical formulation of $\mathbf{T}_r$ and evaluate the residual transport using path tracing. As this is costly, we propose a slightly different perspective on the problem.

First, we remove the source of singularities (i.e. $\hat{L}$) and replace it by a general reflected radiance function ($L - L_e$). Conceptually, the equation remains the same, the only difference is that $\mathbf{T}_r$ is now integrating "reflected light" instead of illumination from VPLs:

$$
L \approx L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L} + \mathbf{T}_r(L - L_e).
\tag{5.4}
$$

By recursively expanding Equation (5.4), we obtain our novel unbiased formulation of the rendering equation for rendering with VPLs:

$$
L \approx L_e + \sum_{i=0}^{\infty} \mathbf{T}_r^i (\mathbf{T}L_e + \mathbf{T}_b\hat{L})
\tag{5.5}
$$

The above equation states that unbiased results can be obtained as an infinite sum of residual operators that are recursively applied to direct illumination and bounded indirect illumination computed with VPLs. That is, we can approximate global illumination using bounded VPL lighting, and afterwards restore *all* residual transport using only the bounded solution. The first term of the sum represents the direct and the bounded indirect illumination. Every further summand represents the (bounded) compensation for the bounded contribution of the previous step. As bounding occurs in each iteration, the results will be unbiased only for infinite sums.

In practice, we can only evaluate a finite number of iterations. The error ϵ^k introduced by considering the first k iterations can be expressed as a sum over all omitted higher order terms, or even more concisely as:

$$
\epsilon^k = \mathbf{T}_r^{k+1}(\mathbf{T}L_e + \mathbf{T}\hat{L}).
\tag{5.6}
$$

Notice that the error uses the complete transport operator $\mathbf{T}$ for the VPL lighting, therefore no further compensation is necessary. An important observation is that the energy gain due to the compensation is $(k+1)$-times convolved with the BRDF, and therefore dropping exponentially with increasing k. For practical applications, this translates into choosing k such that the *visible* bias is removed. In all our test scenes we used 1 to 3 iterations, which yielded nearly indistinguishable results from unbiased reference solutions.

Using our reformulation, we derive a new rendering algorithm with bias compensation that consists of two major steps:

1. We first compute direct and bounded indirect illumination of each shading point by applying $\mathbf{T}$ to light from primary light sources and $\mathbf{T}_b$ to illumination from VPLs, respectively.

2. Next, we apply the residual operator iteratively k-times to compensate for the bounding.

Instead of storing shading points over all surfaces, we will use a screen-space approach: we compute illumination for visible surfaces only, and also use exclusively these surfaces to compensate for the bias. The camera projection thus defines the integration domain and the raster provides an efficient domain discretization for storing intermediate results.

5.3   Integration in Screen Space

In this section, we describe how to transform the residual transport from Equation (5.5) into an efficient screen-space integration technique, which we denote as *screen-space bias compensation* (SSBC). Compensating for the bias in screen space has two major advantages: first, we can apply the residual transport operator as a post-processing filter that operates on the illumination sampled and stored in screen space. This is achieved by projecting screen-space patches (pixels) back to world space and integrating over the corresponding subtended area. Second, we can easily find all (visible) surfaces that potentially contribute energy to the compensation, as these have to be nearby in world space, and thus also in screen space.

Evaluating $\mathbf{T}_r$ in screen space amounts to replacing the integral by a sum over a finite number of pixels (we discuss how to handle the inherent limitations of screen-space approaches in Section 5.3.2). For every pixel, we obtain its position $\mathbf{y}_i$ and normal $n(\mathbf{y}_i)$ from a G-buffer of the rendered image, and compute the surface area A_i, which the pixel represents in world space, using screen-space derivatives. For a shading point $\mathbf{x}$, we sum over M pixels and get:

$$\mathbf{T}_r L \approx \sum_{i=1}^{M} f_s(\mathbf{z} \leftarrow \mathbf{x} \leftarrow \mathbf{y}_i) G_r(\mathbf{x} \leftrightarrow \mathbf{y}_i) V(\mathbf{x} \leftrightarrow \mathbf{y}_i) L(\mathbf{y}_i \rightarrow \mathbf{x}) A_i. \tag{5.7}$$

As previously mentioned, only surfaces near $\mathbf{x}$ contribute to the compensation – otherwise the residual geometry term $G_r(\mathbf{x} \leftrightarrow \mathbf{y}_i)$ becomes zero. This has two consequences: first, we can estimate the radius of this region in screen space and restrict the sum to pixels therein. Second, we observe that nearby surfaces are rarely occluded and we can thus omit the visibility function, as also proposed in [Arikan et al. 2005, Davidovič et al. 2010] and the previous chapter.

To restrict the sum to nearby surfaces, we first estimate a world-space bounding region, which contains surfaces contributing to the residual transport. A conservative spherical estimate, which does not require any *a-priori* information about the geometry in the G-buffer, defines the bounding region only based on distance (assuming the two cosine terms in the geometry term to be 1) and the bound b. The radius r of the bounding sphere in world space equals $r = 1/\sqrt{b}$, and can be transformed to screen-space radius r_s, e.g. for a perspective projection as:

$$r_s = \frac{r}{\tan \frac{\delta}{2} \|\mathbf{z} - \mathbf{x}\|}, \tag{5.8}$$

where δ is the field of view. Note that r_s depends on the distance from the camera and for closer $\mathbf{x}$ the screen-space radius becomes bigger spanning more pixels. Since the number of pixels lying inside the bounding bounding region can exceed several thousands (see Figure 5.2.b; 1 mipmap level), we derive a hierarchical integration scheme that greatly helps to achieve interactive performance.

(a) Sampling in screen space   **(b)** Bounding region   **(c)** Mipmap level l **(d)** Mipmap level $l-1$
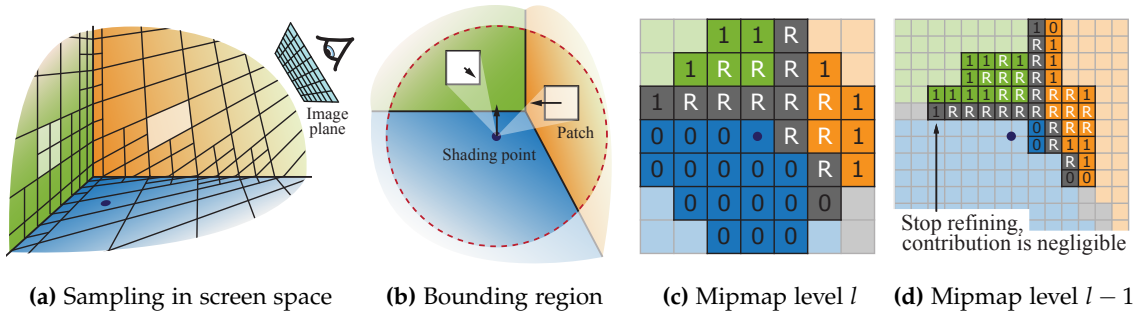
Figure 5.1: We compute the bias compensation by adaptively applying the residual operator to world-space patches obtained by projecting screen-space pixels back to world space (a). We first estimate a bounding region (b) that contains all surfaces with possibly non-zero contribution, and then, using a hierarchical traversal, we evaluate the contribution of individual neighboring patches (c) and (d). Patches spanning discontinuities or subtending a large projected solid angle are refined (R), until their contribution can be estimated accurately (1), or drops to zero (0).

5.3.1   Hierarchical Integration

The goal of the hierarchical integration is to compute the contribution from smooth or more distant (yet still in the bounding region) surfaces using less pixel samples. Figure 5.1 demonstrates the general idea of adaptively sampling the integration domain and refining where the information is inaccurate. For this we compute a mipmap chain for the G-buffer, which contains averaged positions, normals, and material properties, summed pixel areas, and illumination computed from the bounded light transport, i.e. the color of the pixels. Similarly to Nichols et al. [2009], we also compute a discontinuity buffer (and a mipmap chain thereof) to avoid integrating over sharp edges and depth discontinuities.

When integrating the surfaces' contributions, we start on the coarsest mipmap level (typically with a resolution of 64^2 for an image resolution of 1024^2), and determine whether the computation using this patch is accurate enough, or if we need to refine and proceed to the finer mipmap level. In order to avoid spatial and temporal artifacts in dynamic scenes, we refine the integration if:

1. the projected solid angle of the patch exceeds a given threshold (typically ≈ 0.08 sr) because it is too large or too close to the shading point $\mathbf{x}$,

2. the current patch spans a discontinuity. Then the position, normal, and area at the coarser mipmap level do not represent the original geometry correctly.

If at least one of the criteria is met, we refine by recursively integrating over the four corresponding sub-patches at the next finer level. The second criterion can cause a lot of refinement in areas with negligible contribution, e.g. distant surfaces. Therefore, we only refine at discontinuities when the projected solid angle is higher than a user specified threshold (≈ 0.04 sr). This significantly improves the performance without introducing noticeable artifacts, as it is only effective for patches with low possible contribution.

(a) Number of processed patches per pixel

(b) Number of processed patches with varying number of mipmap levels

(c) Relative execution time
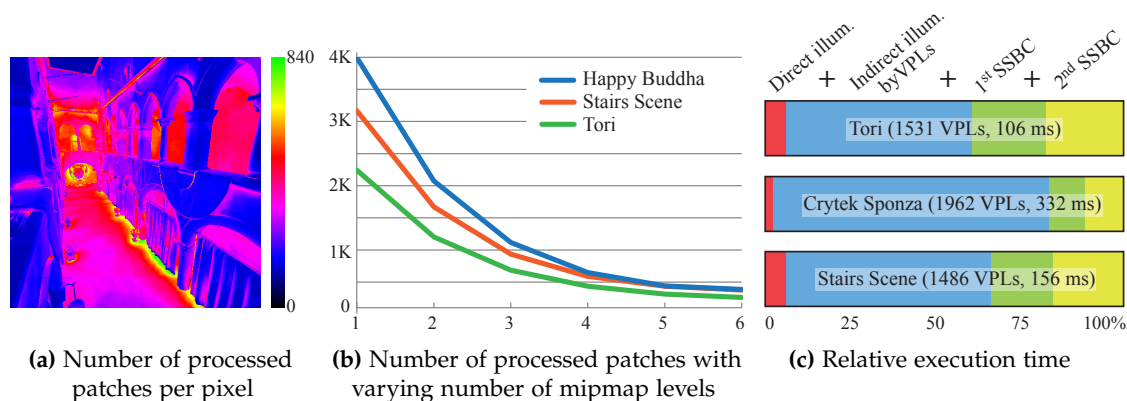
Figure 5.2: In (a), we visualize the total, per-pixel number of patches used to estimate the residual transport in the Crytek Sponza scene. In (b), we show how the average, per-pixel number of patches depends on the number of mipmap levels in the hierarchical G-buffer. The plot in (c) illustrates the relative execution time spent on different rendering tasks: the cost of two compensation steps is moderate compared to that of computing illumination from VPLs. The number of VPLs and the total rendering time is given in parentheses.

5.3.2 Avoiding Overcompensation

There are several inherent problems common to most screen-space approaches. Information about the geometry can either be missing completely when the surface is not directly visible to the camera, or sampled too sparsely if surfaces are seen from grazing angles. The problem of hidden surfaces can be lifted using depth peeling or multi-fragment rendering. This complicates the rendering algorithm and hinders the elegance of the screen-space integration. Fortunately, since the residual transport represents only a fraction of the total energy, the absolute error is still relatively small. In most cases, it is thus sufficient to use the frontmost surfaces only.

When a surface is nearly perpendicular to the view direction, the corresponding screen-space pixels represent a large area in world space, even on the finest mipmap level. This can result in overestimating the residual transport and thus bright areas in the image. The brightening would sometimes be more distracting than the unbounded contribution of a nearby VPL; therefore, we use a quadratic falloff to decay the contribution of pixels that have the angle between the surface normal and the view direction greater than 80 degrees.

5.4 Implementation Details

We implemented our technique in DirectX 11 with all steps of our bias compensation executed on the GPU. The only exception are the random walks for distributing VPLs, which are computed on the CPU. Note that this is never the bottleneck, as we only generate a few thousands of VPLs.

During rendering, the visibility of VPLs is computed using imperfect shadow maps [Ritschel et al. 2008]. The bias compensation is implemented in a compute shader, which enables more efficient handling of the hierarchical refinement. The compute shader takes a mipmap chain of the G-buffer storing position, normal, area, BRDF, and illumination (direct and bounded VPL contribution) as input. Then we apply the hierarchical screen-space compensation to restore
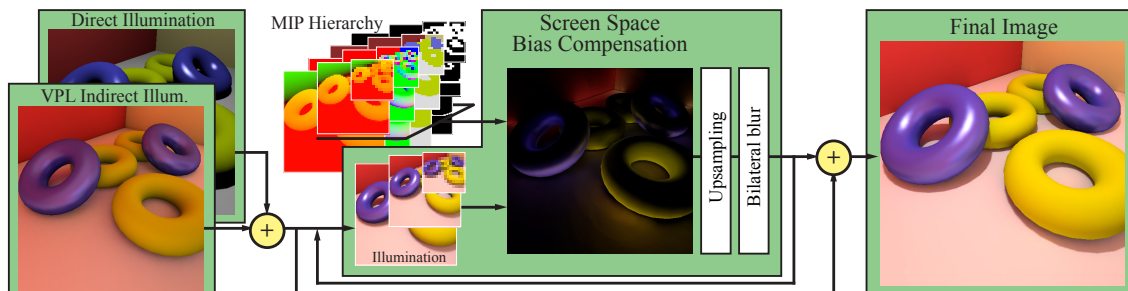
Figure 5.3: Our SSBC takes a multi-resolution image of direct and clamped indirect illumination, and the hierarchical G-buffer as input. The compensation result is upsampled and optionally blurred, and then added to the direct and clamped illumination yielding the final image.

the missing short-distance one bounce illumination. Optionally, we use an edge-preserving bilateral Gaussian blur on the compensation result to avoid blocky artifacts that can appear when the hierarchical integration uses too few samples. When applying multiple compensation steps (i.e. multiple iterations in Equation (5.5)), we always use the illumination buffer from the previous step as the input for the next iteration. This requires updating the illumination mipmap. Figure 5.3 shows the workflow of our technique.

As the indirect illumination usually exhibits smooth gradations only, we can increase the performance – without noticeable impact on visual quality – by computing the indirect illumination and bias compensation at half resolution. We then use bilateral upsampling as in [Ritschel et al. 2009a] and compute these two components only at full resolution where the upsampling failed.

5.5 Results

In this section, we compare the quality of our results to an implementation of the unbiased instant radiosity algorithm [Kollig and Keller 2006], analyze the rendering performance, and discuss different settings. All renderings have been computed using an ATI Radeon HD 5870 running on an Intel Core i7 860 with 2.8 GHz and 8 GB of RAM.

Figure 5.2.b shows the dependency of the number of screen-space patches, required for the compensation, on the number of mipmap levels of the G-buffer. Without the hierarchical approach, the algorithm gathers the illumination from up to four thousand patches for *every* pixel. This number quickly decreases if we use more levels in the mipmap. Figure 5.2.c reports the relative time spent on the individual parts of the rendering pipeline. Here we always used hierarchies with 6 levels computing the compensation for 1024×768 images and performing two compensation steps (green and yellow).

In Figure 5.4, we compare the quality of the SSBC to a ground-truth [Kollig and Keller 2006]. For this comparison, we used ray traced shadows instead of shadow maps to focus only on the error due to the integration in screen space (which was still done on the GPU). The ground-truth rendering took about 10.9 hours to produce results with an acceptable noise level. Our bias compensation obtains very similar results at interactive frame rates (3 SSBC steps at highest quality settings take 550 ms).

Figure 5.5 demonstrates SSBC on three scenes and illustrates the differences between one and

(a) Bounded VPLs   **(b)** Bounded VPLs + 3-step BC   **(c)** Bounded VPLs + 3-step SSBC

(d) 3-step BC   **(e)** 3-step SSBC
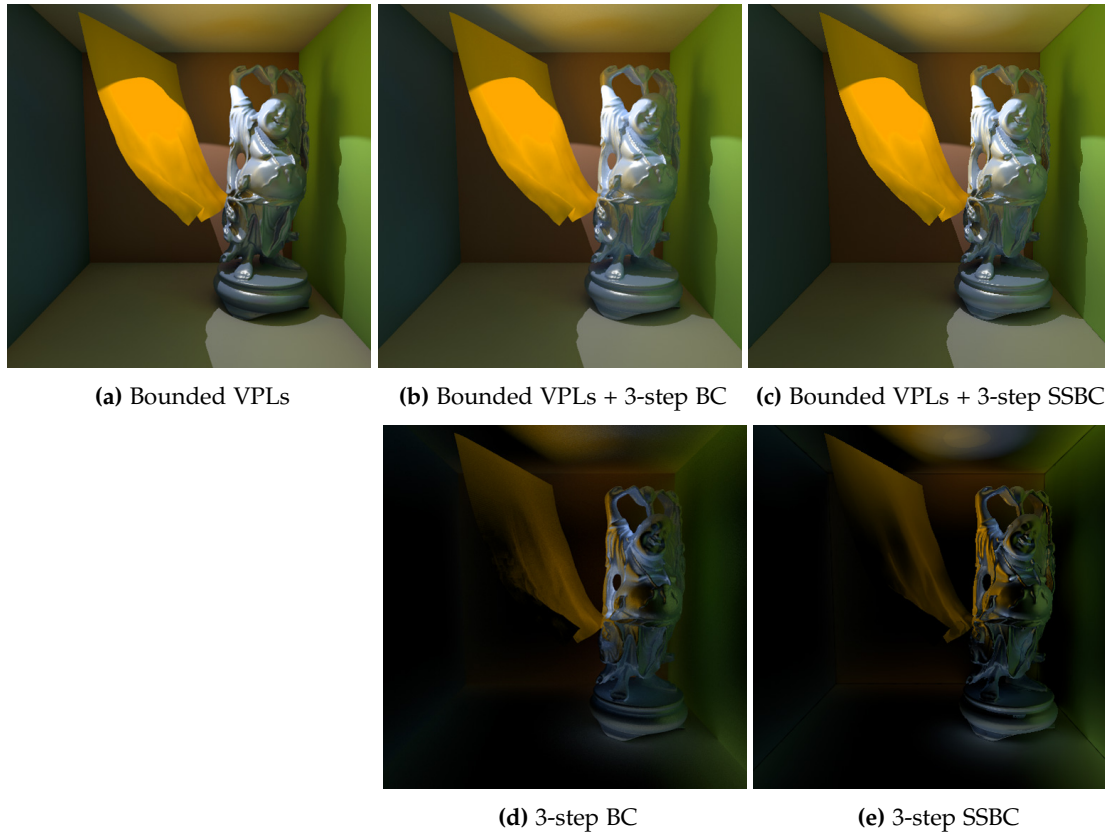
Figure 5.4: Comparison of our SSBC to a ground-truth solution: (a) offline rendering with bounded indirect illumination without bias compensation (20 min), and (b) with bias compensation [Kollig and Keller 2006] (additional 10.9 hours). (c) shows the result with three SSBC steps (i.e. three iterations in Equation (5.5)). The energy recovered from both compensation methods is shown in (d) and (e), respectively.

two bias compensation steps. The figure also provides a comparison to bounded and reference solutions. We observe that one compensation step is often sufficient for diffuse surfaces, while the second step still contributes noticeably on glossy surfaces.

5.6   Conclusion

In this chapter, we presented a novel many-light method for interactive rendering of high-quality global illumination. We show that bias compensation can be formulated as a post-processing step enabling efficient, screen-space approximations. Our method improves the quality and correctness of VPL-based methods by recovering the residual energy using a hierarchical, screen-space approach, whereas previous approaches, operating in world space, require expensive ray casting and are thus orders of magnitude slower.

The relative time required for our compensation is only a fraction of the time spent on illuminating the scene with VPLs. This highly contrasts with the original bias compensation [Kollig and Keller 2006], that can easily increase the total rendering time by orders of magnitude. Unlike the hierarchical image-space radiosity [Nichols et al. 2009], we still evaluate visibility between
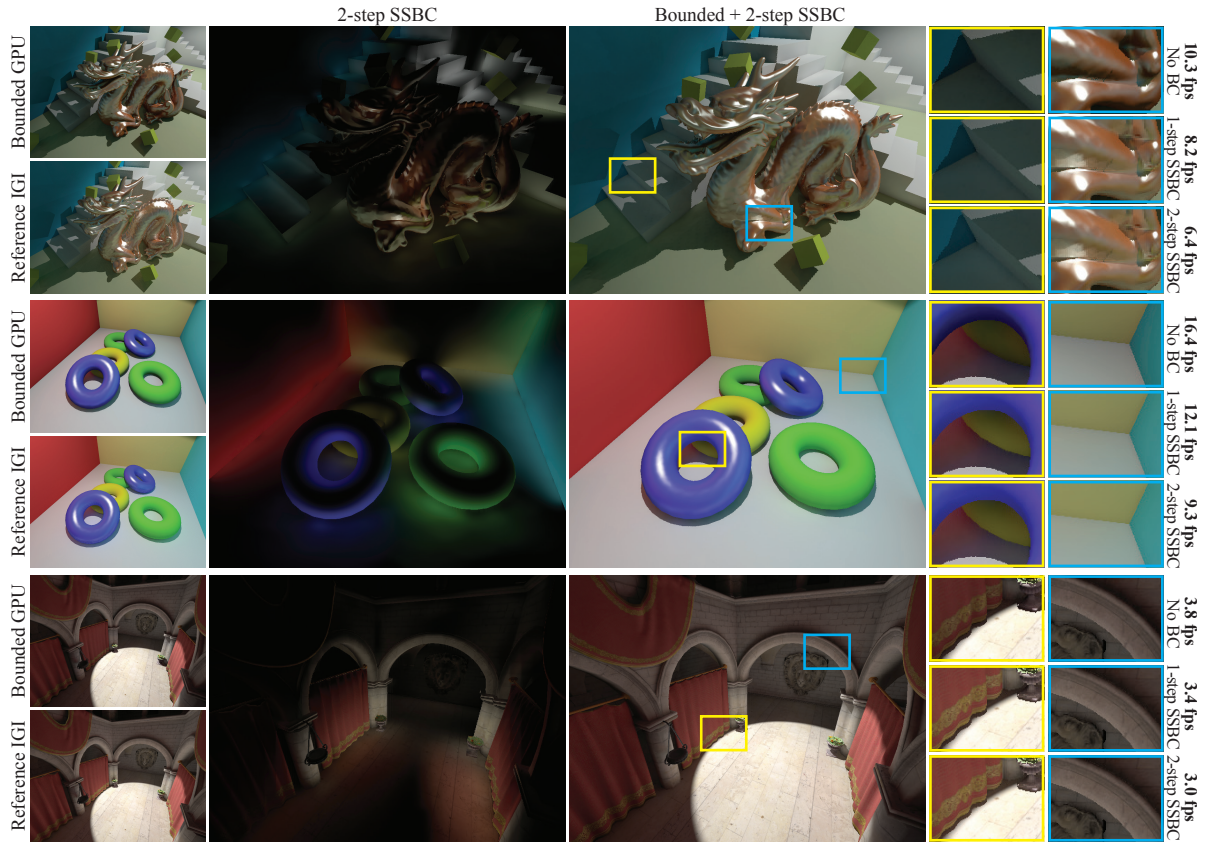
Figure 5.5: Three of our test scenes for which we show a bounded GPU and a reference CPU solution (left column), and our technique with 2 compensation steps (middle columns). The insets on the right detail parts, where the compensation contributes significantly. Note that a single compensation step is typically sufficient for diffuse surfaces, while glossy surfaces (e.g. the dragon) benefit from more steps. Each compensation step takes approximately 27 ms for an image resolution of 1024×768.

the shaded point and VPLs; visibility tests are omitted only locally and only for the residual transport.

In order to overcome the artifacts stemming from the incomplete information in screen space, our method could be combined with any technique for revealing information about hidden surfaces, such as depth peeling, scene voxelization, or surfel injection. These improvements trade speed for quality and might be suitable for accelerating high-quality bias compensation in offline renderers.

Virtual Ray Lights

Death is pitch-dark, but colors are light.
To be a painter, one must work with rays of light.

— EDVARD MUNCH (1863–1944)

While the previous two chapters focused on how to approximate and accelerate bias compensation to achieve interactive performance, in this and the following chapter, we aim more at the quality and convergence to ground truth. We also strive to solve the complete radiative transport, not just the surface illumination. Though variants of path tracing (e.g. [Lafortune and Willems 1996]) provide general solutions covering most lighting configurations, they converge slowly. Alternatives based on diffusion theory (e.g. [d'Eon and Irving 2011, Jensen et al. 2001]) yield efficient approximations, but are only applicable with strong constraints on the scattering properties of the medium. For transport in general participating media, two-pass approaches, such as many-light algorithms and density estimation techniques, are often the preferred choice for offline rendering.

One of the main disadvantages of many-light algorithms that somehow compensate for the bounding is that they split the light transport into two parts, each of which is computed differently. First, the rendering equation is partially solved by computing the bounded illumination due to VPLs; second, the residual energy is added using a different integration scheme. The initially very appealing assumption of representing multibounce illumination with a set of VPLs turns into an obstacle when aiming for results that are free of visible artifacts. This is because contracting the energy into a finite number of infinitesimal point lights emphasizes the singularity of the area-formulated rendering equation. We thus need to bound the transport and add an extra estimator to recover the residual energy, which makes the implementation more involved and somewhat hinders the initially very elegant and straightforward idea.

In this chapter, we try to reduce the singularity so that we no longer need to bound the contribution of the virtual light. We leverage an important observation that absorption and scattering in the medium can be formulated continuously along directions that light follows. As a result, we can take entire segments (not just the vertices) of a photon path and convert them into *virtual ray lights* (VRLs), see Figure 6.1. This, analogously to photon beams [Jarosz et al. 2011a], results in denser sampling of the integration domain. However, compared to photon beams and VPL-based many-light techniques, VRLs provide higher quality multiple scattering at a fractional cost by exploiting these important benefits:

- line segments, as opposed to points, sample the medium more densely requiring fewer virtual lights;

(a) VPLs   **(b)** VRLs   **(c)** Lighting with a VPL   **(d)** Lighting with a VRL
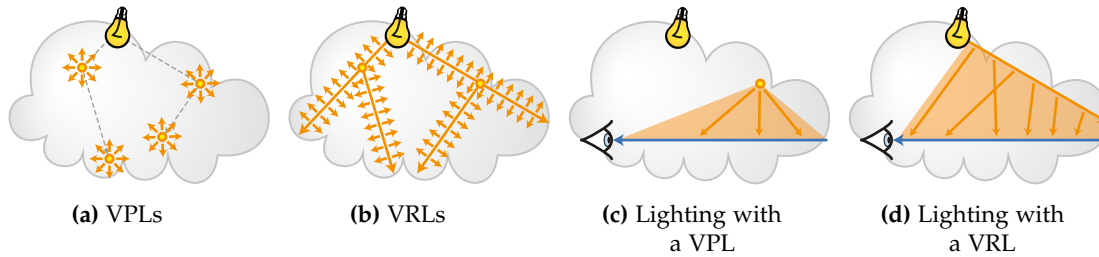
Figure 6.1: Virtual point light methods (a) convert vertices of each random-walk into a collection of virtual point lights (VPLs). In this chapter, we propose to convert entire segments of the random-walk into virtual ray lights (b). This results in denser sampling, provably weaker singularities, and higher quality when estimating illumination from the collection of virtual lights (c, d).

- distributing energy over line segments (instead of concentrating it at discrete points) provably reduces the order of the singularity, diminishing the need to bound energy;

- using a product importance sampling scheme to efficiently integrate over the 2D domain defined by a camera ray and a VRL allows us to more robustly handle the traditionally challenging cases (for VPL methods) of scattering from anisotropic phase functions; and,

- computing the illumination from VRLs acts as a "guided" final gather pass over photon beams, leading to a more efficient integration scheme.

The third point in the list above reveals an important aspect of lighting with VRLs. In order to calculate the energy transfer between a single ray light and a single camera ray, we need to solve an 2D integral. One dimension in the 2D integration domain corresponds to a position on the VRL; the second dimension then represents to the position on the camera ray. In Section 6.3.1, we present a product sampling method to efficiently compute these integrals in a Monte Carlo framework. Our unbiased approach is effective in the presence of highly-anisotropic scattering/lighting and outperforms other variance reduction techniques, such as multiple importance sampling Veach [1997], by accurately capturing the shape of the integrand in the 2D domain.

6.1 From Points Towards Rays

In this section, we list several examples demonstrating how using lines instead of points for sampling, integrating, or storing data can improve the result. In photon mapping, several authors proposed to use entire segments of the photon path to reduce the various forms of the density estimation bias [Havran et al. 2005, Herzog et al. 2007, Lastra et al. 2002]. This work has been extended to participating media by Jarosz et al., who first demonstrate that looking up photons using a beam query (instead of a number of point queries along the camera ray) is more efficient [Jarosz et al. 2008b], and then present a comprehensive study that compares point and line primitives for storing and querying radiance values [Jarosz et al. 2011a]. The authors also proposed to use *photon beams*, i.e. segments of photon paths with finite thickness, and later formulated a progressive convergent algorithm [Jarosz et al. 2011b] that shrinks the thickness of beams over time to gradually reduce the bias to zero. A similar idea to photon beams was concurrently presented by Sun et al. [2010] to simulate single scattering and volumetric caustics by finding nearly intersecting camera and light paths.

In addition to light transport, line samples were used to improve other aspects of rendering as well. Jones and Perry [2000] place lines over pixels and analytically compute triangle coverage to obtain the final pixel color. Line samples can also be used to improve the integration of stochastic effects. Tzeng et al. [2012] demonstrate the benefits of evaluating visibility along lines for depth of field. Gribel et al. [2010] use line samples to analytically integrate visibility for motion blur. Barringer et al. [2012] extend line sampling of visibility to thin structures, such as hair, fur, or grass. Similarly to point sampling, line samples may suffer from aliasing if spaced or oriented regularly. Sun et al. [2013] study different distributions of line segment samples and propose a sampling scheme that retains blue-noise-like properties.

Interestingly, the problem of computing lighting from a VRL is dual to the so-called airlight integral [Lecocq et al. 2000]. The transport of energy from a VRL towards a single point can be seen as a reverse problem to estimating the contribution from a single VPL to entire camera ray. Several techniques have been proposed to evaluate this transport analytically, e.g. [Pegoraro 2009, Sun et al. 2005], or numerically using ray marching [Perlin and Hoffert 1989] or Monte Carlo importance sampling [Kulla and Fajardo 2012, Steinberg and Kalos 1971]. Our approach can thus benefit from these integration schemes and we point to them throughout the text when appropriate.

6.2 Light Transport with VRLs

In this section, we demonstrate how line samples can be used in the context of rendering participating media. More precisely, each our line sample is a continuous linear light source (abbr. ray light) whose geometry is represented by an oriented line segment with origin $\mathbf{x}$, direction $\omega_{\mathbf{x}}$, and length t. We also need to quantify the energy that the ray light emits. This can be done by specifying the radiant flux emitted by the line and how the flux changes with position and direction along the line. Since we want to use ray lights as samples of light transport, a single VRL should represent light that travels from $\mathbf{x}$ in direction $\omega_{\mathbf{x}}$ and scatters exactly once somewhere along the straight path. Furthermore, the amount and the distribution of the scattered light should adhere firmly to the parameters of the medium that the ray goes through.

For each VRL, we thus specify only flux Φ that is *all* emitted from the origin $\mathbf{x}$ in direction $\omega_{\mathbf{x}}$.[1] The flux "emitted" from an arbitrary point $\mathbf{x}_v$ on the ray light is then defined as $\Phi T(\mathbf{x} \leftrightarrow \mathbf{x}_v)\kappa_s(\mathbf{x}_v)$. Its directional distribution, i.e. the radiant intensity at $\mathbf{x}_v$, is given by the phase function $f_p(\omega_{\mathbf{x}} \rightarrow \omega)$. Putting all these together, we can define the radiance emitted from a point $\mathbf{x}_v$ on the VRL in direction ω as:

$$L(\mathbf{x}_v \rightarrow \omega) = \Phi T(\mathbf{x} \leftrightarrow \mathbf{x}_v)\kappa_s(\mathbf{x}_v)f_p(\omega_{\mathbf{x}} \rightarrow \omega). \tag{6.1}$$

We can now express the fluence due to a single VRL at an arbitrary point $\mathbf{y}$ in the scene. We simply take the intensity at each point of the VRL towards $\mathbf{y}$ and account for visibility, transmittance, and the quadratic distance fall-off, yielding:

$$E(\mathbf{y}) = \Phi \int_0^t \frac{\kappa_s(\mathbf{x}_v)f_p(\omega_{\mathbf{x}} \rightarrow \omega)T(\mathbf{x} \leftrightarrow \mathbf{x}_v)T(\mathbf{x}_v \leftrightarrow \mathbf{y})V(\mathbf{x}_v \leftrightarrow \mathbf{y})}{\|\mathbf{x}_v - \mathbf{y}\|^2} \mathrm{d}v, \tag{6.2}$$

---

[1] Here, we could indeed talk about emitted radiance, nevertheless, we stick to the convention of specifying the total emitted energy of a light source using flux.

(a) Media-to-media transport L_m^m **(b)** Media-to-surface transport L_s^m
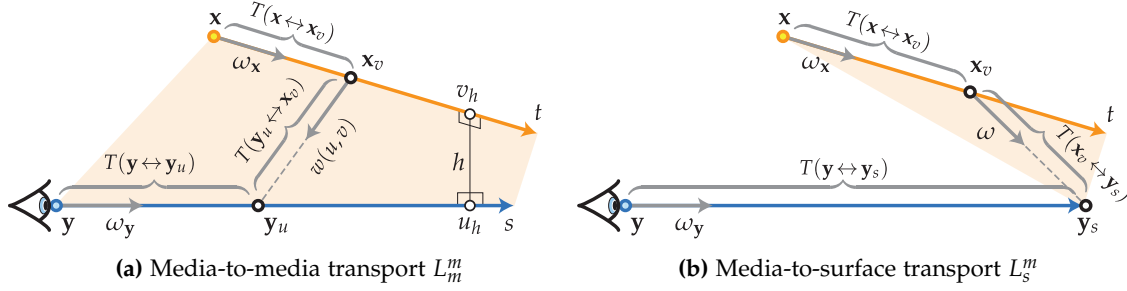
Figure 6.2: Illustration of the media-to-media (a) and media-to-surface (b) transport showing some of the terms, e.g. transmittance and inverse-squared distance, that form the integrand of Equations (6.3) and (6.4).

where $\mathbf{x}_v = \mathbf{x} + v\omega_\mathbf{x}$ and ω is the direction from $\mathbf{x}_v$ towards $\mathbf{y}$.

Let us now consider a camera ray with origin $\mathbf{y}$ and direction $\omega_\mathbf{y}$ that hits a surface point $\mathbf{y}_s$ at distance s. We shall now calculate the fraction of the VRL's flux that reaches the camera ray and scatters towards its origin $\mathbf{y}$. This can be split into two components. The first component, which we will refer to as *media-to-media* transport L_m^m, accounts for the in-scattered light along the camera ray, see Figure 6.2.a. This can be expressed as the fraction of fluence that scatters towards the origin and does not get absorbed or out-scattered along the camera ray:

$$L_m^m(\mathbf{y} \leftarrow \omega_\mathbf{y}) =$$
$$\Phi \int_0^s \int_0^t \frac{\kappa_s(\mathbf{x}_v)\kappa_s(\mathbf{y}_u)f_p(\omega_\mathbf{x} \rightarrow \omega)f_p(\omega \rightarrow -\omega_\mathbf{y})T(\mathbf{x} \leftrightarrow \mathbf{x}_v)T(\mathbf{x}_v \leftrightarrow \mathbf{y}_u)T(\mathbf{y}_u \leftrightarrow \mathbf{y})V(\mathbf{x}_v \leftrightarrow \mathbf{y}_u)}{\|\mathbf{x}_v - \mathbf{y}_u\|^2} \mathrm{d}v\mathrm{d}u,$$
$$(6.3)$$

where $\mathbf{y}_u = \mathbf{y} + u\omega_\mathbf{y}$. The above equation represents the core of VRL-based many-light algorithms. It simulates two bounces of volumetric light transport and can thus be—with VRLs generated accordingly—used for efficiently computing multiple-scattering.

The second component, referred to as *media-to-surface* L_s^m, accounts for the fraction of irradiance from Equation (6.2) that hits the surface at $\mathbf{y}_s$, scatters towards $\mathbf{y}$, and reaches the origin of the camera ray:

$$L_s^m(\mathbf{y} \leftarrow \omega_\mathbf{y}) =$$
$$T(\mathbf{y} \leftrightarrow \mathbf{y}_s)\Phi \int_0^t \frac{\kappa_s(\mathbf{x}_v)f_p(\omega_\mathbf{x} \rightarrow \omega)f_s(\omega \rightarrow \mathbf{y}_s \rightarrow -\omega_\mathbf{y})D_{\mathbf{y}_s}(\mathbf{x}_v)T(\mathbf{x} \leftrightarrow \mathbf{x}_v)T(\mathbf{x}_v \leftrightarrow \mathbf{y}_s)V(\mathbf{x}_v \leftrightarrow \mathbf{y}_s)}{\|\mathbf{x}_v - \mathbf{y}_s\|^2} \mathrm{d}v.$$
$$(6.4)$$

For an illustration of the individual terms, see Figure 6.2.b.

So far we considered emission from the intermediate points of the VRL only. If the end-point of the VRL $\mathbf{x}_t$ is on a surface, there will be some non-zero flux that bounces off $\mathbf{x}_t$. We propose to treat $\mathbf{x}_t$ as a surface VPL with incident power $\Phi_{\mathbf{x}_t}$:

$$\Phi_{\mathbf{x}_t} = \Phi T(\mathbf{x} \leftrightarrow \mathbf{x}_t), \qquad (6.5)$$

and calculate the *surface-to-media* L_m^s analogously to the media-to-surface illumination: in both cases we deal with a transport between a volumetric line and a surface point, which requires

solving the well-known airlight integral. The only difference between the two transports is the direction in which the light propagates.

In case when the end-points of the VRL and of the camera ray are both on a surface, we should also evaluate the respective *surface-to-surface* transport. As this amounts to calculating the contribution of a surface VPL at $\mathbf{x}_t$ to a surface point $\mathbf{y}_s$, which is described in Chapter 3, we omit repeating the equations here. To compute the complete contribution of a VRL to a single camera ray, we simply add all the aforementioned components together:

$$L(\mathbf{y} \leftarrow \omega_{\mathbf{y}}) = L_m^m(\mathbf{y} \leftarrow \omega_{\mathbf{y}}) + L_s^m(\mathbf{y} \leftarrow \omega_{\mathbf{y}}) + L_m^s(\mathbf{y} \leftarrow \omega_{\mathbf{y}}) L_s^s(\mathbf{y} \leftarrow \omega_{\mathbf{y}}). \tag{6.6}$$

Generation of VRLs. In order to provide more intuition on how to use VRLs as samples of global illumination, we now briefly describe the process of generating them. This is similar to generating VPLs in traditional many-light algorithms. We first emit random-walk photon paths from light sources that scatter at surfaces and within the medium. At each bounce of the photon path, point-based many-light methods store a VPL, i.e. the "flux" carried by the photon path, the position of the bounce, direction of incidence, and possibly some other additional information. In case of VRLs, we store entire segments defined by the origin $\mathbf{x}$ (i.e. the previous path vertex), direction $\omega_{\mathbf{x}}$ (from $\mathbf{x}$ towards the current path vertex), and length t. For the length, we use the distance from the origin to the nearest surface in direction $\omega_{\mathbf{x}}$. Alternatively, one can also use the actual length of the random-walk segment—we discuss the implications in Section 6.7—but we found that longer ray lights yield better results in our test scenes. Finally, we define the flux Φ of the VRL as the flux carried by the photon path in direction $\omega_{\mathbf{x}}$ after bouncing at $\mathbf{x}$.

For media-to-media transport (multiple scattering) we evaluate Equation (6.3) for each camera ray and VRL using an efficient importance sampling strategy that we describe next. For media-to-surface and surface-to-media transport (i.e. indirect illumination from the volume) we solve the simpler 1D problem expressed in Equation (6.4) at the surface hit points. These steps can be trivially repeated in a progressive fashion to provide interactive previews.

6.3 Importance Sampling for Transport between Two Rays

Equation (6.3) forms the basis for an efficient evaluation of multiple scattering in participating media. It defines a 2D integration domain (abbr. *uv*-domain), where one axis is the length t along a VRL and the other axis is the length s along a camera ray. The integrand within this domain incorporates scattering coefficients, phase functions, and transmittances along the camera and light ray, and the inverse-squared distance, visibility, and transmittance between the points corresponding to u and v. We visualize the *uv*-domain for a single, unoccluded VRL/ray pair in Figure 6.3.b. Unfortunately, a closed-form solution to this double integral is currently not known. As such, we estimate the value of the integral using Monte Carlo integration.

If we denote the integrand of Equation (6.3) as $g(u, v)$, we are interested in evaluating the following unbiased Monte Carlo estimator:

$$\langle L_s^m(\mathbf{y} \leftarrow \omega_{\mathbf{y}}) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{g(u_i, v_i)}{p(u_i, v_i)}, \tag{6.7}$$

| (a) Scene | (b) *uv*-domain | (c) Marginal PDF | (d) Conditional PDF |
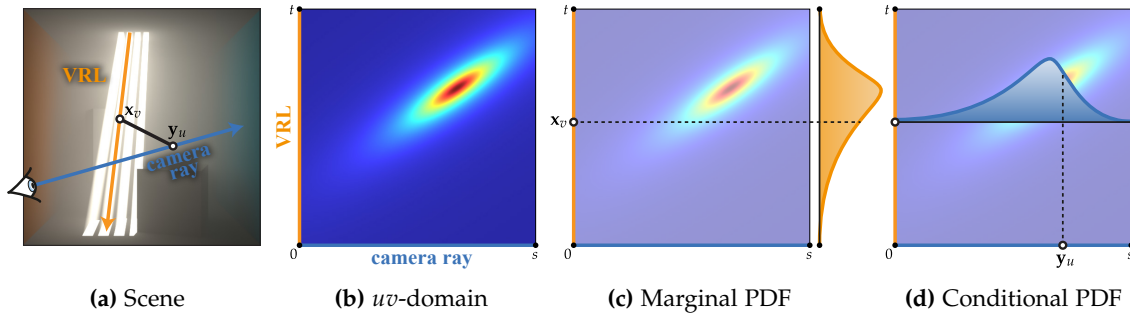
Figure 6.3: For every pair of a VRL and a camera ray (a), we compute the transport by integrating over the 2D *uv*-domain (b), here visualizing just the inverse-squared distance $w(u, v)^{-2}$. We first sample point $\mathbf{x}_v$ on the VRL using a marginal PDF (c) and then, for this $\mathbf{x}_v$, we construct a conditional PDF to sample point $\mathbf{y}_u$ on the camera ray (d).

where $p(u_i, v_i)$ is the probability of choosing a point (u_i, v_i) in the *uv*-domain. To evaluate this estimator efficiently, the PDF should include as many properties of *g* as possible. We begin with the case of isotropic phase functions, generalizing later to anisotropic unimodal phase functions.

6.3.1 Isotropic Scattering

In the isotropic case, the inverse-squared distance term in Equation (6.3) causes the most variation in the *uv*-domain. All the other terms are either constant or strictly bounded to some maximum value and so we primarily target a PDF proportional to the inverse-squared distance:

$$p(u_i, v_i) \propto w(u, v)^{-2}, \tag{6.8}$$

where *w* denotes the distance between the points on the VRL and the camera ray, i.e. $w(u, v) = \|\mathbf{x}_v - \mathbf{y}_u\|$. Unfortunately, sampling according to this PDF using the inversion method is not possible as computing and *inverting* the corresponding CDF eludes analytical computation.

Kulla et al. [2012] recently proposed an equiangular approach for importance sampling a camera ray according to the inverse-squared distance to a point light, and also applied this idea to a randomly sampled rectangular area light. We could trivially apply this approach to our context of a linear light (the VRL) by first uniformly choosing a random length v_i along the VRL, and then importance sampling the length u_i along the camera ray according to inverse-squared distance. Unfortunately, by (uniformly) sampling v_i along the VRL without regard for the camera ray, the sampling density along the *v* axis does not account for variation in $w(u, v)^{-2}$, resulting in a suboptimal distribution. We visualize this in the *uv*-domain in Figure 6.4 along with a corresponding Cornell box rendering, illustrating the effects of the suboptimal sampling distribution.

In order to incorporate variation in $w(u, v)^{-2}$ along both *v* and *u*, we propose to construct a joint distribution for the entire *uv*-domain and sample it using a two-stage procedure. We first distribute a sample v_i along the VRL using a marginal PDF $p(v)$ (see Figure 6.3.c) and then sample a length u_i along the camera ray according to a conditional PDF $p(u|v)$ based on the inverse-squared distance to the point at v_i (Figure 6.3.d). In order to make the sampling routine efficient, we strive for an analytic marginal PDF.

We start by applying a change of variables $\hat{u} = u - u_h$ and $\hat{v} = v - v_h$, and similarly for the VRL start and end points $\hat{v}_0$ and $\hat{v}_1$. Here, u_h and v_h are the ray parameters of the two closest points along the camera ray and VRL, which are separated by a Euclidean distance h (see Figure 6.2.a). Our motivation for these substitutions is to express all distances w.r.t the two closest points, which makes the derivation easier to follow. Using the law of cosines, we can now define the squared distance as $w(\hat{u}, \hat{v})^2 = h^2 + \hat{u}^2 + \hat{v}^2 - 2\hat{u}\hat{v}\cos\theta$, where $\cos\theta$ is the dot-product of direction of the camera ray and the VRL, i.e. $\cos\theta = \omega_{\mathbf{y}} \cdot \omega_{\mathbf{x}}$. The marginal PDF we seek can now be expressed as:

$$p(\hat{v}) = \frac{\int_{\hat{u}_0}^{\hat{u}_1} w(\hat{u}, \hat{v})^{-2}\, d\hat{u}}{\int_{\hat{v}_0}^{\hat{v}_1} \int_{\hat{u}_0}^{\hat{u}_1} w(\hat{u}, \hat{v})^{-2}\, d\hat{u}\, d\hat{v}}. \tag{6.9}$$

To best of our knowledge, there is no analytic solution to such integrals in the current math literature [Gradshteyn and Ryzhik 2007]. Therefore, we opt to simplify the inner integral by assuming the camera ray to be infinite (i.e. $\hat{u}_0 = -\infty$, $\hat{u}_1 = \infty$). With this change, the numerator evaluates to:

$$\int_{-\infty}^{\infty} w(\hat{u}, \hat{v})^{-2}\, d\hat{u} = \frac{\pi}{\sqrt{h^2 + \hat{v}^2 \sin^2\theta}}, \tag{6.10}$$

and the normalization term in the denominator evaluates to:

$$\int_{\hat{v}_0}^{\hat{v}_1} \int_{-\infty}^{\infty} w(\hat{u}, \hat{v})^{-2}\, d\hat{u}\, d\hat{v} = \pi \frac{A(\hat{v}_1) - A(\hat{v}_0)}{\sin\theta}, \tag{6.11}$$

where $A(x) = \sinh^{-1}\left(\frac{x}{h}\sin\theta\right)$.

With these analytic anti-derivatives we can further integrate Equation (6.9) to obtain the marginal CDF:

$$P(\hat{v}) = \frac{A(\hat{v}_0) - A(\hat{v})}{A(\hat{v}_0) - A(\hat{v}_1)}, \tag{6.12}$$

which can be readily solved for the inverse CDF:

$$P^{-1}(\xi) = \frac{h \sinh(\mathrm{lerp}(A(\hat{v}_0), A(\hat{v}_1), \xi))}{\sin\theta}. \tag{6.13}$$

We sample a length v_i (and thus a position $\mathbf{x}_{v_i} = \mathbf{x} + v_i\omega_{\mathbf{x}}$ on the VRL) by generating a random number $\xi_{i,1} \in \langle 0, 1 \rangle$ and passing it into the inverse CDF. The result can be interpreted as a VPL located at $\mathbf{x}_{v_i}$.

In the second step, we apply Kulla et al.'s method to obtain a sample location $\mathbf{y}_{u_i} = \mathbf{y} + u_i\omega_{\mathbf{y}}$ along the camera ray that is distributed according to the inverse-squared distance to $\mathbf{x}_{v_i}$. This involves generating another random number $\xi_{i,2} \in \langle 0, 1 \rangle$ and inserting it into their inverse CDF:

$$P^{-1}(\xi) = h \tan(\mathrm{lerp}(B(\hat{u}_0), B(\hat{u}_1), \xi)), \tag{6.14}$$

where $B(x) = \tan^{-1}(x/h)$. Note that this is actually equivalent to the well-known truncated Cauchy distribution. The final PDF is simply the product of the PDFs from these two sampling steps.

(a) Medium with an isotropic PF   **(b)** Uniform and equiangular   **(c)** Our analytic
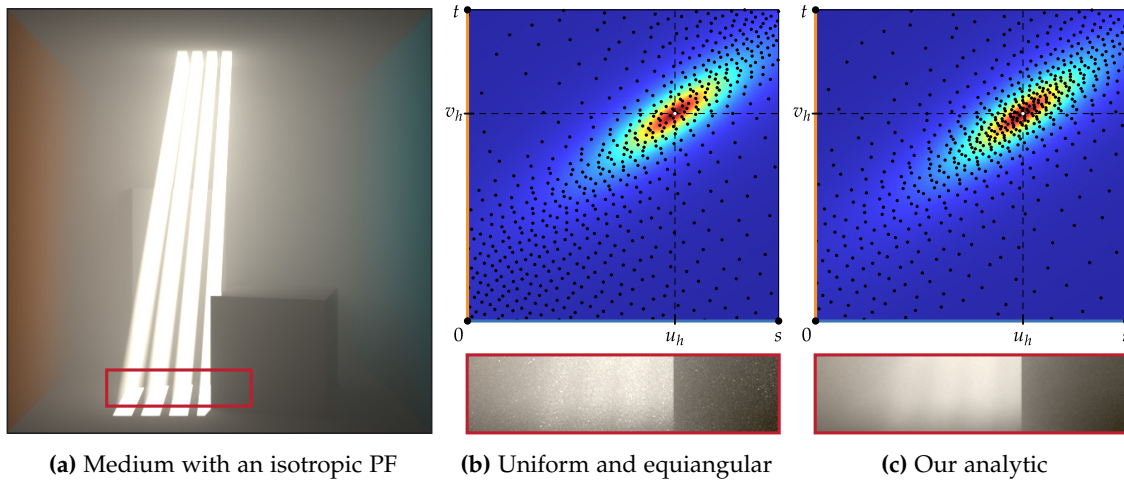
Figure 6.4: Different sampling strategies of media-to-media transport (multiple scattering) in a scene with an isotropic homogeneous medium (a). The straightforward approach is to uniformly sample a point along one axis and then importance sample the inverse-squared distance (i.e. equiangular sampling [Kulla and Fajardo 2012, Steinberg and Kalos 1971]) to this point along the other axis (b). We derive an analytic marginal PDF that allows to consider the inverse-squared distance already when sampling along the first axis (c). The inverse-squared distance is thus taken into account when sampling both points, resulting in better distribution of samples and less noise in the same amount of time.

We illustrate the effect of incorporating variation along the v axis in Figure 6.4. Notice that the sample distribution more closely matches the target $w(u,v)^{-2}$ density, which results in less noise in the rendered Cornell box image.

6.3.2   Anisotropic Scattering

The anisotropic case is unfortunately significantly more complex and, in this case, we aim to sample according to the *product* of the two phase functions as well as the inverse-squared distance:

$$p(u_i, v_i) \propto \frac{f_p(\omega_\mathbf{x} \to \omega)\, f_p(\omega \to -\omega_\mathbf{y})}{w(u,v)^2}. \tag{6.15}$$

In order to illustrate the impact of the anisotropic phase functions, we visualize this density function in Figure 6.5 in the same *uv*-domain as for the isotropic case.

Though importance sampling is possible for many commonly used phase functions, these routines consider the entire spherical domain. Even if we only considered the phase function at a single point light, the samples would need to be constrained to lie along an arbitrary spherical arc (e.g. the projection of the camera ray onto the point as shown in Figure 6.6.a). This type of domain restriction is uncommon hence, to our knowledge, such sampling routines do not currently exist for phase functions. Unfortunately, our context is even more complex since we need to consider not only the 1D camera ray domain, but also the 1D VRL domain and the product of the phase functions aligned along each of these two domains.

A common alternative to sampling the integrand exactly is to use multiple importance sampling

(a) Medium with an anisotropic PF   **(b)** Our analytic   **(c)** Our semi-analytic
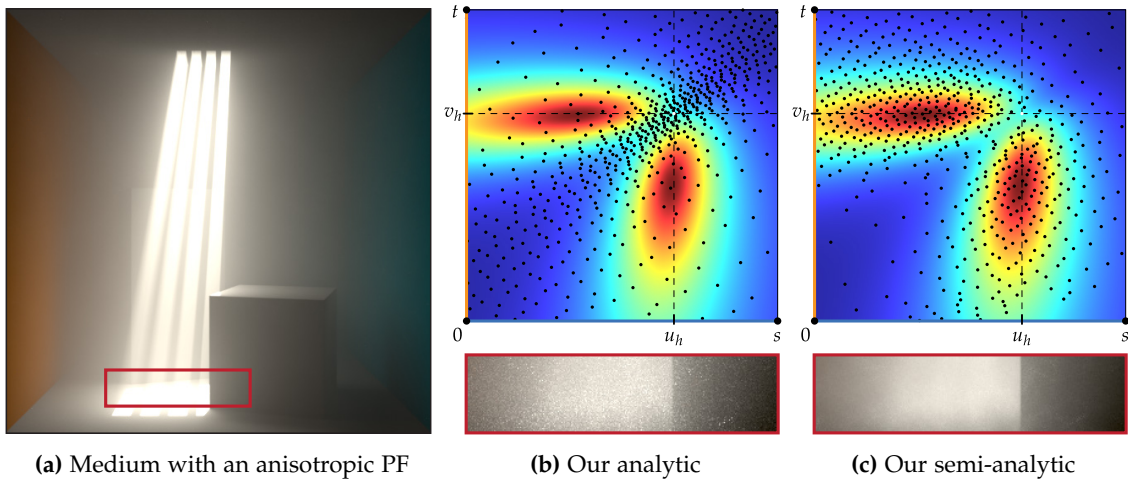
Figure 6.5: Importance sampling of multiple scattering in media with an anisotropic phase function. Since the phase function has high impact on the integration domain, taking into account just the inverse-squared distance results in high amount of noise (b). Therefore, we propose a semi-analytic approach that considers the product of the two phase functions and results in lower variance with only a marginal computational overhead (34% for this simple scene).

(MIS) [Veach 1997]. After analyzing and experimenting with this option (as we will discuss further in Section 6.4), we found that when estimating the integrals of interest with low sample counts, MIS results in significant variance. Therefore, we developed a specialized method to directly importance sample the product of the two phase functions and the inverse-squared distance, which produces better results than MIS. Our solution is a simple generalization of the two-step isotropic approach described in Section 6.3.1 and works the best with unimodal, axially symmetric phase functions.

We first proceed as in the isotropic case, choosing a location $\mathbf{x}_{v_i}$ along the VRL according to the inverse-squared distance to an infinite camera ray, see Equation (6.13). Note that we ignore the length of the camera ray and phase functions in this first step. The resulting sample point $\mathbf{x}_{v_i}$ can be interpreted as an anisotropic VPL along the VRL, see Figure 6.6.a.

In the second step, we wish to sample a location $\mathbf{y}_{u_i}$ along the camera ray according to the product of the inverse-squared distance and both phase functions (one at $\mathbf{x}_{v_i}$ aligned to the VRL, and another aligned to the camera ray, see Figure 6.6.b). By working in the angular domain about $\mathbf{x}_{v_i}$ (along the spherical arc formed by the camera ray's projection onto $\mathbf{x}_{v_i}$, shown in red), the inverse-squared distance term is incorporated implicitly.[2] We denote the remaining *product* of the two phase functions as $f_p^{uv}(u_i)$ and show a few examples in Figure 6.7 as solid red curves. Note that in the isotropic cases, this product is constant and sampling simplifies to the method in Section 6.3.1. Working in the angular domain has one additional advantage: we can easily handle (semi-)infinite camera rays, because the angle covered by an infinite ray in the angular domain is finite and equals π.

For anisotropic scattering, our solution is to construct a compact piecewise-linear PDF, which closely approximates the product f_p^{uv}. We first evaluate f_p^{uv} at a fixed number of variably-spaced points, $\theta_1 \dots \theta_M$, along the spherical arc. Then we fit a piecewise linear PDF to these evaluations,

---

[2]When we project uniformly sampled directions in the angular domain onto a ray, we obtain a distribution of points that is proportional to the inverse-squared distance

(a) Spherical arc   **(b)** Sampling in the angular domain
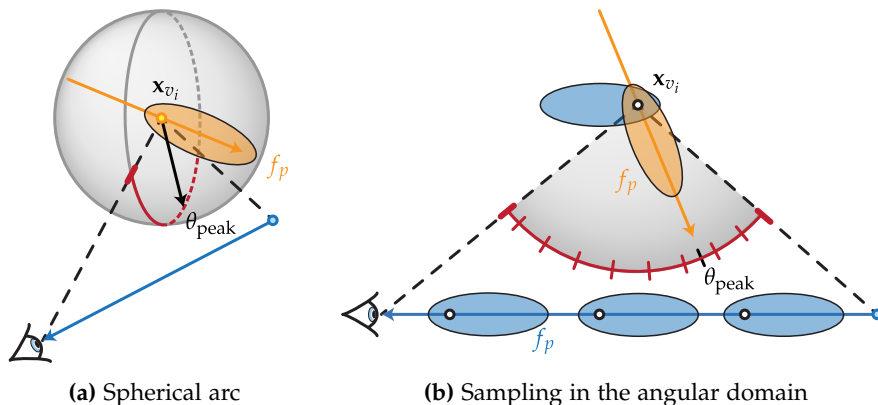
Figure 6.6: (a) Importance sampling the phase function (orange) at a point $\mathbf{x}_{v_i}$ requires generating samples only along the spherical arc (red) formed by the projection of the camera ray onto $\mathbf{x}_{v_i}$. (b) We importance sample the product of two phase functions (blue and orange) within this angular domain, viewed within the plane containing the camera ray and $\mathbf{x}_{v_i}$. In order to find the value of the phase function at the camera ray for a give angle θ, one can simply flip it and place at $\mathbf{x}_{v_i}$, which simplifies the evaluation of the product.
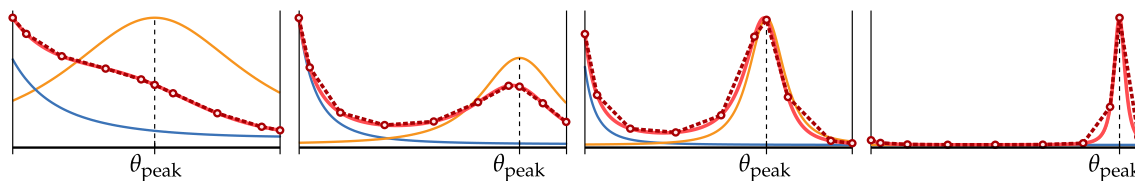
Figure 6.7: Four example configurations of anisotropic phase functions (Henyey-Greenstein with $g = 0.95$) plotted along the angular domain about a point $\mathbf{x}_{v_i}$ on the VRL. The product of the phase function along the camera ray (blue) and along the VRL (orange) results in the product (solid red) to which we fit a piecewise-linear PDF (dashed red). With only 10 vertices, we can reconstruct the product robustly for arbitrary configurations.

and distribute samples by integrating and inverting the corresponding piecewise-quadratic CDF. For this approach to be practical, we must fit f_p^{uv} accurately (to obtain noise-free results) and efficiently (since this operation is performed for every camera ray/VRL pair during rendering).

We experimented with many settings for M and found that even for very anisotropic phase functions, e.g. Henyey-Greenstein with $g = \pm 0.95$, a 10-point piecewise linear fit was sufficient to guarantee high accuracy (see the dashed poly-lines in Figure 6.7), if the vertices were placed adaptively to avoid missing important features. This is possible since we have some *a-priori* information about each of the phase functions, e.g. the direction of its peak(s), and we can thus predict the local extremes of the product and space the vertices accordingly. We detail the placement of the vertices in Appendix A.3.

Figure 6.5.c shows the resulting sample distribution in the *uv*-domain and a corresponding rendering of the Cornell box. Note that our fitting approach is not much slower than our analytic method for isotropic scattering (Figure 6.5.b), but handles the product of two arbitrary phase functions much better. In the simple Cornell box scene, we encountered only a 34% overhead due to constructing and sampling from the numerical CDF, and this overhead becomes negligible as the scene complexity increases.

$f_p(\omega_{\mathbf{x}} \to \omega) f_p(\omega \to -\omega_{\mathbf{y}})$ $w(u,v)^{-2}$ $\kappa_s(\mathbf{x}_v)\kappa_s(\mathbf{y}_u)$ $T(\mathbf{x} \leftrightarrow \mathbf{x}_v)\, T(\mathbf{y}_u \leftrightarrow \mathbf{y})$ $T(\mathbf{x}_v \leftrightarrow \mathbf{y}_u)$ $V(\mathbf{x}_v \leftrightarrow \mathbf{y}_u)$

(a) PF product / distance^2 **(b)** Scat. $\times$ trans.   **(c)** All terms   **(d)** Ratios
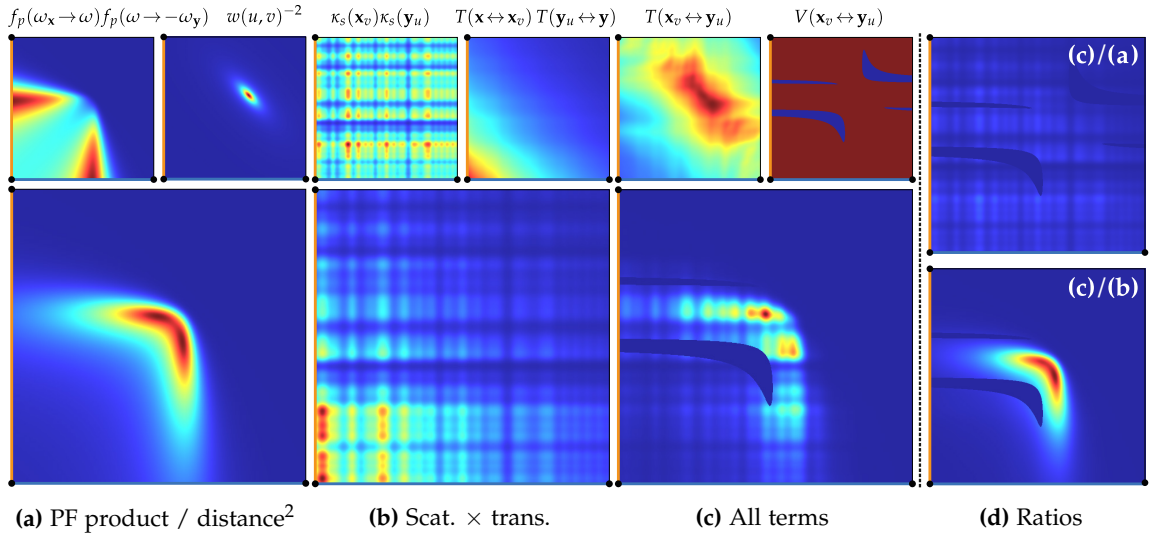
Figure (6.8): Visualizing terms from Equation (6.3) and their impact on the integrand for a scene with occluders, heterogeneous medium, and anisotropic scattering. We compare importance sampling of (a) the product of the phase functions divided by the squared distance (our method), to (b) the scattering and transmittance terms along the VRL and camera ray. The complete integrand is shown in (d), and the relative efficiency of sampling using the two PDFs is visualized in the last column, where the same scaling is used for both ratios to allow comparison.

6.3.3   Importance Sampling for Transport between a Ray and a Point

We can apply a largely identical procedure to sample Equation (6.4), the media-to-surface transport due to a VRL. In this case, we only have a 1D domain since one of the points (the surface point) is fixed, and hence we only need the second step, i.e. the numeric CDF, of our two-step procedure. As mentioned before, the problem of sampling a location along the VRL for a fixed point on the surface is dual to the surface-to-media transport; the only difference is the opposite direction of the light transport. For both L_s^m and L_m^s we proceed similarly, constructing a 1D numeric CDF (again with 10 vertices) for the product of the phase function of the ray and the scattering function of the point, and draw the samples on the ray from this distribution.

6.4   Analysis

In this section, we compare the performance of our importance sampling to other alternatives, such as distance sampling. We also analyze the singularities and demonstrate that spreading energy along lines leads to a provably lower degree of the singularity.

6.4.1   Importance Sampling Alternatives

We proposed to importance sample the transport in arbitrary media by first fixing a point on the VRL using a marginal analytic CDF, and then sampling the camera ray according to the product of phase functions divided by the squared distance. We also considered an alternative: importance sampling according to the transmittance and scattering coefficients along the VRL

(a) MIS   **(b)** Piece-wise linear CDF   **(c)** Reference
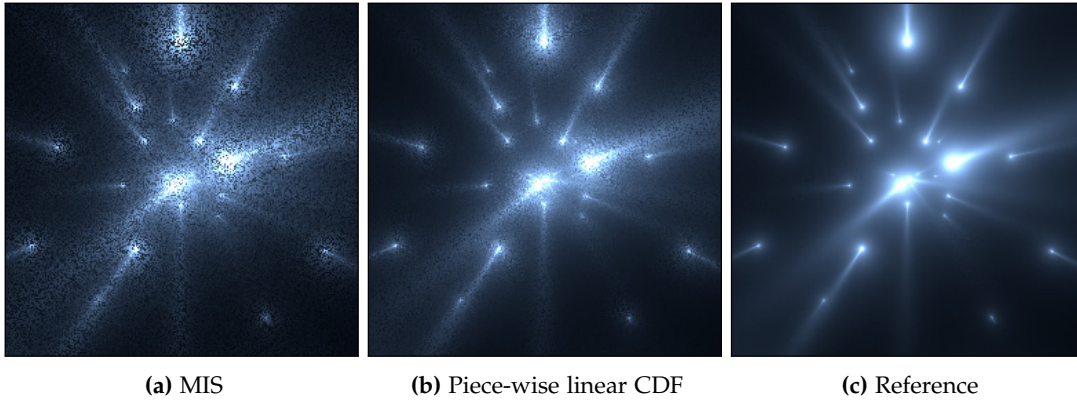
Figure 6.9: Comparison of sampling in the second stage of our method, i.e. according to the phase functions along the camera ray and the VRL. In the first step, we sampled few points along VRLs (always the same and very few only, for illustration purposes). Multiple importance sampling using the balance heuristic (a) performs much worse than a piecewise-linear approximation (with 10 vertices) of the product of both phase functions (b). (c) shows a "reference" solution where the PDF was constructed using 1000 vertices. The remaining variance in (c) is due to the transmittance of the medium.

and the camera ray. Since their product is separable, we can split the 2D PDF into a product of two 1D PDFs (one along each of the rays), avoiding an expensive construction of a numeric PDF for the entire 2D domain.

In the top row of Figure 6.8, we visualize all the individual terms of the integrand from Equation (6.4). Then we compare the outcomes of sampling from two PDFs for: (a) the product of phase functions divided by the squared distance, and (b) the product of scattering and transmittance along the VRL and the camera ray. The complete integrand, i.e. the product of (a), (b), and the visibility and transmittance along the connecting segment, is shown in Figure 6.8.c. All plots to the left of the dashed line are individually normalized.

The last column depicts the ratios of the integrand to each of the PDFs. With ideal importance sampling, this ratio would be constant, i.e. the density is just a scaled version of the integrand. For our method (a) the ratio varies only due to the scattering and transmittance, which have relatively low impact on the integrand. In contrast, the ratio of (c) to (b) has much higher variation. This clearly demonstrates that considering the phase function product divided by the squared distance is crucial for efficient numeric evaluation, and supports our choice for importance sampling described in Section 6.3. We further compare the results obtained with different sampling schemes in Section 6.6.

6.4.2   Multiple Importance Sampling

MIS is a common strategy to importance sample complex integrands by generating samples according to different subsets of the integrand [Veach 1997]. We evaluated the feasibility of an MIS approach for the second stage of our VRL method, i.e. after choosing a point on the VRL. We are not aware of appropriate sampling routines for generating samples along the camera ray according to variation of the phase function towards a fixed point on the VRL, and analogously along a VRL. However, we can use tabulated representations of the two PDFs, which we can both integrate and sample using the inversion method.

(a) Point-to-point	(b) Point-to-line

(c) Line-to-point	(d) Line-to-line
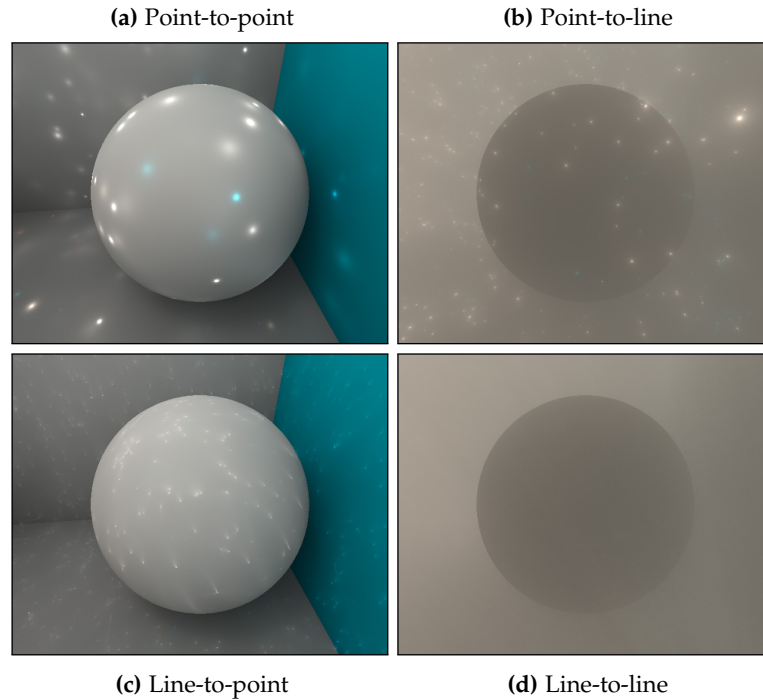
Figure 6.10: We analyze four different cases of energy transfer; for illustration we use an equal number (10k) of VPLs and VRLs to render a sphere in participating media. The left column shows surface illumination only. Top row: light transport computed using VPLs; bottom row: VRLs illuminating surfaces and media.

The results of this experiment, where we randomly chose a strategy and combined the results using MIS, are shown in Figure 6.9. We tried both the power and balance heuristic, where the latter resulted in less noise. For comparison, the right image in the figure shows the result of directly sampling the product (also computed here in a brute force tabulated fashion). This constitutes the best we could possibly hope to achieve, the remaining noise is due to the transmittance, which is not considered here during sampling. We see that combining strategies using MIS, which may initially seem like a good choice, results in significantly more noise than sampling the product directly. This is because the effective PDF of MIS is actually a linear combination of the two PDFs, and not the actual product. Furthermore, analytic sampling would require deriving an efficient arc sampling technique for every desired phase function. Due to these drawbacks, we instead resort to the more general but fast approach for sampling the product directly, without the need for MIS.

6.4.3 Singularities

Similar to VPLs, we obtain VRLs directly from photon tracing. However, the distribution of the energy along VRLs suggests that artifacts should be less pronounced compared to VPLs, where the energy is concentrated at a finite number of discrete points. In the spirit of Jarosz et al. [2011a], we analyze the efficiency of simulating the light transport with different geometric primitives, namely points and lines. In Figure 6.10, we compare the media-to-surface transport with VPLs (i.e. point-to-point) and VRLs (i.e. line-to-point), and the media-to-media transport with VPLs (i.e. point-to-line) and VRLs (line-to-line).

It can be seen that VPLs illuminating media (b) and VRLs lighting surfaces (c) cause less distracting artifacts than VPLs illuminating surfaces (a). When using VRLs to illuminate the medium (d), the artifacts are not visible at all.

In Appendix A.4, we derive the order of the singularity for the aforementioned types of transport. We observe that point-to-point transfer expectedly suffers from the strongest singularity. In the line-to-point and point-to-line transfer, the singularity is effectively reduced by integrating along the line. This trend continues when evaluating line-to-line transfer thanks to the additional integral, except for the special case where both lines are parallel.

In the same spirit as most VPL methods, we could simply bound the VRLs' contributions to avoid any artifacts. As we have seen, these are generally lower then for VPLs, and this means that bounding would remove less energy from the solution, and consequently also has less impact on material appearance [Křivánek et al. 2010]. However, we decided not to clamp in all our renderings, as the artifacts vanish quickly when the number of VRLs increases. We leave the development of bias compensation techniques for VRLs as future work.

6.5   Algorithm Overview

6.5.1   Light Path Splitting

The best choice of methods for computing light transport varies with the respective characteristics of the individual phenomena. For example, progressive photon beams (PPB) are well suited for rendering volumetric caustics, but have no significant advantage[3] over traditional photon mapping for media-to-surface transport (the endpoints of the beams are simply surface photons). VRLs, on the other hand, yield significantly better results for multiple scattering and require less virtual light sources (than VPLs) for surface lighting. Fortunately, both PPB and VRLs can be created from the same photon tracing step and naturally fit complementarily into one rendering framework.

6.5.2   Implementation Details

We implemented our algorithm in a hybrid CPU-GPU framework. We first trace random-walk paths from light sources, scattering photons at surfaces and in the media using a CPU ray-tracer. This step corresponds to photon shooting in standard photon mapping [Jensen and Christensen 1998]. Photons form VPLs or VRLs where, in the latter case, the photon path segments are used instead of the photon locations. In addition to the next event estimation from VRLs and VPLs, we apply progressive density estimation to the photon segments (PPB) [Jarosz et al. 2011b] and photon points (PPM) [Knaus and Zwicker 2011] to simulate volumetric single scattering and caustics, and direct illumination and surface caustics, respectively. As proposed by Jarosz et al. [2011b], we render directly visible single-scattered volume caustics with rasterization and use CPU ray-tracing to handle beams only visible after specular reflection and/or refraction (e.g. all lighting in the glasses in Figure 6.11.a).

---

[3]Some techniques demonstrate that photon beams can be beneficial even for surface illumination, e.g. for reducing the density estimation bias [Havran et al. 2005], but these benefits are in our context rather marginal.

In order to compute media-to-media transport (which includes multiple scattering), we evaluate Equation (6.3) for each VRL/camera ray pair (all of which are uploaded to the GPU) by sampling according to our product importance sampling routine. In order to resolve the visibility in a robust manner, we use ray tracing, as opposed to shadow mapping that introduces bias, and whose performance benefits diminish with increased scene complexity or in the presence of heterogeneous media. Ray tracing also allows us to integrate more readily into existing physically-based rendering frameworks. We employ Aila and Laine's [2009] efficient GPU ray-tracer.

Evaluating the transmittance terms in Equations (6.3) and (6.4) can easily become a bottleneck in the case of heterogeneous media. We improve the performance by precomputing the transmittance along camera rays and VRLs: using Woodcock tracking we sample and cache a number of distances (typically 16) that are later used to approximate the transmittance along the ray in an unbiased manner (see the paragraph about Woodcock multi-tracking in Section 2.6.8). Notice that transmittance between sample points on the VRL and the sample points on the ray (or the surface point) cannot be cached. In this case, we use either fewer Woodcock samples or employ fast ray-marching, which improves the performance at the cost of introducing a small but mostly imperceptible amount of bias into the transmittance computation.

We wrap all of the methods in a progressive framework, providing interactive previews that converge to ground truth all in the same renderer. We also utilize a Russian roulette, which is based on the minimum distance between the virtual light and the query primitive (e.g. between a VRL and a camera ray), to probabilistically prune virtual lights with low contribution.

6.6 Results

We compare the quality and performance of our algorithm against VPLs and PPB on an Intel Core i7 CPU @ 2.8GHz with 8GB RAM and an NVIDIA GTX 470. The CPU ray-tracer and density estimation for PPM and PPB are parallelized over all CPU cores. All our results use the Henyey-Greenstein phase function and show all the energy of interest, i.e. no bounding is used. To ensure fair comparison, we use the same framework to compare to VPLs, and thus both many-light techniques benefit from the same acceleration structure and similar code paths.

Figure 6.11.a shows the Fruit Juice scene with glasses of orange and grapefruit juice rendered at 512×512 resolution. We simulate anisotropic scattering for both materials. The parameters of the media are $\kappa_t = (0.41, 0.72, 5.18)\text{cm}^{-1}$, $\kappa_s = (0.36, 0.50, 0.18)\text{cm}^{-1}$, and $g = 0.5$ for the orange juice; and $\kappa_t = (0.41, 0.95, 4.73)\text{cm}^{-1}$, $\kappa_s = (0.45, 0.32, 0.23)\text{cm}^{-1}$, and $g = 0.6$ for the grapefruit juice.

In Figure 6.12, we show an equal time comparison for media-to-media transport computed using VRLs, PPB, and VPLs. Note that PPB is a biased (but consistent) method and yields darker results prior to convergence; VPLs (without bounding) suffer from bright pixels, even after long render times. In contrast, our method produces unbiased results and does not suffer from concentrated bright pixels. For the final image in Figure 6.11.a, we combine our multiple scattering result with a single scattering (volume caustic) component using PPB, and surface illumination and surface caustics, computed using PPM.

The Smoky Room scene (see Figure 6.11.b) is filled with a heterogeneous isotropic medium (simulated using Perlin noise) whose density decreases with height. Figure 6.13 shows media-to-media transport only, i.e. paths whose last two interactions before reaching the camera happened

<div align="center">

(a) Fʀᴜɪᴛ Jᴜɪᴄᴇ   **(b)** Sᴍᴏᴋʏ Rᴏᴏᴍ

</div>

Figure 6.11: Two scenes used for comparing the performance of our VRL algorithm to VPLs and PPB.

in the medium. At equal time, VRLs yield higher quality results than both PPB and VPLs. Despite the fact that we accelerate PPB using hardware rasterization, it requires millions to billions of beams to sufficiently cover the entire image, and the convergence rate is thus low. PPB works extremely well for volumetric caustic illumination since the beams are concentrated, reducing fill rate, while ensuring high density. For multiple scattering, photon paths quickly become incoherent, which introduces low-frequency noise into the local density estimate. VRLs do not rely solely on local density information and can therefore obtain higher quality multiple scattering using significantly less beams. In contrast to VPLs, our method does not suffer from visible artifacts due to the singularities, trading this for a slight uniform noise distributed evenly across the image. This is because we sample each VRL/ray pair with only one random sample. The final solution consists of media-to-media (100s) and media-to-surface (600s) illumination computed using VRLs, and single scattering (100s) and surface-to-media (300s) light transport computed using PPB.

Figure 6.14 shows media-to-surface illumination, i.e. light paths with the last two scattering events being in volume and then on a surface. When using VRLs, the media-to-surface transport converges faster than with VPLs. Only very little bounding, if any, is required. Importantly, no bounding seems to be required for the media-to-media transport. This is an important improvement over VPLs since bounding removes energy and changes the appearance of materials and media. Bias compensation techniques can recover the lost energy, but at significant additional expense.

In Figure 6.15 we compare different sampling strategies for evaluating the VRL-ray transport using a single sample. For each VRL and camera ray we construct a 1D piecewise linear PDF (with 100 vertices) for the transmittance ($\times$ the scattering coefficient, cf. Figure 6.8.b) along the ray, and sample each ray independently. This leads to much higher variance than sampling according to the product of phase functions divided by the squared distance (cf. Figure 6.8.a), which has significant impact on the integrand from Equation (6.3). Admittedly, there can be situations when sampling according to the scattering $\times$ transmittance can also improve the efficiency. For cases where this additional expense is warranted, the individual sampling strategies can be combined using MIS.
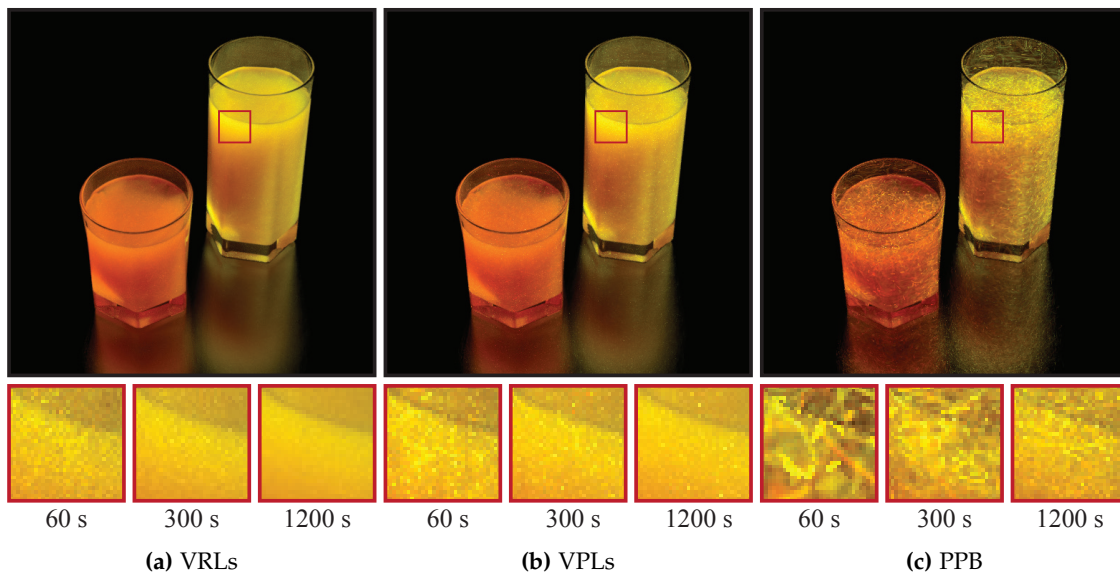
Figure 6.12: Renderings of multiple scattering (media-to-media transport) in the FRUIT JUICE scene after 10 minutes. While VRLs (a) provide a high quality result, both the previous state-of-the-art approaches for general media (virtual point lights, (b); progressive photon beams, (c)) contain significant artifacts. The insets show the quality after 1, 5, and 20 minutes.

	FRUITY JUICE[1]	SMOKY ROOM[2]
Photon (VRL) shooting	6%	2%
Ray tracing[1] / Rasterization[2]	55%	1%
Importance sampling	9%	4%
Visibility test	23%	52%
Light transport	7%	42%

Table 6.1: Performance breakdown of the media-to-media transport.

In the supplemental video, we compare the convergence behavior of our progressive algorithm to both PPB and VPLs. Another common problem with VPL techniques is temporal flickering due to stochastic under-sampling the indirect lighting. The increased sample density provided by VRLs significantly diminishes these artifacts even when using fewer virtual lights.

In Table 6.1 we show the performance breakdown for computing media-to-media transport. Our importance sampling approach is efficient, occupying between 4–9% of the total render time whereas 23–52% of the time is spent on evaluating visibility on the GPU, which would be even higher if performed on the CPU.

6.7 Discussion and Possible Improvements

Our VRL is a new lighting primitive that is well-suited for computing unbiased media-to-media and media-to-surface light transport. It can be easily combined with other techniques, such as VPLs, PPB, and PPM, into a powerful and efficient rendering framework. In the following, we relate VRLs to other techniques and discuss possible improvements.

(a) VRLs (101 sec.)   **(b)** VPLs (102 sec.)   **(c)** PPB (102 sec.)
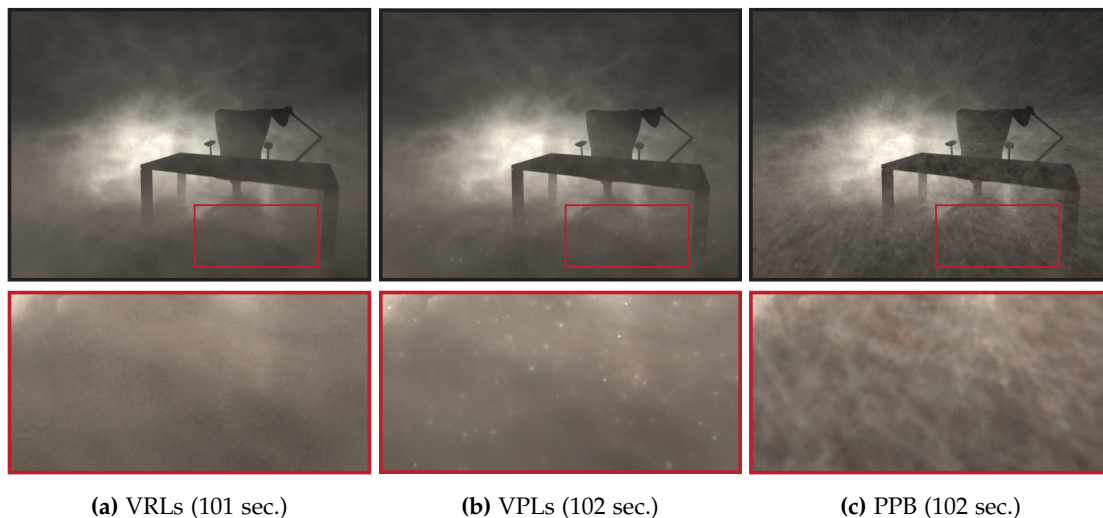
Figure 6.13: Media-to-media transport in a heterogeneous media in the SMOKY ROOM scene. While VPLs (b) suffer from singularities, and PPB (c) would require many more photon beams to sufficiently fill the scene, our method (a) provides artifact-free results faster, benefiting from explicit gathering from the light path segments and denser sampling of the space, respectively.

(a) VRLs (600 s)   **(b)** VPLs (600 s)
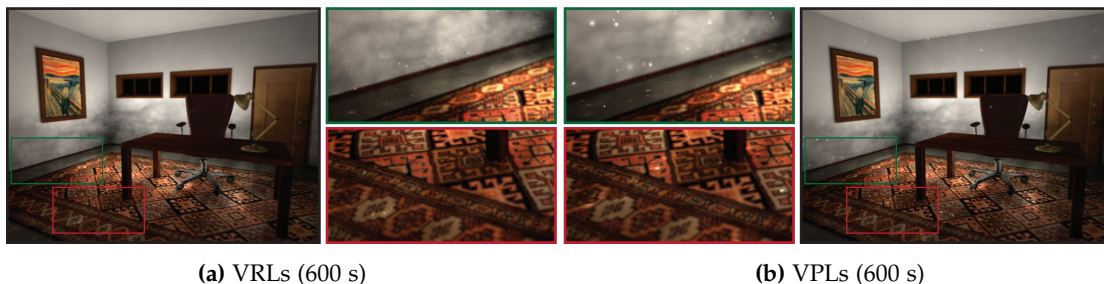
Figure 6.14: These images show media-to-surface transport in the SMOKY ROOM. Some regions are darker due to absorption and out-scattering in the medium. The close-ups reveal the splotchy artifacts of VRLs (a) and VPLs (b), both without bounding.

Analytic Integration, Duality. Equation (6.4) and the inner integral of Equation (6.3) can also be seen as the dual of the airlight integral, for which closed-form solutions exist when constrained to homogeneous media and no occluders [Pegoraro and Parker 2009, Sun et al. 2005]. We have considered leveraging these analytic methods; however, we found that each analytic integration is quite expensive (especially when incorporating anisotropic scattering [Pegoraro et al. 2009; 2010]), and ultimately the homogeneous/visibility assumptions are too restrictive for the general setting we consider. Furthermore, by solving the inner integral of Equation (6.3) in isolation, we would actually be considering only a small portion of a larger 2D integration problem. Some of our sampling schemes (e.g. importance sampling the inverse squared distance); however, are inspired by previous techniques for solving the airlight integral numerically.

General Bidirectional Algorithm. We believe that our idea of sampling transport between rays is an important first step towards developing new bidirectional MC approaches for rendering in the presence of participating media. At the moment, we consider only camera rays and spec-

| (a) Transmittance | (b) Scat. × trans. | (c) PF prod. / distance² | (d) Reference |
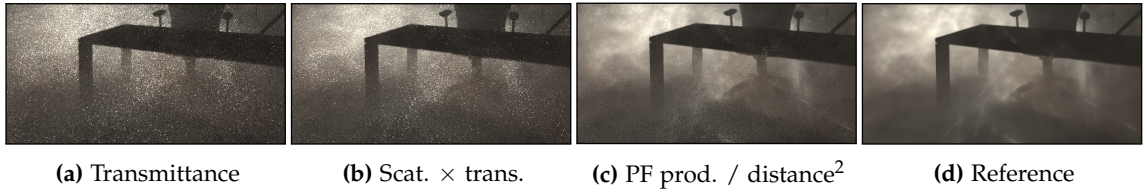
Figure 6.15: Importance sampling strategies for the VRL/ray transport: in (a) and (b) we sample each of the rays independently according to the transmittance and transmittance × scattering. The resulting images suffer from high amount of noise. We propose to sample according to the phase function product divided by the squared distance (c), which leads to significantly less noise, verifying the theoretical analysis from Figure 6.8.

ular camera paths. Formulating a general bidirectional algorithm evaluating transport between arbitrary segments of light and camera paths is an interesting area for future work.

Visibility. In our implementation we did not use shadow mapping, which is often used to accelerate VPL rendering. We opted for ray tracing since shadow mapping introduces bias and prevents a general rendering framework. Moreover, standard shadow maps cannot be used for heterogeneous media. Nevertheless, dedicated shadow techniques for linear lights exist [Heidrich et al. 2000] and seem to be worth investigating. Furthermore, incorporating mutual visibility into the importance sampling could possibly speed-up the convergence for highly occluded scenes. We believe that our approach can be combined with ideas from Georgiev et al. [2012], though this would warrant its own in-depth investigation.

Singularities. We have seen that singularities are virtually unnoticeable when using VRLs for volume rendering, yet they can remain visible on surfaces; however, even in this case they are less noticeable than singularities on surfaces due to VPLs. This empirically supports our proof of progressively lower degree of singularities (see Appendix A.4). Though we are able to obtain unbiased images using relatively few VRLs, convergence is slower for volume-to-surface transport. In the next chapter, we demonstrate how to avoid the remaining structured artifacts by blurring the energy of VRLs spatially.

Long vs. Short VRLs. When generating VRLs, we have two options how to define their length. In our examples, we used "long" ray lights, i.e. rays that extend beyond the next vertex of the photon path to the nearest surface point. The other option is to consider the actual length of the photon path segment. This is conceptually similar to the analog and non-analog MC estimators described e.g. in [Kalos 1963]. If the length is determined by distance sampling (e.g. Woodcock tracking in heterogeneous media), we can omit the transmittance term $T(\mathbf{x} \leftrightarrow \mathbf{x}_t)$ along the ray light. This is because the transmittance $T(\mathbf{x} \leftrightarrow \mathbf{x}_t)$ is already "encoded" in the length of the "short" ray lights. We experimented with both approaches. It seems that for thin and moderately dense media the long ray lights perform better, since they by definition provide higher sampling density. This advantage diminishes in highly scattering media with short mean free path, where the inverse-squared distance-based sampling often samples far away from the origin of the VRL, where almost no energy is transported. In such cases, short VRLs enforce the samples to be within the sampled free path and can thus provide better convergence.

Clustering of VRLs.   Lightcuts clusters millions of VPLs and hierarchically chooses the best representatives for each gather point. Though we do not explicitly cluster virtual lights, each VRL/camera ray pair can be seen as a continuous family of light and gather point pairs. By importance sampling the resulting 2D domain, we are effectively choosing the best representative from this family for each location along a camera ray. Nonetheless, it would be interesting to see if explicit clustering of VRLs, in the spirit of Lightcuts [Walter et al. 2006; 2005; 2012], is possible.

Reconstruction with Sample Reuse.   Our *uv*-domain shares similarities with epipolar slices. This geometric interpretation has previously been used to accelerate volumetric shadows [Baran et al. 2010, Engelhardt and Dachsbacher 2010], and sampling and reconstruction of motion blur and depth-of-field [Hachisuka et al. 2008a]. It would be interesting to consider more sophisticated, but perhaps biased reconstruction of the 2D *uv*-slice, or whether occluders could be directly rasterized into this warped domain.

Line-Sampling of Area Lights   Lastly, VRLs can also be used to compute single scattering or direct illumination. If the light source is planar, we can distribute VRLs over its area and use the emission profile of the light source as the VRL's phase function. The volumetric and surface illumination due to these lights then accounts for single scattering and direct illumination. Our preliminary experiments show that line sampling of area lights outperforms point sampling, and with our semi-analytic sampling, which in this case accounts for the IES profile and the scattering function, outperforms in several cases the current state of the art in multiple importance sampling of area lights and BRDFs. We believe that this direction is worth investigating in the future. One could also distribute VRLs using line segment sampling [Sun et al. 2013] to preserve desirable sampling properties.

6.8   Conclusion

We presented a new lighting primitive—virtual ray lights—for unbiased many-light rendering of indirect illumination in, and from, participating media. VRLs are created from path segments of photon random walks in the medium, and we showed how to compute their contribution to entire camera rays through the medium and to surfaces. VRLs are less prone to singularities and yield high quality multiple scattering faster than previously introduced techniques.

Virtual Beam Lights

A clever person solves a problem.
A wise person avoids it.

— ALBERT EINSTEIN (1879–1955)

In this chapter, we extend the previously introduced virtual ray lights technique and present a formulation that removes the singularity completely. The resulting algorithm, called *progressive virtual beam lights* (VBLs), "inflates" the infinitesimal ray lights into beam lights with finite thickness. The power of the original VRL is uniformly distributed within the beam, which eliminates the singularity in the integral. The bias that VBLs introduce can be treated in a similar spirit to numerous recently published techniques, e.g. Hachisuka and Jensen [2009], Damnertz et al. [2010], or Knaus and Zwicker [2011], that propose progressive formulations producing convergent results while maintaining a fixed memory footprint. As such, we formulate the rendering algorithm progressively, ensuring that the bias approaches zero in the limit. Compared to VRLs, VBLs handle anisotropic scattering and moderately glossy surfaces more robustly. Visually pleasing images with no visible artifacts can thus be obtained in shorter time, while still converging to the ground truth in the limit.

In addition to transport within and from participating media, we also focus on surface-to-media illumination, i.e. photon paths that reach the eye by bouncing off a surface and then scattering in the medium. We devise several practical schemes for importance sampling the various transport contributions between camera rays, light rays, and surface points.

7.1   From Rays towards Beams

In this section, we review previous work that inspired the idea of inflating ray lights into beam lights. The concept of blurring energy over area has been intensively used in photon mapping [Jensen 1996]. Here, the idea is to splat the energy of infinitesimal photon points over nearby surfaces and thus allow for an efficient, although biased, integration. Since splatting requires quantizing the scene into a number of finite elements (e.g. world-space or screen-space patches), to which we can splat the energy, most photon mapping techniques redistribute the energy via density estimation using a kernel with finite support. In either case, the integration suffers from a systematic error that degrades the quality of rendered images around sharp or thin features.

In order to reduce the error, one can formulate the estimation progressively, reducing the support

(a) VPLs		**(b)** VSLs		**(c)** VRLs		**(d)** VBLs
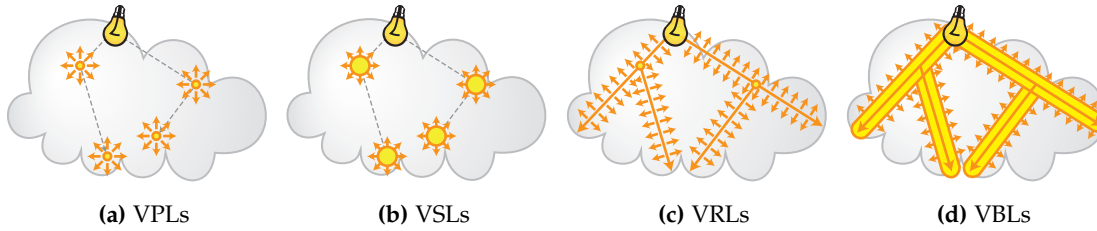
Figure 7.1: We are inspired by VSLs (b), which expand VPLs (a) into spheres with finite radius. We similarly expand VRLs (c) into VBLs (d) with finite thickness. This removes the singularity and reduces render times for indirect lighting to and from surfaces and media.

of the kernel over time [Hachisuka and Jensen 2009, Hachisuka et al. 2008b]. If this is done carefully, the bias vanishes in the limit and the estimation becomes consistent [Kaplanyan and Dachsbacher 2013a, Knaus and Zwicker 2011].

In the context of many-light algorithms, the idea of spreading the energy of a VPL over nearby surfaces has been explored by Hašan et al. [2009]; please see Section 3.3.4 for an overview of the technique. In a similar spirit, we can take each virtual ray light and distribute its energy throughout the nearby volume, see Figure 7.1. We also leverage the observation that VSL rendering (or more precisely, rendering with photon lights [Hašan et al. 2009]) can be formulated as photon mapping with final gathering, and vice versa. This allows us to leverage the previously mentioned progressive, bias-reduction techniques, namely the progressive photon beams (PPB) [Jarosz et al. 2011b], and formulate a consistent rendering algorithm based on progressively shrinking virtual beam lights.

7.2 Light Transport with VBLs

Given a virtual ray light and a length r, we define a virtual beam light as a convolution of the ray light with a sphere of radius r. The power of each point p on the VRL is spread uniformly over the volume encapsulated by the sphere centered at p. The emissive volume of the VBL thus forms a cylindrical region with spherical caps. It is worth noting that with this definition, the volumetric density of flux is not uniform; points inside the two hemi-spherical regions centered at the end points of the finite ray receive less energy. Compared to a strictly uniform distribution, this formulation turns out to be advantageous as it simplifies the integration: the VBL can be seen as a VSL swept along the ray light. Instead of evaluating the transport between a point (or a ray) and a cylindrical region, we can simply sample points along the VBL and then integrate over the solid angle subtended by the VSLs.

In Chapter 6, we introduced four types of light transport w.r.t whether the shading point resides on a surface or in a medium, and whether the incident light at this point arrives from another surface or medium. For completeness, we could also consider light arriving directly from light sources. However, this transport is substantially simpler and we thus leave direct illumination and single scattering out, and focus solely on indirect illumination, namely the surface-to-surface L_s^s, media-to-surface L_s^m, surface-to-media L_m^s and media-to-media L_m^m transport paths. The first row in Figure 7.2 shows renderings of these four transport paths computed using unbounded VPLs and VRLs. The visual artifacts due to the singularity in the integrand can be avoided by
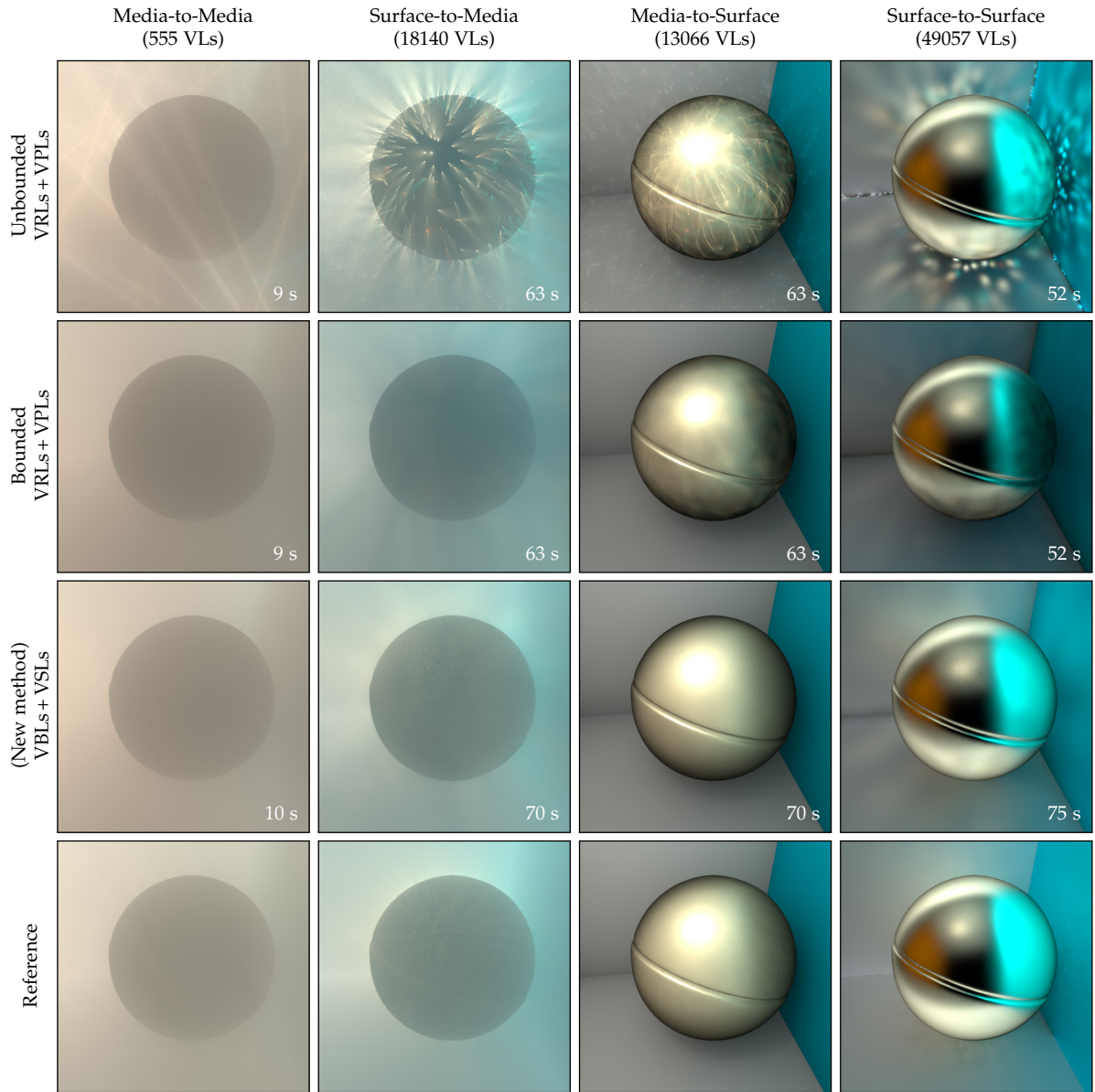
Figure 7.2: Comparison between unbounded and bounded VRLs + VPLs, VBLs + VSLs, and the reference on different transports paths. Images in each column (except for the reference) were computed using the same number of virtual lights.

bounding; however, this results in energy loss and modifies material appearance, as shown in the second row of Figure 7.2.

The VRL method approximates the light transport by first performing a random-walk photon simulation and converting the segments of the random-walk paths into linear light sources. In Chapter 6, we introduced several importance sampling strategies that can be applied to L_s^m, L_m^m, and L_m^m (L_s^s does not involve any integration as it is a deterministic connection between the endpoints of the VRL and the camera ray).

(a) Media-to-Media L_m^m

(b) Media-to-Surface L_s^m

(c) Surface-to-Media L_m^s

(d) Surface-to-Surface L_s^s
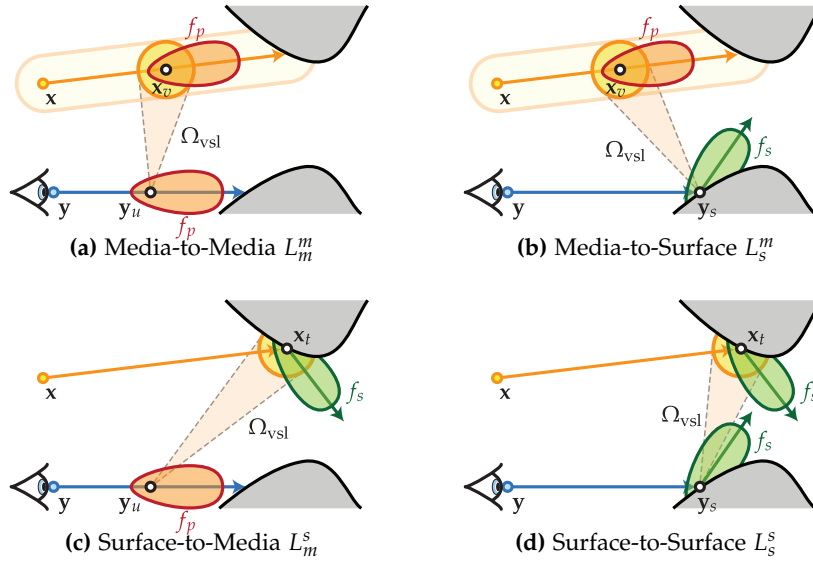
Figure 7.3: Visualization of each of the four light transport paths. For L_s^m and L_m^m, VBLs are illustrated as VSLs swept along the VRL.

In the following sections, we show how the integration domain changes when inflating VRLs into VBLs, and detail the necessary changes to importance sampling described in Section 6.3 to preserve low variance estimation. As the VBL is formulated as a swept VSL, we build on multiple importance sampling developed for VSLs [Hašan et al. 2009], extending it to the additional dimension (i.e. the length of the VBL). Additionally, we show how to formulate the rendering algorithm progressively to achieve consistent estimation. The resulting images, see Figure 7.2, do not suffer from distracting artifacts, preserve the overall energy, and are thus visually closer to the reference than the previously mentioned techniques.

7.2.1   Media-to-Surface Transport

When rendering with VBLs, computing L_s^m, i.e. radiance that scatters in media and reaches the camera by bouncing off $\mathbf{y}_s$, requires expanding every point along the VRL into a VSL (see Figure 7.3.b). This modifies Equation (6.4) to:

$$L_s^m(\mathbf{y} \leftarrow \omega_\mathbf{y}) = T(\mathbf{y} \leftrightarrow \mathbf{y}_s) \int_0^t \kappa_s(\mathbf{x}_v) T(\mathbf{x} \leftrightarrow \mathbf{x}_v) T(\mathbf{x}_v \leftrightarrow \mathbf{y}_s) V(\mathbf{x}_v \leftrightarrow \mathbf{y}_s) \, L_s^{\text{vsl}}(\mathbf{y}_s \rightarrow \mathbf{y}) \mathrm{d}v \qquad (7.1)$$

where $L_s^{\text{vsl}}(\mathbf{y}_s \rightarrow \mathbf{y})$ is the fraction of irradiance at $\mathbf{y}_s$ due to the VSL at $\mathbf{x}_v$, which is reflected towards $\mathbf{y}$:

$$L_s^{\text{vsl}}(\mathbf{y}_s \rightarrow \mathbf{y}) = \frac{\Phi}{\pi r^2} \int_{\Omega_{\text{vsl}}} f_p(\omega_\mathbf{x} \rightarrow -\omega) f_s(-\omega \rightarrow \mathbf{y}_s \rightarrow -\omega_\mathbf{y}) D_{\mathbf{y}_s}(\omega) \mathrm{d}\omega. \qquad (7.2)$$

Notice that none of the equations above contains the inverse squared distance; it was replaced by the integration over the solid angle Ω_{vsl} subtended by the VSL. The transmittance and visibility between points on the VSL and the surface point $\mathbf{y}_s$ are indeed functions of ω. To make the integration tractable we introduce similar assumptions to those in [Hašan et al. 2009] and
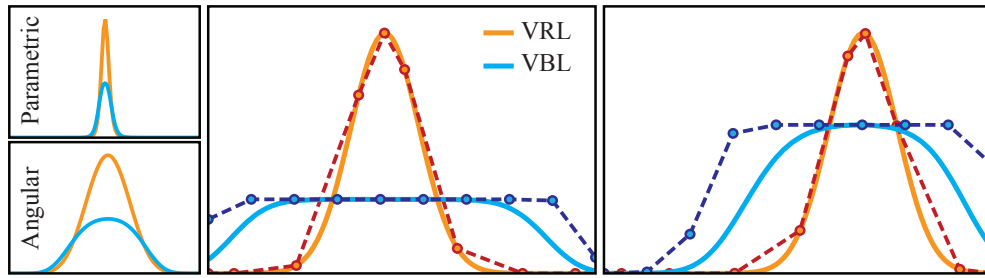
Figure 7.4: A VRL's (orange) and VBL's (blue) contribution to a shading point. Working in the angular domain smooths out the peak and implicitly accounts for the inverse-squared falloff. The middle and right plots show two examples of piecewise-linear PDFs for VRLs and VBLs (dashed) compared to the actual integrand (solid).

approximate the functions with a single, point-to-point evaluation between $\mathbf{x}_v$ and $\mathbf{y}_s$. As such, the two terms can be factored out from the integral Equation (7.2).

In order to provide some physical intuition about Equation (7.1), one can reason about the integral as a special case of radiance estimate [Jarosz et al. 2008a], which interprets the volumetric sphere as a disc, of radius r, that always faces the gather direction ω (see Figure 7.5.a). We can also interpret the VSL as a point emitter enclosed in a perfectly transmissive diffuse sphere, or as a spherical volumetric emitter (please see Appendix A.5 for details). All these interpretations result in the same set of equations that we need to evaluate numerically.

Figure 7.4 plots the contributions of a VRL (orange) and a VBL (blue) to a single shading point. The figure also illustrates the different shapes of the integrand in the angular (about the shading point) and parametric (along the ray/beam light) domains. Notice how the fact that the angular domain implicitly incorporates the inverse squared distance fall-off reduces high-frequency variations, making the sampling less prone to miss important parts of the integrand.

In order to importance sample the location $\mathbf{x}_v$ on the VRL (i.e. to sample the VSL location) we construct a piecewise-linear PDF in the angular domain about $\mathbf{y}_s$, much like in the case of VRLs (see Section 6.3.2). Figure 7.4 visualizes the piecewise-linear PDF that we designed for sampling the VRL transport. Unfortunately, this PDF does not match the VBL response well and we thus need to find a better approximation of the integrand.

At each vertex of this PDF we would ideally evaluate the contribution of a VSL placed at that location. However, integrating the contribution (e.g. with Monte Carlo integration of Equation (7.2) over many directions) would make the construction of the piecewise-linear PDF prohibitively costly and noisy due to possibly glossy BRDFs and phase functions.

Instead, we observe that while *over*estimating the integrand response in the PDF is suboptimal, it does not lead to nearly as much variance as *under*estimation. At each PDF vertex, we therefore quickly approximate the maximum contribution of the integrand's terms within Ω_{vsl}. Thus, for every PDF vertex, we (see Figure 7.5.b):

- find the direction ω_{max} where incident light at the surface contributes most to the reflection according to the BRDF (usually the axis of the primary lobe of the BRDF),

- find the direction ω'_{max} within the cone of directions subtended by the VSL, which requires least rotation to ω_{max}, and

(a) VSL integration as BRE   **(b)** Computing ω'_{max}
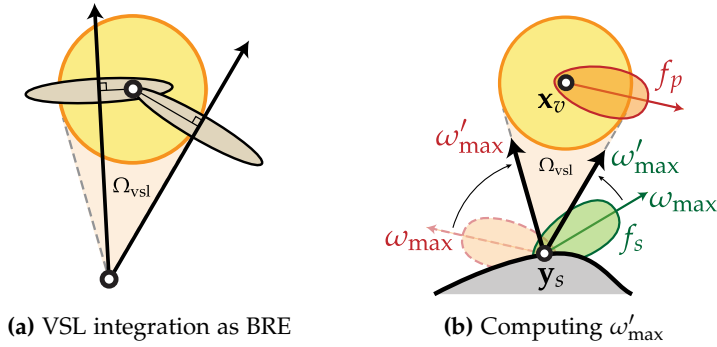
Figure 7.5: Evaluating the volumetric VSL using a beam radiance estimate with final gather (a) and estimating the max BRDF and phase direction (angle) ω'_{max} (b).

- evaluate the product of BRDF and phase function at ω'_{max}.

We repeat this procedure also for the phase function and use the bigger value of the product to estimate the upper bound on the integral

Finding ω_{max} is relatively simple for symmetric BRDFs (e.g. Phong, Ward); we discuss arbitrary BRDFs in Section 7.5. Note that when the VSL overlaps with the shading point, the resulting piecewise-linear PDF may still severely underestimate the value of the integrand. In order to reduce variance *in this case*, we apply a dilation filter to the PDF.

Computing light transport during rendering now becomes straightforward: we sample a point along the VRL according to our constructed PDF, inflate the point into a VSL, and numerically integrate Equation (7.2) by multiple importance sampling the solid angle Ω_{vsl}, the BSDF $f_s(-\omega \to \mathbf{y}_s \to -\omega_{\mathbf{y}})$, and the phase function $f_p(\omega_{\mathbf{x}} \to -\omega)$.

7.2.2 Media-to-Media Transport

For L_m^m (see Figure 7.3.a) we similarly inflate points along the VRL into VSLs, obtaining the following modification of Equation (6.3):

$$L_m^m(\mathbf{y} \leftarrow \omega_{\mathbf{y}}) = \int_0^s \int_0^t \kappa_s(\mathbf{x}_v)\kappa_s(\mathbf{y}_u)T(\mathbf{x} \leftrightarrow \mathbf{x}_v)T(\mathbf{x}_v \leftrightarrow \mathbf{y}_u)T(\mathbf{y}_u \leftrightarrow \mathbf{y})V(\mathbf{x}_v \leftrightarrow \mathbf{y}_u)L_m^{vsl}(\mathbf{y}_u \to \mathbf{y})\mathrm{d}v\mathrm{d}u \ (7.3)$$

where $L_m^{vsl}(\mathbf{y}_u \to \mathbf{y})$ is defined analogously to L_s^{vsl} but using two phase functions instead of a phase function and a BRDF.

For this transport type we extend our procedure in Section 7.2.1 by wrapping it into a two-stage method to account for the 2D integration domain. We first assume the camera ray to be an infinite line and importance sample the point $\mathbf{x}_v$ along the VRL according to the inverse squared distance (as in Section 6.3) where we place a VSL. Once we have a VSL, we proceed analogously to the media-to-surface case: we construct a piecewise-linear PDF in the angular domain along the spherical arc obtained by projecting the camera ray onto the VSL. As both locations (VSL and the point on the camera ray) are in the media, the PDF construction evaluates two phase functions instead of a phase function and a BRDF. We find ω'_{max} based on the phase function at $\mathbf{x}_v$; for analytic models, e.g. Henyey-Greenstein, this is again trivial.

7.2.3   Surface-to-Media Transport

L_m^s transport (Figure 7.3.c) is the dual of L_s^m: to sample a point on the camera ray, we construct the PDF in the angular domain about the surface point. The surface VPL (or endpoint of the VRL) is inflated into a VSL and its contribution to the camera ray location is evaluated numerically.

7.2.4   Surface-to-Surface Transport

L_s^s transport is computed with a modified application of the traditional VSL approach [Hašan et al. 2009]: we compute the transport progressively (see Section 7.3.1) with a shrinking VSL radius, resulting in converging results.

7.3   Algorithm

We implemented our algorithm in a hybrid CPU-GPU rendering system. We use CPU ray-tracing to trace light sub-paths (scattering in the media and at surfaces) and reflected/refracted camera rays. To connect the light and camera sub-paths (i.e. to evaluate the four different light transport types), we use a GPU-accelerated integration scheme, which employs GPU ray-tracing [Aila and Laine 2009] to resolve visibility queries.

Segments of the light sub-paths form our VBLs and their surface endpoints yield surface VSLs. These are used to estimate all indirect lighting (except for caustics, which are captured using progressive photon mapping (PPM) and progressive photon beams (PPB)[1]). Indirect illumination coming from surfaces is computed with VSLs [Hašan et al. 2009] using our progressive estimation. Indirect illumination coming from media is estimated by evaluating Equation (7.1) and Equation (7.3) for every VBL.

7.3.1   Progressive Rendering

Our complete rendering algorithm is contained within a progressive estimation framework. We render multiple independent passes, the running average of which is displayed as the rendered image.

Inflating VPLs into VSLs and VRLs into VBLs introduces bias by blurring out the illumination and scattering functions. Fortunately, each of these transport types can be viewed as an explicit final gather over either progressive photon beams [Jarosz et al. 2011b], or progressive photon mapping [Hachisuka et al. 2008b, Knaus and Zwicker 2011]. We can therefore rely on progressive radius reduction to ensure that both bias and variance converge to zero in the limit. In our context, the radius reduction for all four types of light transport corresponds to a 2D blur (surface VSLs are blurred into discs, and volumetric VSLs are integrated using the 2D blurring of the BRE). We use the improved reduction formula from Jarosz et al. [2011b] (which minimizes the impact of the number of photons per pass M on the final result) applied to the *squared* radius

---

[1]We also use PPM and PPB to evaluate direct illumination and single scattering.

due to the 2D blurring of VSLs and VBLs. The squared radius in the ith pass is:

$$r_i^2 = r_0^2 \left(\prod_{k=1}^{Mi-1} \frac{k+\alpha}{k} \right) \frac{1}{Mi},$$ (7.4)

where $\alpha \in (0,1)$ is a user parameter that specifies the aggressiveness of the radius reduction, and M the number of photon paths traced per pass. We currently set the initial radius r_0 of all VBLs and VSLs to a user-specified constant, though this could be modified to use an adaptive per-VBL/VSL radius for improved results.

7.4   Results

In this section, we compare VBLs + VSLs to VRLs + VPLs. The full global illumination solutions for our three scenes are shown in Figure 7.6, where we additionally include volume and surface caustics computed using PPB and PPM. All timings were measured on an Intel Core i7 with 12 CPUs @ 3.2GHz, 24GB RAM, and an NVIDIA Quadro 6000. The CPU ray-tracer and PPM/PPB are parallelized over all CPU cores. We use the Henyey-Greenstein phase function for all our results with anisotropic media. We use up to 100 rays to numerically integrate the solid angle of VSLs, as suggested by Hašan et al. [2009]. For all our results we use $\alpha = 0.7$.

Figure 7.7 shows the BUDDHA scene (720×720 resolution) which is filled with dense media with a strongly anisotropic ($g = 0.7$) HG phase function; the statue is highly glossy with a Phong exponent of 80. The majority of the illumination in the scene is due to indirect contributions. Unbounded VRLs + VPLs result in visible artifacts highlighting the presence of singularities. Bounded VRLs + VPLs lose energy and modify material appearance (note the regions around the Buddha and the orange wall). Our technique avoids both of these drawbacks.

Figure 7.8 details the progressive radius reduction using zoomed-insets from Figure 7.7. The images show a small region (L_m^s on the top, L_s^m on the bottom) of sharp illumination. Though the initial estimate is somewhat blurred using VBLs and VSLs, the progressive reduction converges to the reference solution (computed using many unbounded VRLs and VPLs).

Figure 7.9 shows a comparison for the SMOKYROOM scene, which is filled with an isotropic, heterogeneous medium (computed using Perlin noise) with density decreasing with height.

(a) BUDDHA scene   **(b)** SMOKYROOM scene   **(c)** CARS scene

Figure 7.6: Full global illumination for the BUDDHA, SMOKYROOM, and CARS scenes, including the sum of the four light transport paths from Figures 7.2 and 7.3 as well as caustics computed using progressive photon beams and progressive photon mapping.
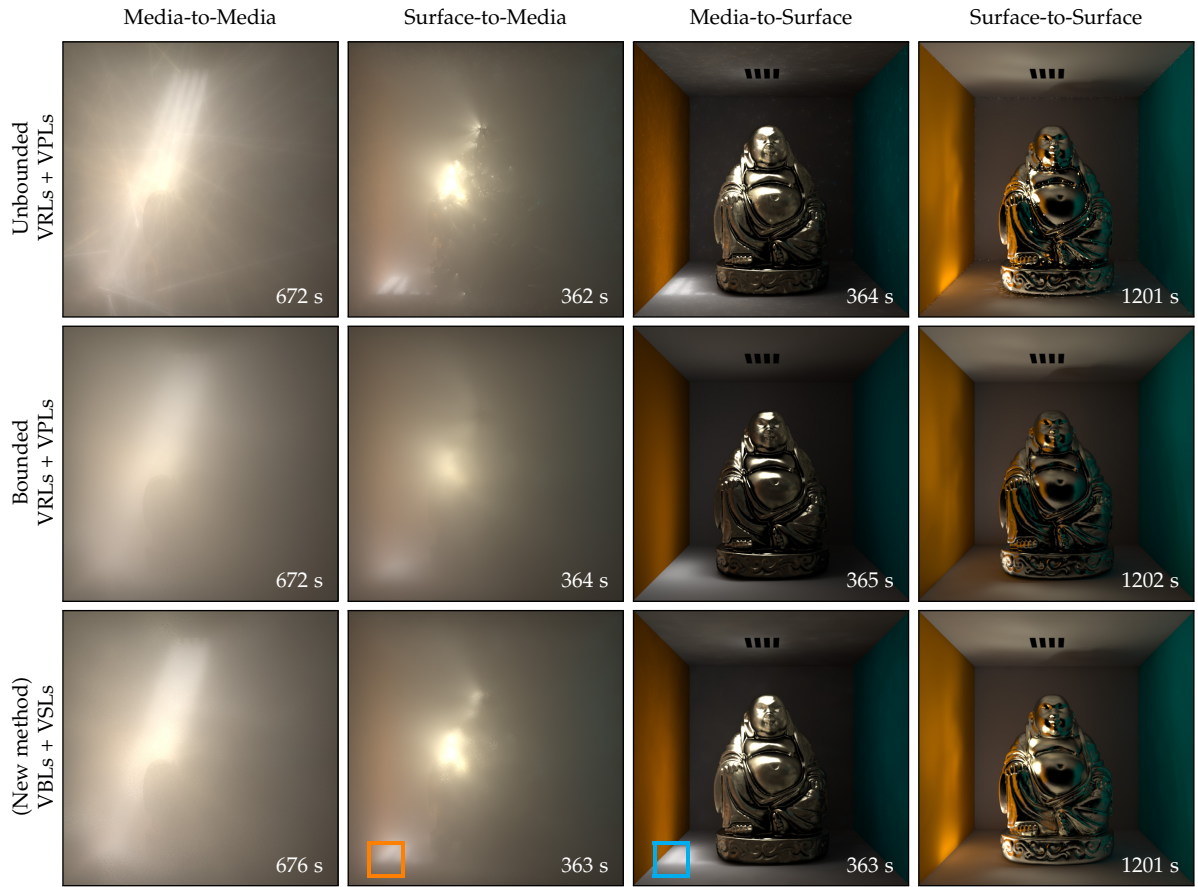
Figure 7.7: An equal-time (unconverged) comparison of unbounded/unbiased VRLs + VSLs (top), bounded VRLs + VSLs (middle), and our method (bottom) in the BUDDHA scene. The orange and cyan frames mark regions for which Figure 7.8 shows the progressive convergence over time.
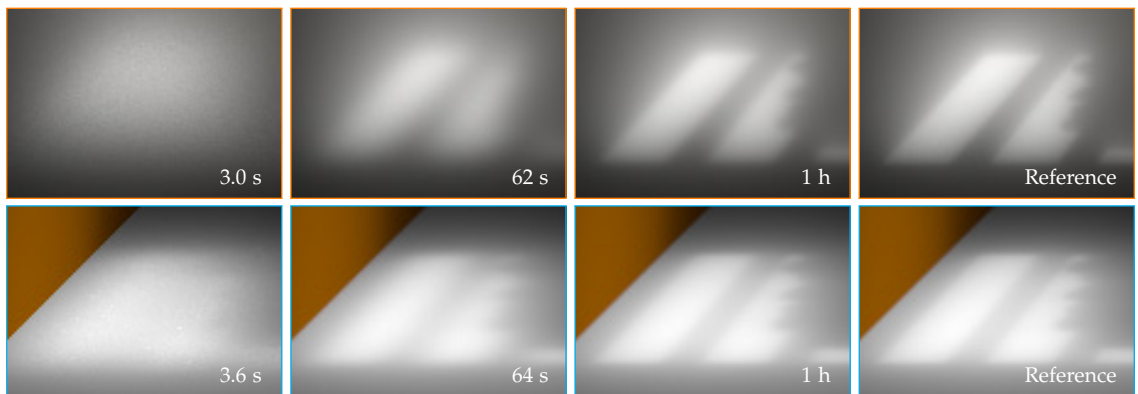
Figure 7.8: These images show small regions from the BUDDHA scene (Figure 7.7 shows full renderings) where we compare the convergence for the surface-to-media (top) and media-to-surface (bottom) transport. Our progressive VBLs + VSLs technique converges to the reference, here computed using a large number of VRLs + VPLs.
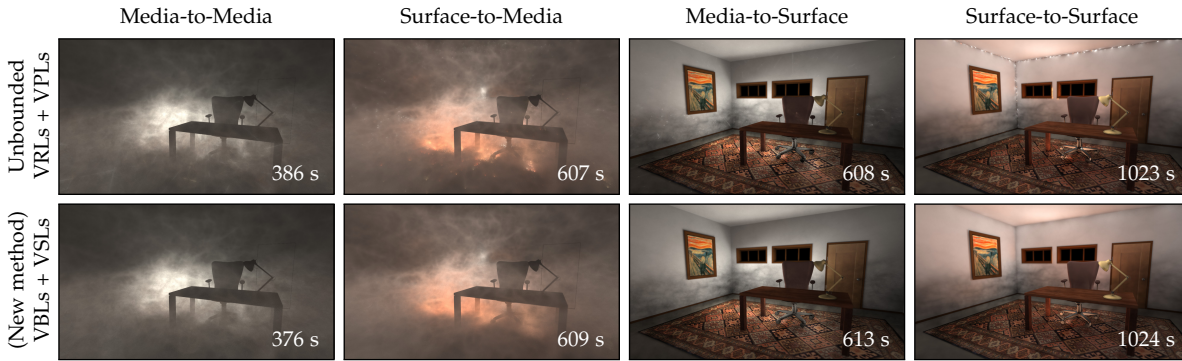
Figure 7.9: An equal-time (unconverged) comparison of unbounded/unbiased VRLs + VPLs (top) and our method (bottom) in the SMOKYROOM scene. Please zoom-in in the electronic version to better see the difference.

(a) VRLs + VPLs   **(b)** VBLs + VSLs
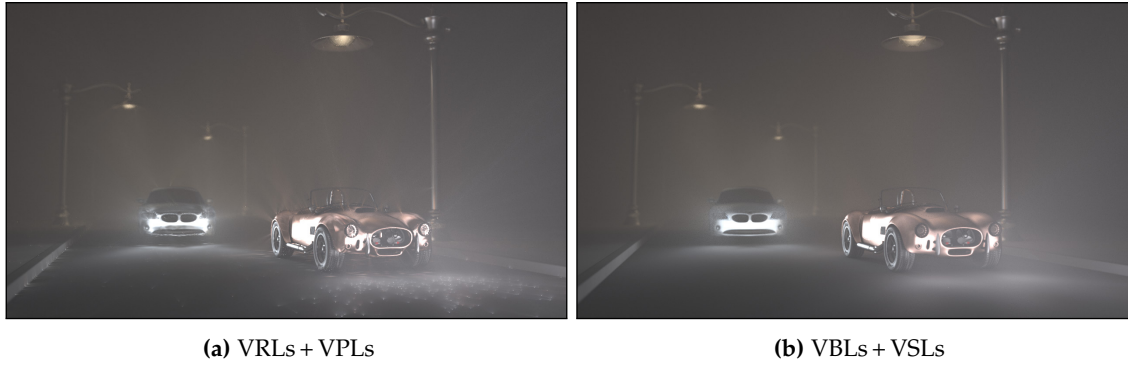
Figure 7.10: Surface and volumetric indirect illumination (i.e. $L_m^m + L_s^m + L_m^s + L_s^s$) in the CARS scene rendered with VRLs + VPLs (a) and VBLs + VSLs (b). Figure 7.11 shows few selected regions to better depict the differences.
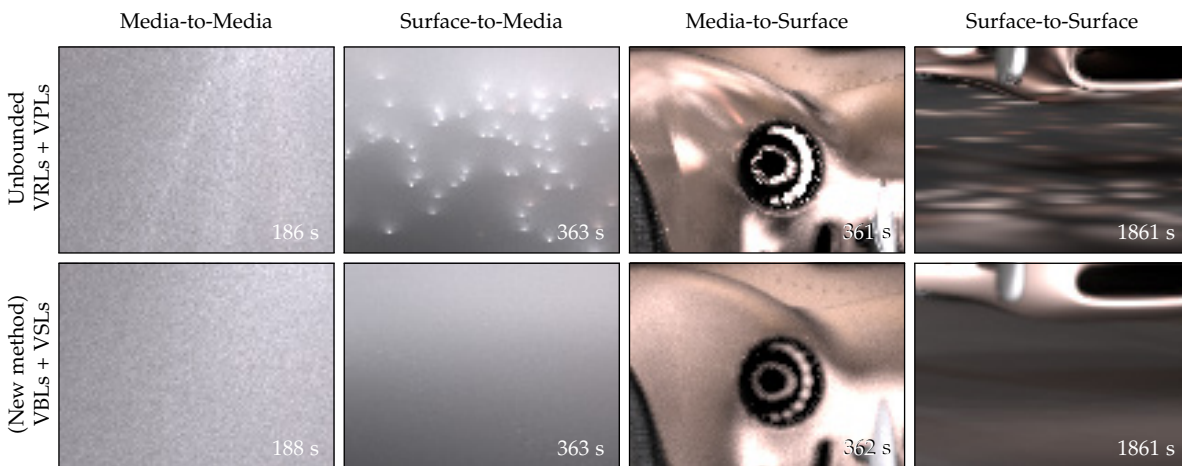
Figure 7.11: An equal-time (unconverged) comparison of unbounded/unbiased VRLs + VPLs (top) and VBLs + VSLs (bottom) in the CARS scene. These images are insets from renderings in Figure 7.10 and were inidividually tone-mapped for easy comparison. The timings in the bottom right corner of each inset report the rendering time of the entire 1280×720 image.

Figure 7.10 show the CARS scene, which consists of 2.2 million triangles (1.2 million vertices). Both cars are made of highly glossy materials (Phong BRDF with exponents for the car bodies of 100 and 60). The surrounding media is homogeneous with anisotropic scattering (HG coefficient $g = 0.25$). Note that light sources are encased in glass, i.e. all direct illumination in the scene is due to caustics. Figure 7.10 shows that virtually no high frequency artifacts (except for some noise) remain visible with our method after 46.2 minutes render time (1280×720 resolution). Figure 7.11 shows an equal-time comparison of the individual transport components that were computed using unbounded VRLs + VPLs and our VBL + VSL method; each set of transport types is tone mapped separately for better comparison.

7.5 Discussion and Possible Improvements

In general, our method eliminates distracting artifacts faster than VRLs + VPLs, especially when glossy surfaces and/or anisotropic phase functions are present (see also Figure 7.2). The singularities are traded for some blurring, which is less objectionable, especially in early passes, and results in fewer visual differences from reference than bounding. In the limit however, this blurring disappears completely due to the progressive radius reduction (which is not currently possible with bounding). We similarly improve standard VSLs, enabling convergent results with fixed memory footprint. Though the mathematical convergence rate is slower than with unbiased methods [Knaus and Zwicker 2011], VBLs deliver an acceptable image faster, which the user often cares about more than numerical convergence.

Arbitrary BRDFs. For BRDFs that do not exhibit perfect circular symmetry, our approach should still perform well as long as the BRDF has a single major peak from where its value falls-off. In essence, the maximum angle defines a cone of directions that is guaranteed to contain the maximum value. In the case of symmetric BRDFs, all values along this cone are equal to the maximum. For general BRDFs, a more sophisticated technique could try to solve for the unique maximum within this cone. We leave this as future work.

Perfect PDF. Our approximate PDF for the product of the two scattering functions may become slightly suboptimal in certain cases. We avoid expensive numeric construction by identifying directions where the BRDF (or the phase function at the VRL for L_m^m) reaches its maximum value. We then evaluate the product of both scattering functions assuming that all the transport occurs along that direction, using it as the value when constructing the PDF. In some cases this might not give the actual maximum of the product.

Investigating alternative approaches to efficiently construct more accurate PDFs for our estimators is an interesting area of future work. One option for circularly symmetric and monotonically decreasing functions (measured about deviations from the axis of symmetry, e.g. the HG phase function or Phong BRDF) is to also identify the maximum direction of contribution of the second scattering function. The product's maximum is guaranteed to lie on the great arc between these two directions and may be, in some cases, found analytically.

7.6   Conclusion

In this chapter, we presented an extension to VRLs; a new lighting primitive that eliminates the singularities present in existing many-light techniques for volumetric media. Our technique avoids the downsides of bounding. The benefits of VBLs over VRLs are akin to the benefits of VSLs over VPLs, although we efficiently handle a larger range of transport scenarios. We also devised novel importance sampling schemes to explicitly sample these complex transport paths in, from, and to surfaces and media. Our method converges faster than the state-of-the-art, without distracting singularities or energy loss, in scenes with complex lighting, glossy BRDFs, anisotropic phase functions, and heterogeneous media.

We extended progressive radiance estimation to handle VBLs and counteract the bias they introduce, ensuring that we converge to the correct result with a fixed memory footprint. As a result, we can generate fast previews and converged renderings with the same algorithm.

Rasterized Bounding Volume Hierarchies

To improve is to change;
to be perfect is to change often.

— Winston Churchill (1874–1965)

T he efficiency of Monte Carlo integration depends on two major criteria: how well we distribute our samples, and how quickly we can draw them. In order to render high-quality images fast, we need to perform well in both. While in previous chapters we strived to maximize the outcome of each sample without introducing artifacts, in this one, we shift our focus towards another important ingredient of rendering; the visibility testing. The common denominator of most high-quality rendering algorithms, including those in Chapters 4 to 7, is the need to resolve dozens of visibility queries. These queries are costly, if not the most expensive part of evaluating samples.

A popular technique for computing visibility, which has been intensively studied in past decades, is ray tracing. While the first challenge was to develop appropriate acceleration structures for static scenes, research next began to focus on efficiently building these data structures for fully dynamic scenes in every frame [Wald et al. 2007], and also on graphics hardware [Lauterbach et al. 2009, Pantaleoni and Luebke 2010, Zhou et al. 2008].

Rasterization is another technique for resolving visibility. For primary rays, both ray tracing and rasterization essentially produce identical results; and both can be accelerated using spatial data structures, either for minimizing the number of intersection tests or culling geometry that cannot be rasterized to the frame buffer (see [Cohen-Or et al. 2003] and [Dachsbacher 2010] for overviews). In the case of point-based many-light algorithms, we can exploit hardware rasterization to quickly create omni-directional shadow maps [Williams 1978] that can resolve VPL occlusion very efficiently. However, in the case of VRLs and VBLs the benefits of rasterization cannot be exploited that easily as the queries are not known *a-priori*; we thus cannot precompute and use a set of shadow maps.

The more general ray tracing seems to provide better means to resolve visibility queries in our case. The generality is indeed one of the characteristic advantages of ray tracing over the throughput oriented rasterization. In this chapter, we focus on accelerating ray tracing by creating an approximate accelerating structure called a *rasterized bounding volume hierarchy* (RBVH). Our desire is to combine the high throughput of rasterization with the generality of ray tracing: we use hardware rasterization to quickly create an accelerator, which can be later used for fast ray tracing of arbitrary rays.

(a) Height fields   **(b)** Node hierarchy and atlas
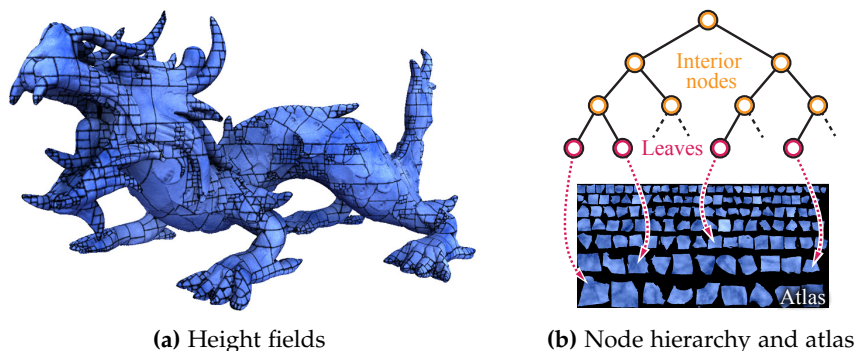
Figure 8.1: A rasterized bounding volume hierarchy (RBVH) represents surfaces using a set of height fields (a) that are organized into a hierarchy and stored in an atlas (b).

A common consensus is that the intersection computation for secondary rays is often not required to be accurate, e.g. when computing indirect lighting or glossy reflections [Yu et al. 2009]. In contrast to standard bounding volume hierarchies, we try to leverage this observation when constructing the RBVH: we identify subtrees containing surfaces which can be represented by, and thus rasterized to, a single height field. For these subtrees the conventional ray-surface intersection, possibly involving a large number of triangles, is replaced by a simple ray marching procedure to find the intersection with the surface.

In general, RBVHs are shallow structures with leaves storing geometry in the form of height fields (see Figure 8.1). These can be rasterized at an arbitrary resolution and thus, thanks to decoupling from the input surfaces, inherently provide means to adjust the level of detail (LOD).

RBVHs are best suited for complex scenes consisting of large number of primitives, e.g. obtained from subdivision surfaces or scanned environments. We show that in these cases an RBVH can achieve better *approximate* ray tracing performance than other accelerating structures. In addition, RBVHs have further beneficial properties:

- the height fields are stored in a texture atlas, which automatically provides a parameterization of surfaces and can be used to store on-surfaces signals, e.g. for interactive painting or photon mapping,

- RBVHs are not restricted to polygonal meshes: all representations that can be rasterized, e.g. point clouds [Gross and Pfister 2007], can directly be used during construction, and,

- the memory footprint of RBVHs is typically much lower than that of the original geometry as they can be created with just the required amount of detail. This enables ray casting of large scenes with limited memory, e.g. on GPUs.

In the next section, we review the most relevant previous work. In Sections 8.2 through 8.4, we describe how to construct and traverse RBVHs, and also introduce a *hybrid* RBVH, which stores the original geometry in leaves (like a traditional BVH) whenever height fields are not well-suited to represent the geometry. In Sections 8.5 and 8.6, we present a parallel construction scheme to build the accelerator on the GPU and provide necessary implementation details. Finally, in Sections 8.7 and 8.8, we present qualitative and quantitative evaluation of the RBVH and demonstrate the benefits of using the accelerator in applications such as rendering caustics with photon mapping, glossy reflections, interactive painting, and ray casting of point clouds.

8.1 Previous Work

Acceleration Structures for Ray Tracing.   Hierarchical and non-hierarchical spatial index structures, with all their advantages and disadvantages, have been explored for accelerating ray casting in various applications on different architectures; Wald et al. [2007] have an excellent overview.   Recent work focuses on building BVHs and kD-trees on the GPU, where they can be directly used for ray casting. Lauterbach et al. [2009] construct LBVHs by linearizing primitives along a space filling Morton curve combined with the surface area heuristic (SAH) yielding close to optimal hierarchies, and thus good overall performance for both construction and traversal. This approach has been improved using a hierarchical formulation [Pantaleoni and Luebke 2010] and further accelerated with work queues while using less memory [Garanzha et al. 2011]. Karras [2012] presents an in-place algorithm for constructing binary radix trees and later Karras and Aila [2013] propose a massively parallel construction with tracing performance close to sequential algorithms.

Ray tracing performance of BVHs can be improved by creating tighter axis-aligned bounding boxes (AABBs), e.g. by split clipping [Ernst and Greiner 2007], subdividing triangles recursively [Dammertz and Keller 2008], or adapting the SAH [Stich et al. 2009]. The potential of these approaches has been analyzed by Popov et al. [2009] who also present a generic algorithm. Aila and Laine [2009] analyze the traversal of BVHs on GPUs and their work can be considered the state of the art in terms of ray casting performance.

kD-trees can also be efficiently built on the GPU using a data-parallel spatial median algorithm for the upper levels of the tree to partition the workload between streaming processors [Zhou et al. 2008]. In contrast, Choi et al. [2010] focus on precise SAH-optimized kD-trees on architectures with less cores. Various two-level hierarchies were proposed for ray tracing dynamic scenes with nested grids [Kalojanov et al. 2011] and handling tessellated and displaced patches [Hanika et al. 2010]. The benefits of using shallow hierarchies were explored in [Dammertz et al. 2008], but only in the context of multicore CPUs. Note that RBVHs are also shallow, as entire subtrees are replaced by single height fields.

Ray Tracing with Sample-Based Representations.   This topic is intensively studied and closely related to our work. A classic sample-based representation is the voxelization of a scene, possibly stored as a hierarchy in an octree, e.g. [Crassin et al. 2009; 2011]. Voxel data structures allow for high ray casting performance and adaptive accuracy [Laine and Karras 2010], but often require significant construction time and memory.

Detailed displacements of smooth surfaces can be represented as height fields and ray casted very efficiently (see Szirmay-Kalos and Umenhoffer [2008] for an overview). Ray casting height fields has further been extended to handle arbitrary geometry using non-orthogonal projections [Baboud and Décoret 2006]. A set of depth cube maps, rendered from well-chosen locations within a scene, can also be used to accelerate ray casting for photon mapping [Yao et al. 2010].

Carr et al. [2006] use geometry images that enable efficient ray casting, since AABBs can be easily obtained from min/max-mipmaps. Note that this approach handles deforming geometry but the topology is not allowed to change, as it is too costly to recompute the parameterization on-the-fly. Other examples of approximate scene representations are point hierarchies used in micro rendering [Ritschel et al. 2009a] to synthesize GI. A common aspect of most of the previously mentioned approaches is that the input surfaces, typically triangle meshes, are resampled.

Closely related to our work, de Toledo et al. [2008] partition an object's surface and represent the individual parts as height fields. In contrast to our work, they do not construct a hierarchical data structure and the build process is too costly for frequent updates. Although used for storing textures, the partitioning scheme used in TileTrees [Lefebvre and Dachsbacher 2007], which splits the surfaces into parts that can be bijectively projected onto a cube's faces, is in spirit similar to ours, but not feasible for updating the data structure on-the-fly.

Level of Detail.   An important feature of RBVHs is the inherent possibility to adjust the LOD of the representation. Pantaleoni et al. [2010] report that voxelization is well-suited as an approximate representation for small and medium sized scenes, but fails to handle large, complex scenes. They propose to combine acceleration structures with a multiresolution scheme for LOD. In principle, any mesh decimation method, e.g. progressive meshes [Hoppe 1997], could be used; however, this requires maintaining and implementing two intricate algorithms for LOD and construction.

Related to our work are the volume surface trees [Boubekeur et al. 2006] that combine an octree and a set of quadtrees to represent surfaces. However, their goal is to resample surfaces for reconstruction and mesh simplification; the resulting structure is not suitable for ray casting and adaptive level of detail.

8.2   RBVH Construction

In contrast to a triangle-based BVH, our RBVH can be seen as a two-level data structure, where the upper part consists of a shallow tree, and the lower part represents the geometry using height fields. We build the RBVH in a top-down manner, i.e. we start from the scene's bounding box representing the root node and continue with the inner nodes towards the leaves. In general, each node is split and the primitives (e.g. triangles or points) partitioned into child nodes until: (1) the geometry can be faithfully represented as a single height field, and (2) the cost of intersecting the height field is smaller than the traversal of an interior node (resembling the idea of the SAH). Once the geometry is partitioned, we rasterize the primitives of every individual leaf using an orthogonal projection into a texture atlas, where each tile stores depth values of a height field that represents the surfaces in the corresponding leaf.

The RBVH is an approximate, sample-based structure providing several means to control the quality of the representation: in Section 8.2.1 we describe two *local* criteria to measure and control the accuracy of representing a surface by a height field. Then we introduce heuristics for node splitting and minimizing traversal costs in Section 8.2.2, and finally, in Section 8.2.3, we detail the *global* quality control (i.e. the sampling density) achieved through varying the resolution of the rasterized height fields.

8.2.1   Refinement Criteria

One integral component of the RBVH construction is an efficient way to determine whether we can, and should, represent a part of a surface as a single height field. Such representation is only possible without loss of information, if we find a projection of the surface onto a plane without folding. Additionally, we strive to sample surfaces as uniformly as possible and thus we

(a) Cone of normals   **(b)** Projected surface area   **(c)** Projection frustum
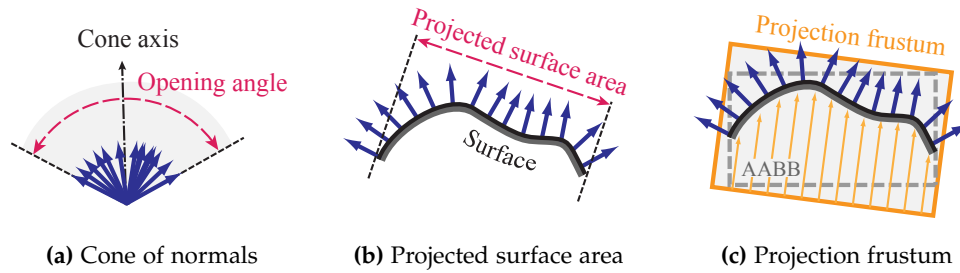
Figure 8.2: During the RBVH construction, nodes are further refined if the cone of normals (a) or the projected area (b) (depending on which one is used) exceeds a user-defined quality threshold. In the opposite case, the surfaces within the node are rasterized using an orthogonal projection (c).

should avoid rasterizing surfaces from grazing angles. In order to find a suitable direction and to minimize the projection error, we consider one of the following measures: the minimum cone subtended by the normals of the surface, and the area of the projected surface. Both measures are shown in Figure 8.2 and detailed in the following paragraphs.

Cone of Normals. The cone of normals of the primitives within a node can be used to determine if there exists an orthogonal projection where all surfaces are front-facing: if the opening angle of the cone is less than π, such directions exist, e.g. the cone axis, and all primitives are front facing and can be represented with a single height field. Otherwise, we should split the surface and represent it with multiple height fields. In practice, we compute an approximation [Shirman and Abi-Ezzi 1993] and use even narrower cones (e.g. $\pi/2$) enforcing more uniform sampling of surfaces. To maximize the *minimum* sampling density (of the most diverted surface) we orient the projection frustum along the cone axis. As the direction defining the orientation of the projection frustum we use the axis of the cone.

Projected Surface Area. Another good projection direction is the average surface orientation, computed as the area-weighted sum over all primitives' normals. It maximizes the *average* sampling density, but does not guarantee sampling of the entire surface, as some primitives might be back-facing and thus occluded. As a quality metric we use the projected area $A^\perp$ of the primitives, which equals to the length of the summed area-weighted normals. We rasterize the surfaces if the ratio $A^\perp/A$, where A is the surface area, is greater than a user-defined threshold α; otherwise we split the node.

Discussion. The cone of normals is a restrictive criterion, splitting a node whenever there is no projection possible without back-facing primitives. According to our experiments, it is best-suited for (manually) modeled scenes, e.g. from subdivision surfaces. The relative projected area is robust against noise in the primitives' orientation, which is often present in scanned geometry. In either case the projection frustum is defined by the the minimum bounding box that is oriented along the projection direction and contains the AABB of the surfaces. Similar to traditional acceleration structures, we also consider whether it is beneficial to split the surface, even if it can already be represented as a height field. Heuristics for splitting and an analysis of their parameters are discussed in Sections 8.2.2 and 8.2.4.

| **(a)** Without object split | **(b)** With object split | **(c)** Uniform sampling |
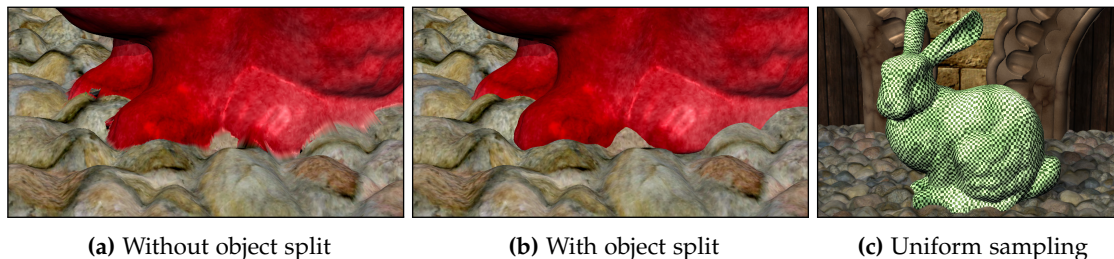
Figure 8.3: Subdividing and rasterizing surfaces can fuse surfaces of different objects (a). This can be avoided by splitting the node w.r.t object IDs. The RBVH achieves almost uniform sampling across all surfaces (shown as a checker board on the bunny model) (c).

8.2.2 Subdivision Strategies

The previously described refinement criteria determine *whether* a surface has to be split. In this section, we discuss heuristics determining *how* to actually split the surface, i.e. how to partition primitives within a node. We discuss and compare two approaches partitioning the primitives with respect to a split plane: the RBVH pendant of the surface area heuristic (SAH) [Havran 2000] and the simple spatial median. We also introduce a complementary object split strategy that avoids fusing surfaces of different objects by splitting according to object ID.

Surface Area Heuristic. Ray casting accelerators are often built according to the SAH, which defines the cost of a partition by summing up the costs of intersecting each child, weighted by the respective probability that a ray passes through them. The minimum is usually greedily searched by evaluating the cost of several candidate split-planes at the scope of the current node. In case of the RBVH, the cost of intersecting a surface amounts to ray marching the respective height field. The number of ray marching steps depends on the resolution of the height field, which is in turn linked to the surface area, as we strive to achieve uniform sampling (described in Section 8.2.3). Therefore, the area of the children's surfaces is already a good estimate for the intersection cost. Note that we use the SAH only to determine where to split, not whether to split. The modified SAH for finding the best partitioning of a node a into two children b and c is then:

$$C(b,c) = p(b|a)A(b) + p(c|a)A(c),\tag{8.1}$$

where $A(b)$ is the area of the surface in b, and $p(b|a)$ is the geometric probability of intersecting b: $p(b|a) = S(b)/S(a)$, where $S(.)$ is the surface area of a node's bounding box.

Spatial Median. The second strategy places the split plane always in the middle of the bounding box perpendicular to the longest axis. This can be highly suboptimal in the case of regular BVHs, as the number of primitives on both sides can differ significantly. However, since RB-VHs decouple from the actual tessellation, the spatial median somewhat resembles the idea of Equation (8.1), and, for smooth surfaces in particular, also tends to split the surface into roughly equal areas (see Section 8.2.4 for analysis).

Object Split. When the surfaces of two or more objects intersect, it is reasonable to split them according to their object IDs (which is typically available from the scene modeling or hierarchy). By this, we can avoid excessive refinement due to a low projected area or diverging normals of nearby objects. Figure 8.3 shows an example where the object splitting also avoids fusing of two meshes. We perform the object split whenever a node contains surfaces of exactly two objects.

8.2.3 Rasterization to the Atlas

After splitting the nodes and creating the tree hierarchy of the RBVH, we rasterize the surfaces to the atlas. For every leaf node, we compute an orthogonal bounding frustum that is aligned with the projection direction and contains the axis-aligned bounding box (AABB) of the corresponding surface (see Figure 8.2.c). Next we determine the resolution allocated to the respective atlas tile, i.e. how densely we sample the surface. Our goal is to retain a (roughly) uniform sampling of surfaces, which can be intuitively controlled by the user. For this, we use a *global* pixel-to-area ratio ρ to compute the resolution R of a square tile as: $R = \sqrt{\rho A^2 / A^\perp}$, where ρA is the total number of pixels for representing the surface scaled by the inverse relative projected area $A / A^\perp$. Although the actual number of samples used to store the height information is usually smaller (as a tile is typically not fully covered by the projection), we found that this approach still provides almost uniform sampling of the entire surface (see Figure 8.3.c). Complementing the local refinement criteria, the pixel-to-area ratio is a means to adjust the quality globally.

As we create square tiles, we can easily and tightly pack them into the atlas. For that we sort them according to their descending resolution R and pack them row-wise. Within each row the tiles are placed from the largest to the smallest from left to right. Resolution of the first (largest) tile in a row determines the vertical offset to the next row. For each leaf, we store a final projection matrix that is computed from the bounding frustum, tile resolution, and position in the atlas. Lastly, we transform all primitives using the corresponding matrices and rasterize them to the atlas. Note that we can use the atlas to store arbitrary on-surface signals, e.g. normals or surface colors, by rasterizing them to additional atlas layers.

As we rasterize surfaces from different directions and at different resolutions, the height fields typically do not match exactly at their boundaries. To avoid cracks in the reconstruction, we ensure that the height fields of neighboring surface parts slightly overlap by duplicating primitives within a certain region around the split plane ($\pm 5\%$ in our scenes), and assigning them to both children. After rasterization we also apply a dilation filter on the atlas that creates an additional "safety-border" for all tiles. Note that this process is common to many atlas-based techniques (e.g. see [de Toledo et al. 2008]). In order to create meaningful data, we compute the gradient of the surrounding pixels to ensure that the dilated surfaces do not create distracting extrusions.

8.2.4 Analysis of Subdivision Strategies

In addition to the subdivision strategies, the RBVH has another degree of freedom, which trades between the size of the node hierarchy and the resolution of tiles: we may still want to split a surface, which could already be represented using a single height field, if it improves the traversal performance. Note that this consideration is similar to the choice of how many primitives should be stored in the leaves of regular BVHs.
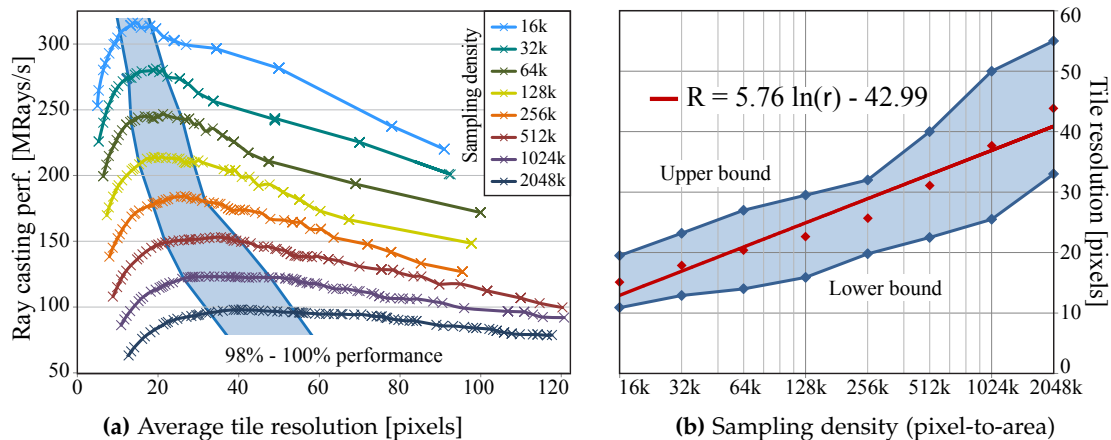
Figure 8.4: In (a) the horizontal axis shows the average tile resolution and individual series depict the dependency of the ray casting performance on the sampling density. In (b) we identify intervals of the resolution with at least 98% of the peak performance and approximate the midpoints of these intervals with a logarithmic function, that is used to select the optimal tile resolution during the RBVH construction.

To this end, we ran a series of benchmarks (primary and secondary rays separately) to find a good balance between the depth of the node hierarchy and the tile resolution. In each test we subdivided the nodes until the atlas tiles had resolutions lower than a specified threshold. Note that higher tile resolution results in shallower hierarchies and vice versa. Figure 8.4.a illustrates that the peak performance is obtained with different tile resolutions for different pixel-to-area ratios (curves in Figure 8.4.a), i.e. depending on the sampling density, there is an optimal resolution and we should refine the nodes until their tiles reach it. To derive the *optimal* tile resolution we fit a logarithmic function to the midpoints of parameter intervals with at least 98% of the peak performance. This has proven to be more reliable than fitting to absolute maxima. For coherent rays the optimal tile resolution is $5.76 \ln(\rho) - 42.99$; running the benchmark for secondary rays yielded $1.23 \ln(\rho) - 0.17$. That is, depending on the application, different construction parameters yield optimal RBVHs.

We also compare the impact of the SAH and spatial median heuristics in terms of ray tracing performance and the number of RBVH nodes in Table 8.1. Although the total number of nodes created using the spatial median was sometimes up to $2.5\times$ higher, the ray tracing was always at least 78%, and 86% on average, of the performance of the RBVH built with SAH. Considering the build times, it is obvious that the spatial median is the better choice for dynamic scenes.

8.2.5   Adaptive, Varying Level of Detail

In Sections 8.2.1 and 8.2.3 we described two means to control the accuracy and memory requirement of the RBVH. Altering these parameters also influences the performance:

1. loosening the refinement criterion, e.g. allowing larger cones of normals, effectively prunes the tree by representing surfaces, that would be otherwise refined, with a single height field. This sacrifices uniform sampling (eventually even bijective projections) for reducing the number of nodes in the tree;

Scene	# of faces	Spatial Median			Surface Area Heuristic			Relative
		# of nodes	CPU build	Tracing	# of nodes	CPU build	Tracing	perf.
Armadillo	332k	2908	0.5 s	129.1	2553	12.9 s	140.7	**92%**
Hand	655k	4829	0.9 s	182.3	2893	24.9 s	204.4	**89%**
Dragon	699k	2999	0.9 s	146.3	2674	26.8 s	161.3	**91%**
Happy Buddha	1.37M	8065	1.8 s	138.1	3266	50.6 s	169.3	**82%**
Children	1.45M	4369	1.7 s	147.7	2936	50.9 s	172.3	**86%**
Beast	2.82M	7639	3.5 s	109.6	4696	118.1 s	140.5	**78%**
Asian Dragon	7.22M	6345	5.9 s	159.9	5227	173.0 s	180.2	**89%**

Table 8.1: Number of nodes, CPU construction time (using 1 core), and tracing performance (in MRays/s) of our RBVH built using either the spatial median along the longest axis, or using the SAH selecting the best from 32 split candidates along each axis of the bounding box. The last column shows relative tracing performance of the tree built with spatial media to the one constructed with SAH.

2. decreasing the sampling density when rasterizing the tiles reduces memory footprint and speeds up the ray-height field intersection.

Instead of setting these parameters once for the entire RBVH, we can determine them for every node during construction. This enables us to adapt the level of detail according to a quality function that, for a given point in space, defines the desired quality (which is mapped to construction parameters). By this, for instance, we can locally adjust the accuracy of the RBVH depending on the distance to the viewer or any other point of interest (Figure 8.5).

In situations when the reconstruction of the entire RBVH is not desired, we adapt the level of detail by mip-mapping the atlas, and select an appropriate mip level during traversal. Since the tree structure does not change, this strategy is suboptimal to a full rebuild, but faster and well-working as long as the atlas tiles do not fuse during the mip-map creation.

Figure 8.5: RBVHs support level of detail rendering. Here the refinement criterion is loosened and the tile resolution decreases with the distance to the point of interest. The top inset depicts the boundaries of tiles, the bottom inset shows the varying sampling density.

(a) Visualization of representations   **(b)** Raytraced image
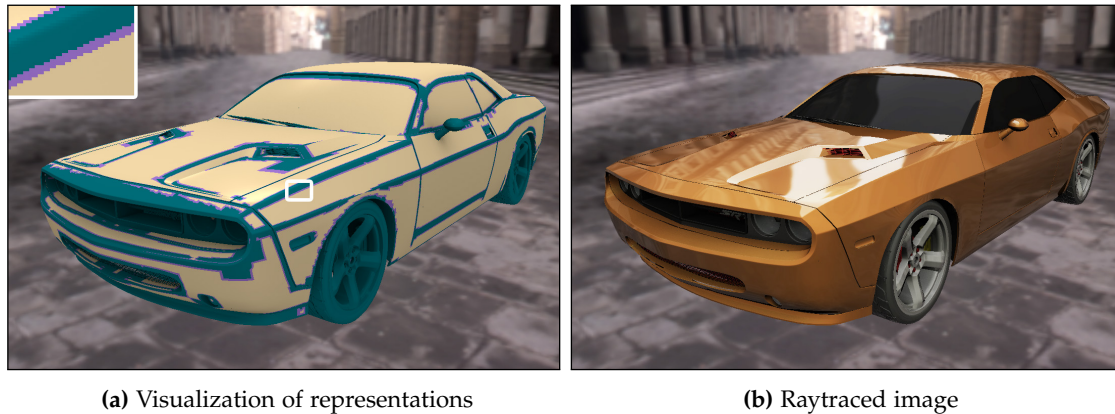
Figure 8.6: Hybrid RBVH; in (a) we visualize surfaces represented by height fields (yellow), triangles (green), or both (purple). The resulting rendering, computed by tracing primary and secondary rays, is free of any visible artifacts (b).

8.3   Hybrid RBVH

RBVHs work best when the input surfaces are highly tessellated and can be faithfully represented by height fields. Obviously this is not always the case and would restrict the applicability of the RBVH in many cases. To this end, we propose a hybrid RBVH: whenever the surfaces of a node cannot be efficiently represented as a height field (e.g., few triangles with large area, or surfaces with sharp bends) we build the subtree as a traditional BVH. This creates hybrid RBVHs where surfaces are partly represented as height fields and partly by triangles.

Figure 8.6 shows a car model with surfaces color-coded according to whether they are represented as height fields, triangles, or both (similar to two height fields, triangles and height fields of adjacent nodes may overlap). For the hybrid RBVH we precompute the surface curvature for the input mesh vertices and resort to triangles when (1) the percentage of vertices above a certain curvature exceeds 80%; (2) there are less than 8 triangles in the node.

8.4   Traversal of the RBVH

Using RBVHs for ray casting is similar to the traversal of traditional BVHs: the traversal starts at the root node, tests the ray against the childrens' bounding boxes, and stores the nodes to be visited on a stack. In the case of a leaf node, the ray segment overlapping the leaf's bounding box is transformed into atlas space using the stored projection matrix. Then we march along the corresponding line segment and test for an intersection with the height field using linear plus secant search (as in [Szirmay-Kalos and Umenhoffer 2008]). If an intersection is found, we transform the location back to world space. Note that the texture coordinates of the intersection can be used to look up surface attributes from other layers of the atlas. To reconstruct these attributes without seams, filtering across tiles can be implemented [Lefebvre and Dachsbacher 2007], but in our examples, increasing the resolution of the attribute layer provided sufficient quality.

Retrieving stored on-surface information for a given point in world space (on the surface) is

also easy: we first search for the leaf node whose bounding box contains this point, and then transform the point's coordinates into atlas space using the projection matrix. Note that due to the overlaps a surface point may be represented by two or more height fields. We account for this when writing data into the atlas and "splat" the value to all the texels corresponding to a single point.

8.5 GPU Construction

In this section, we describe an RBVH construction algorithm for massively parallel architectures, such as GPUs. We favor simplicity and fast construction using the inexpensive spatial median for partitioning the primitives. The slightly lower traversal performance is compensated by the faster construction, which is beneficial in dynamic scenarios. As the refinement criterion we use the more robust projected surface area, which also requires only one pass over the primitives (cone of normals needs two).

The construction proceeds in a breadth-first manner starting from the root node (containing all primitives) and creating nodes level by level. To construct a single level, we first compute the axis-aligned bounding boxes (AABB) and the average surface orientation for each node in the level. Next we determine which nodes can be rasterized and remove the corresponding primitives from the subsequent construction steps. All other primitives are assigned to the respective child nodes. We show a pseudocode and the meaning of the variables in Figure 8.7 and detail the construction in the next paragraphs.

We start the construction of the node hierarchy by initializing two arrays storing references to the primitives, *primRefs*, and the node to which every primitive belongs to, *nodeRefs*. Since we want the child nodes to slightly overlap (to prevent cracks), both arrays have to be large enough to allow duplicating references (in all our examples 150% of the original size was sufficient). We also keep track of the number of nodes N, the number of remaining references R, and the references stored in the final arrays of leaves F.

The construction of the RBVH continues until all remaining references R are processed, i.e. placed in the leaves (while $R > 0$, line 2). We assume that, at the beginning of each iteration, the primitive references are sorted according to the node they belong to (stored in *nodeRefs*), thus forming segments with the same node reference. In order to evaluate the refinement criterion, we compute the AABB, total area, and the sum of area-weighted normals for every node. For this, we fill three auxiliary arrays (*nodeBounds*, *nodeANorm*, and *nodeA*) with the primitive data (lines 3-7), and perform a parallel segmented reduction to obtain a single value per node (lines 8-10).

Next, using this per-node information we determine if primitives will be rasterized or further split, thus going into new child nodes: we evaluate the splitting criterion (Section 8.2.1) and optimal subdivision (Section 8.2.4) (line 14) and mark every primitive either for rasterization (line 15) or splitting (lines 17-18). In the latter case, we store two flags per primitive: *count* contains the number of children the primitive will go to (1 or 2), and *child* refers to the actual children (1 - left child, 2 - right child, 3 - both).

These flags are used to send the primitives to the final array (when they are flagged for rasterization) or to the construction array of the next RBVH level. To determine the locations within these array, we first compute parallel prefix sums (lines 19-20). Note that computing the prefix sum on the *count*-array automatically reserves space for duplicating primitives. Then we again

BuildNodeHierarchy(primBounds, primNorm, primA)

    nodeBounds[], nodeA[] *// AABB and surface area of nodes*
    nodeANorm *// sum of area weighted normals in each node*
    primRefs ← {1, 2, ...N} *// references to primitives*
    nodeRefs ← {1, 1, ...1} *// references to nodes*
    finPrimRefs *// final array with references to primitives*
    finNodeRefs *// final array with references to nodes*
    N *// (maximum) number of nodes potentially created so far*

1 $N \leftarrow 1$, $R \leftarrow$ number of primitives, $F \leftarrow 0$
2 **while** $R > 0$:
3   **for all** i in $[0, R)$ **in parallel** :
4        $j \leftarrow$ primRefs[i]
5   bounds[i] ← primBounds[j]
6   ANorm[i] ← primA[j] ∗ primNorm[j]
7   A[i] ← primA[j]

    // compute per-node information
8   nodeBounds[$N..2N$] ← **reduceByKey**(bounds, nodeRefs)
9   nodeANorm[$N..2N$] ← **reduceByKey**(ANorm, nodeRefs)
10   nodeA[$N..2N$] ← **reduceByKey**(A, nodeRefs)

    // decide whether to split or rasterize
11   rasterize[$0..R$] ← count[$0..R$] ← {0, 0, . . . 0}
12   **for all** i in $[0, R)$ **in parallel** :
13       $n \leftarrow$ nodeRefs[i]
14   **if** **not** *requiresSplit(n)* **and not** *shouldBeSplit(n)* :
15   rasterize[i] ← 1
16   **else** :
17   count[i] ← *toNChildren(i)*
18   child[i] ← *toWhichChildren(i)*

19   finRank[$0..R$] ← **scan**(rasterize)
20   cRank[$0..R$] ← **scan**(count)

    // filter primitives for rasterization; compact the others
21   **for all** i in $[0, R)$ **in parallel** :
22   **if** rasterize[i] :
23   finPrimRefs[$F +$ finRank[i]] ← primRefs[i]
24   finNodeRefs[$F +$ finRank[i]] ← nodeRefs[i]
25   **else** :
26   primRefs[cRank[i]] ← primRefs[i]
27   nodeRefs[cRank[i]] ← nodeRefs[i] ∗ 2 + child[i]&1
28   **if** child[i] = 3 :
29   primRefs[cRank[i] − 1] ← primRefs[i]
30   nodeRefs[cRank[i] − 1] ← nodeRefs[i] ∗ 2

    // sort primitives according to the node; update counters
31   **sortByKey**(primRefs, nodeRefs)
32   *update(R, F)*
33   $N \leftarrow 2N + 1$

Figure 8.7: Parallel algorithm for constructing the upper part (hierarchy of nodes) of the RBVH.

process all primitives: those flagged for rasterization will be simply appended to the final arrays according to the prefix sum (lines 22-24); primitives that go into child nodes are kept in the construction arrays, but are compacted to remove unused entries (we double-buffer the arrays to avoid write-after-read hazards). Primitives that are sent to both child nodes (recall that we accounted for that in the prefix sum) are duplicated in lines 29-30. Note that we use implicit addressing to avoid computing and storing pointers during the construction. In contrast to regular BVHs, the upper part of the RBVH is very shallow (hundreds or thousands of nodes), and thus the memory required due to storing a full tree for implicit addressing is small.

The last step of constructing a single RBVH level is to sort the references again according to the node indices (for the next iteration), and to update the number of remaining primitive references and references in leaves (lines 31-33).

Finalizing the RBVH. After the hierarchy construction, per-node attributes (e.g. the bounding box) are stored in the *node**** arrays. We first separate interior and leaf nodes into two arrays, and remove the unused entries that were introduced due to the implicit addressing. Lastly, we sort all leaf nodes according to their tile resolution, compute transformation matrices and rasterize all primitives referenced by *finPrimRefs* as described in Section 8.2.3.

8.6   Implementation Details

We implemented the RBVH construction on the CPU (for evaluating the subdivision strategies) and on the GPU. On the GPU, we use CUDA for the hierarchy construction and ray casting/traversal, and Direct3D 10 for rendering the primitives into the atlas. For all data parallel primitives (**reduceByKey**, **sortByKey**, **scan**) we used the Thrust CUDA library [Hoberock and Bell 2010].

After the hierarchy construction, we compact the nodes into a single tightly packed array, obtaining an efficient representation for fast traversal. Each interior node occupies $48 + 8$ bytes for both child-AABBs (6 floats each) and references to the child nodes (2×4 bytes). For leaves we only need to store the projection matrix. As we use orthogonal projections only, the last row is always $\{0, 0, 0, 1\}$, thus we can store the matrix as 12 floats, or 48 bytes, respectively.

8.7   Results and Discussion

In this section, we evaluate the RBVH regarding its ray casting performance, level of accuracy and approximate nature, construction time, and memory requirements. All results were measured on an Intel Core i7 system running at 2.8GHz with an NVIDIA GTX 470 GPU.

Quality vs. Performance. One key feature of the RBVH is that it allows trading quality for performance. Figures 8.8 and 8.9 show images where RBVHs with different quality configurations were used to trace primary rays, or secondary (diffuse) rays, respectively. When using the RBVH for primary rays, the approximate nature becomes visible for low quality settings such as Q4, but not for high quality settings. For benchmarking the secondary rays, the objects are rendered

Figure 8.8: Quality and performance of individual RBVH configurations for tracing primary rays. All scenes, except for the Hand, have been constructed using the less restrictive projected surface area. The construction parameters are shown in parenthesis next to the quality level. The left most column contains references ray traced using the BVH from [Aila and Laine 2009]. Numbers in the corners report ray tracing performance in million rays per second.

Figure 8.9: Quality and performance of individual RBVH configurations for tracing secondary (diffuse) rays. All scenes, except for the Hand, have been constructed using the less restrictive projected surface area. The construction parameters are shown in parenthesis next to the quality level. The left most column contains references ray traced using the BVH from [Aila and Laine 2009]. Numbers in the corners report ray tracing performance in million rays per second.

Scene	# of triangles	Primary (Coherent) Rays [MRays/s]						
		BVH	Q0	Q1	Q2	Q3	Q4	Speedup
Hand	655 k	178.7	145.3	176.6	209.9	266.0	300.0	**0.8 - 1.7**
Dragon	699 k	137.2	111.6	139.8	169.4	229.4	287.2	**0.8 - 2.1**
Happy Buddha	1.37 M	144.6	147.5	183.9	226.2	307.2	386.4	**1.0 - 2.7**
Children	1.45 M	123.2	121.2	148.8	177.6	232.2	282.8	**1.0 - 2.3**
Beast	2.82 M	120.4	131.9	163.1	195.4	256.0	298.3	**1.1 - 2.5**
Asian Dragon	7.22 M	85.9	158.5	190.2	216.1	263.7	304.4	**1.8 - 3.5**
Thai Statue	10.02 M	122.9	197.4	250.5	302.7	375.4	454.6	**1.6 - 3.7**

Table 8.2: Detailed performance of tracing primary rays (in million rays per second) for the scenes shown in Figure 8.8 with different quality settings (Q0 - finest, Q4 - coarsest; see Table 8.4 for the construction parameters). The "BVH" column reports the performance using Aila and Laine's [2009] method, "Speedup" shows the relative performance range of the RBVHs to the BVH.

Scene	# of triangles	Primary (Coherent) Rays [MRays/s]						
		BVH	Q0	Q1	Q2	Q3	Q4	Speedup
Hand	655 k	55.9	32.5	37.1	44.3	56.6	69.1	**0.6 - 1.2**
Dragon	699 k	38.2	20.7	21.7	27.6	38.7	53.3	**0.5 - 1.4**
Happy Buddha	1.37 M	37.2	21.7	24.9	30.5	41.4	54.6	**0.6 - 1.5**
Children	1.45 M	36.1	21.2	23.6	28.1	39.1	51.0	**0.6 - 1.4**
Beast	2.82 M	37.8	20.7	23.3	27.8	37.9	47.4	**0.5 - 1.3**
Asian Dragon	7.22 M	36.7	29.4	31.9	35.1	43.7	53.9	**0.8 - 1.5**
Thai Statue	10.02 M	32.9	30.7	35.5	41.9	51.7	59.9	**0.9 - 1.8**

Table 8.3: Detailed performance of tracing secondary rays (in million rays per second) for the scenes shown in Figure 8.9 with different quality settings (Q0 - finest, Q4 - coarsest; see Table 8.4 for the construction parameters). The "BVH" column reports the performance using Aila and Laine's [2009] method, "Speedup" shows the relative performance range of the RBVHs to the BVH.

Quality settings:	Q0	Q1	Q2	Q3	Q4
pixel-to-area ratio ρ_s	2M	1M	512k	128k	32
ratio of projected to world area α	0.85	0.80	0.75	0.70	0.65
cone of normals aperture φ [deg]	45	50	55	60	65

Table 8.4: Parameters of different quality configurations used to render images in Figures 8.8 and 8.9 and measure performance in Tables 8.2 and 8.3. All scenes were normalized to a total area of 1 to achieve equal sampling density. Q0, the finest configuration, subdivides aggressively to obtain high-quality RBVHs; Q4, the coarsest configuration, results in highly approximate but compact and high performance RBVHs.

using rasterization and the RBVHs are used to compute environmental lighting by randomly sampling the hemisphere with 4000 rays per pixel. Note that even for the coarse Q4 settings we obtain satisfying results.

Tables 8.2 and 8.3 present the corresponding tracing performance. To compare our accelerator against a triangle-based BVH, we integrated Aila and Laine's [2009] GPU ray tracer (which can currently be considered as the state of the art) into our application to ensure that we cast exactly the same rays with both acceleration structures. Note that the RBVH linearly trades speed for approximation, which is important for finding the right configuration under given constraints. Table 8.4 lists the local and global construction parameters for the five different quality levels.

	BVH			RBVH Q0			Q2	Q4
Scene	Tree	Triangles	**Total**	Tree	Atlas	**Total**	**Total**	**Total**
Dragon	14.5	12.0	**26.5**	0.65	15.2	**15.8**	**3.86**	**0.38**
Happy Buddha	25.5	23.5	**48.9**	1.04	15.9	**17.0**	**4.13**	**0.40**
Asian Dragon	153.0	123.9	**276.9**	0.90	12.2	**13.1**	**3.43**	**0.40**
Thai Statue	204.4	171.1	**375.5**	2.29	14.3	**16.6**	**4.21**	**0.56**

Table 8.5: Memory consumption (in megabytes) of a triangle-based BVH and our RBVH with quality levels Q0, Q2, and Q4. Bold numbers show the total required memory; other numbers denote the memory required for the tree hierarchy and the geometry, i.e. triangles or atlas, respectively.

	CPU Construction			GPU Construction		
Scene	Q0	Q2	Q4	Q0	Q2	Q4
Dragon	663	530	431	121	101	82
Happy Buddha	1160	901	704	182	152	120
Beast	2395	1758	1472	330	281	239

Table 8.6: Build times (in milliseconds) of the CPU and GPU construction using the spatial median and Q0, Q2, and Q4 quality settings.

As the RBVH is a sample-based structure, its ray casting performance does not depend on the number of triangles. We adjusted the quality levels such that Q0 is visually indistinguishable from the reference for primary rays at a resolution of 1024×1024. The coarser levels, Q1 to Q4, are appropriate for secondary rays, or for primary rays if the object is further from the camera. Here we keep the same distance to better visualize the approximate nature of the RBVH. For primary rays and complex scenes (more than 1.5 million triangles in our examples) the RBVH outperforms the regular BVH, as it does not store more information than actually necessary. The absolute performance of the RBVH depends on the curvature of the surfaces; nevertheless, it linearly grows with decreasing quality for all tested scenes. The relative speed-up of RBVHs to BVHs increases with the scene size.

Memory Requirements. The node hierarchy of the RBVH is typically shallow and the memory requirements are dominated by the texture atlas. The atlas size directly depends on the desired sampling density, which enables us to build compact acceleration structures at the expense of lower accuracy. Table 8.5 shows memory requirements of a triangle-based BVH, where the geometry is represented by indexed vertices, and our RBVH for four different models consisting of 699k (Dragon) to 10.0M triangles (Thai Statue). In all cases, the RBVH requires less memory even at the highest quality level. The most significant *compression* can be observed for the Thai Statue, where Q0 and Q4 require only 4% and 0.2% memory of the regular BVH, respectively. This is obviously achieved by decoupling from the input geometry, which is desirable for removing detail not required for a given resolution or rendering task.

GPU Construction. Our CUDA-based construction algorithm enables using the RBVH for interactive rendering of fully dynamic scenes. Table 8.6 reports the CPU and GPU construction timings of the RBVH for three different quality levels. Compared to existing methods for BVH construction on the GPU, our algorithm seems to build RBVHs for comparable scenes faster than the hybrid LBVH [Lauterbach et al. 2009], on par with HLBVH [Pantaleoni and Luebke 2010]

(the original algorithm constructs an HLBVH for the Dragon model in 81 ms on an NVIDIA GTX 480), but not as fast as the recently introduced more efficient HLBVH [Garanzha et al. 2011]. As most of the concepts introduced in [Garanzha et al. 2011] (e.g. the work queues) are general, we believe that they can be used to further accelerate our current straightforward GPU construction.

8.8   Applications

In this section, we outline some of the applications of RBVHs. In addition to "just accelerate" ray casting, the representation as height fields and the inherent surface parameterization allow several applications, that otherwise require dedicated methods.

Approximate Ray Tracing.   The RBVH enables fast, approximate ray tracing with adaptive accuracy. Note that the term "approximate" does not necessarily mean low-quality: the renderings obtained with quality settings Q0 and Q1 are visually almost indistinguishable from triangle-based rendering even for primary rays. However, in certain global illumination computations, e.g. computing indirect lighting or glossy reflections, the full accuracy is not required. In such cases, primary rays can be efficiently replaced by rasterization, and complemented by an RBVH for secondary rays. Figure 8.10.a shows an example of using the RBVH for glossy reflections and caching diffuse interreflections.

Using the Atlas for Texturing.   Due to the implicit surface parameterization, RBVHs inherently provide means to store surface information. For testing purposes, we implemented three practical applications: real-time photon mapping (see Figure 8.10.b), progressive ambient occlusion (see Figure 8.10.c), and real-time on-surface painting (see Figure 8.11.a). In addition to a traditional accelerator, all of these would normally require additional parameterizations or data structures to store the on-surface signal. With our RBVH we can store surface data by casting rays to determine the corresponding atlas texture coordinates, e.g. for a photon-surface intersection, and retrieve the information later during rendering using the algorithm described in Section 8.4.

Point-Based Rendering.   The RBVH construction supports all primitive types that have finite area and can be subdivided and rasterized. We built RBVHs from point clouds obtained by resampling a scanned bust and directly from a 3D laser (see Figure 8.11.b). During rasterization of the atlas tiles, the point primitives were simply rendered as discs, but any other more sophisticated point rendering technique can be used to improve the surface quality (see [Gross and Pfister 2007] for an overview). In contrast to other accelerators, RBVHs provide ray casting of different primitives in a unified way.

8.9   Discussion and Possible Improvements

The major advantage, but also the most restricting limitation of RBVHs, stems from the fact that we strive to represent the geometry using height fields. While in some cases this provides
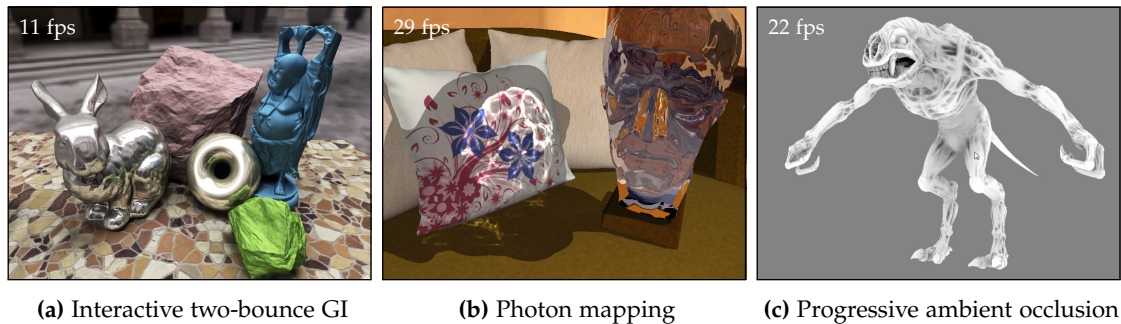
(a) Interactive two-bounce GI   **(b)** Photon mapping   **(c)** Progressive ambient occlusion

Figure 8.10: RBVHs provide advantages over traditional acceleration structures in several applications. The fast ray tracing and the ability to store on-surface signals allow: e.g. (a) interactive two bounce global illumination: glossy reflections are computed using 16 secondary rays; diffuse interreflections are progressively computed using 1 sample per frame and cached in the atlas. (b) real-time photon mapping where the RBVH is used for tracing 400k photons, casting shadow and specular rays; caustics are generated using density estimation of photons stored in the atlas, or (c) progressively computing and caching ambient occlusion. All images were rendered at 1280×720 and cropped for this figure.

(a) Real-time on-surface painting   **(b)** Ray tracing of point clouds
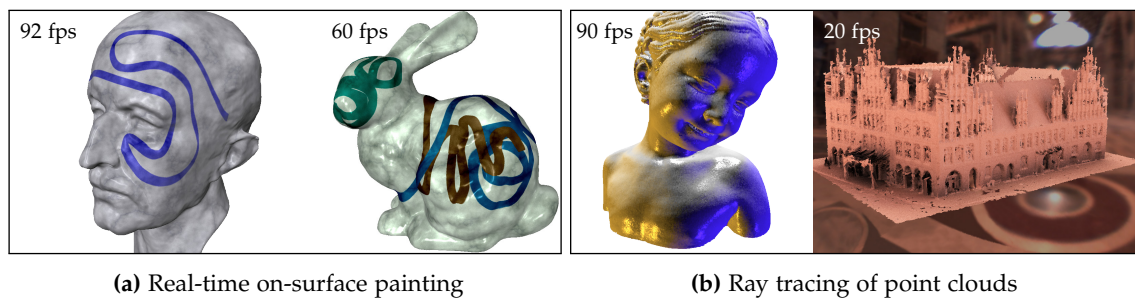
Figure 8.11: RBVH can also be used for real-time on-surface painting (a). The paint is projected onto the surface using ray casting and stored in the atlas. The accelerator also supports other primitives than triangles, such as point-based representations; (b) shows ray casting of the Bimba Con Nastrino and the Old Town Hall in Hannover point clouds.

great compression abilities and fast intersection tests, for high frequency and possibly disconnected surfaces, e.g. foliage or hair, height fields are simply not capable of representing the fine structure efficiently. This is however true for most of the resampling based accelerators. The hybrid RBVH can at least provide smooth transition between fully height field-based or fully triangle-based accelerators. We believe that further investigations, for instance in the direction of enhancing the tree with some statistical data (e.g. the fractional opacity to rays), are an interesting future work.

Surfaces reconstructed using ray casting can occasionally suffer from overlappings and cracks. Both of these can cause visual artifacts under extreme zoom-ins. It would be interesting to extend the current approach to ensure watertightness.

In the future, the RBVH could be further improved by using more elaborate projections, e.g. a reverse perspective [Baboud and Décoret 2006], and signal-specialized tile resolutions. Incremental maintenance via local rasterization and refitting would be another interesting extension, as well as an out-of-core construction for fast visualization of extremely large scenes.

8.10   Conclusion

In this chapter, we presented rasterized bounding volume hierarchies for approximate ray casting of triangle- and point-based surface representations with adjustable level of detail. Our data structure can be efficiently constructed on the CPU and GPU, and provides an inherent surface parameterization for storing data on the surfaces. We described several means to control the accuracy of the resulting RBVH locally and globally, and determined optimal construction parameters for primary and secondary rays. Hybrid RBVHs avoid excessive ray marching and provide high accuracy by partly keeping the input geometry.

Conclusion

One's work may be finished some day,
but one's education never.

— Alexandre Dumas (1802–1870)

In this thesis, we presented several techniques that extend the concept of many-light rendering. We sought to provide high quality results while preserving the elegance and efficiency of many-light approaches. While a large body of previous work exists; which we reviewed in Chapter 3, it rarely meets both of these criteria.

For the classic, VPL-based rendering, we presented two bias compensation techniques, one tailored for surfaces and the other aiming at participating media. When combined into a single algorithm, these can efficiently resolve the local, residual transport, and thus improve the quality without requiring an excessive number of VPLs. The technique can handle glossy surface materials, as well as anisotropic, heterogeneous participating media. Most of the improvements are due to a new mathematical formulation of the residual energy. When combined with several carefully chosen assumptions, this allows for an efficient, hardware-accelerated implementation that generates images close to ground truth at interactive frame rates, i.e. in a fraction of time required by previous techniques.

We also presented a new category of lighting primitives for many-light rendering of scenes with participating media. These leverage entire linear segments of random photon paths, utilizing them either as infinitesimal virtual ray lights, or progressively shrinking virtual beam lights. For each of them, we derived semi-analytic importance sampling for efficiently integrating the light transport in, from, and to surfaces and media. Furthermore, we demonstrated, both mathematically and empirically, that spreading the energy along lines effectively reduces the singularity, and thus oftentimes sidesteps the need for bounding and subsequent bias compensation. In cases when the remaining singularity still causes visible artifacts, we propose to inflate the ray lights into beam lights, and progressively reduce their diameter to achieve convergent results. We show that the new lighting primitives handle anisotropic scattering and heterogeneous participating media well, and can be easily combined with existing progressive photon mapping approaches.

Finally, we devised a new, hierarchical rasterization-based structure for accelerating approximate visibility queries. In addition to fast ray tracing capabilities, we demonstrate that RBVHs are highly scalable and provide adjustable level of detail with accuracy controlled through global and local construction parameters. For the best performance, we analyzed these parameters and determined optimal configurations for coherent and incoherent rays. Furthermore, we addressed

the shortcomings, that may emerge when the input geometry cannot be represented well with height fields, through the hybrid RBVH that combines height fields with the original geometry. We also demonstrated that numerous applications, e.g. photon mapping or irradiance caching, benefit from the inherent surface parametrization that RBVHs provide. The entire data structure is designed to allow fast construction and tracing on parallel architectures, and thanks to the small memory footprint, often fits into the hardware caches.

We believe that many-light algorithms feature properties that are extremely valuable for applications requiring high quality results and predictable render times. The contributions described in this thesis expand these properties to cover a wider spectrum of input scenes, and thus increase the versatility of many-light rendering.

Derivations and Analysis

A.1 Derivation of the Volumetric Three-Point LTE

In order to derive the volumetric three-point formulation of the light transport equation we first express the exitant radiance $L(\mathbf{x} \rightarrow \omega)$. Clearly, we need to distinguish whether $\mathbf{x}$ is on a surface or in a volume:

$$
L(\mathbf{x} \rightarrow \omega) = \begin{cases} L_e(\mathbf{x} \rightarrow \omega) + \int_{\mathcal{S}^2} f_s(\omega \leftarrow \mathbf{x} \leftarrow \omega') |n(\mathbf{x}) \cdot \omega'| L(\mathbf{x} \leftarrow \omega') \mathrm{d}\omega' & \text{if } \mathbf{x} \in \partial\mathbb{V}, \\ \kappa_a(\mathbf{x}) L_e(\mathbf{x} \rightarrow \omega) + \kappa_s(\mathbf{x}) \int_{\mathcal{S}^2} f_p(\omega \leftarrow -\omega') L(\mathbf{x} \leftarrow \omega') \mathrm{d}\omega' & \text{if } \mathbf{x} \in \mathbb{V}. \end{cases} \tag{A.1}
$$

Using the generalized emission $\hat{L}_e$ and the generalized scattering function f we can write the above equation more concisely without distinguishing between surface and volumetric points:

$$
L(\mathbf{x} \rightarrow \omega) = \hat{L}_e(\mathbf{x} \rightarrow \omega) + \int_{\mathcal{S}^2} f(\omega \leftarrow \mathbf{x} \leftarrow \omega') D_\mathbf{x}(\omega') L(\mathbf{x} \leftarrow \omega') \mathrm{d}\omega', \tag{A.2}
$$

where:

$$
D_\mathbf{x}(\omega) = \begin{cases} |n(\mathbf{x}) \cdot \omega| & \text{if } \mathbf{x} \in \partial\mathbb{V}, \\ 1 & \text{if } \mathbf{x} \in \mathbb{V}. \end{cases} \tag{A.3}
$$

We now expand the incident radiance $L(\mathbf{x} \leftarrow \omega')$ according to Equation (2.76) and reorganize the terms to separate contributions of volumes and surfaces:

$$
L(\mathbf{x} \rightarrow \omega) = \hat{L}_e(\mathbf{x} \rightarrow \omega) + \underbrace{\int_{\mathcal{S}^2} f(\omega \leftarrow \mathbf{x} \leftarrow \omega') D_\mathbf{x}(\omega') T(\mathbf{x} \leftrightarrow \mathbf{x}_b) L(\mathbf{x}_b \rightarrow -\omega') \mathrm{d}\omega'}_{\text{contribution of surface points}}
$$
$$
+ \underbrace{\int_{\mathcal{S}^2} \int_0^b f(\omega \leftarrow \mathbf{x} \leftarrow \omega') D_\mathbf{x}(\omega') T(\mathbf{x} \leftrightarrow \mathbf{x}_t) L(\mathbf{x}_t \rightarrow -\omega') \mathrm{d}t \mathrm{d}\omega'}_{\text{contribution of volumetric points}}, \tag{A.4}
$$

where $\mathbf{x}_b = \mathbf{x} + b\omega'$ and $\mathbf{x}_t = \mathbf{x} + t\omega'$. The hemispherical integration of surface contributions can be written as an integral over the set of all surface points $\partial\mathbb{V}$. Similarly, the hemispherical and the surface integrals in the volumetric contribution can be replaced by an integral over the set of all volumetric points $\mathbb{V}$:

$$
L(\mathbf{x} \rightarrow \omega) = \hat{L}_e(\mathbf{x} \rightarrow \omega) + \int_{\partial\mathbb{V}} f(\omega \leftarrow \mathbf{x} \leftarrow \omega') \frac{D_\mathbf{x}(\mathbf{y}) \left| n(\mathbf{y}) \cdot \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|} \right|}{\|\mathbf{x} - \mathbf{y}\|^2} T(\mathbf{x} \leftrightarrow \mathbf{y}) V(\mathbf{x} \leftrightarrow \mathbf{y}) L(\mathbf{y} \rightarrow \mathbf{x}) \mathrm{d}\mathcal{A}(\mathbf{y})
$$
$$
+ \int_{\mathbb{V}} f(\omega \leftarrow \mathbf{x} \leftarrow \omega') \frac{D_\mathbf{x}(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} T(\mathbf{x} \leftrightarrow \mathbf{y}) V(\mathbf{x} \leftrightarrow \mathbf{y}) L(\mathbf{y} \rightarrow \mathbf{x}) \mathrm{d}\mathcal{V}(\mathbf{y}). \tag{A.5}
$$

Finally, since $\partial\mathbb{V} \cap \mathbb{V} = 0$ and $\partial\mathbb{V} \cup \mathbb{V} = \mathbb{R}^3$, we can combine the two integrals into a single one:

$$L(\mathbf{x} \rightarrow \omega) = \hat{L}_e(\mathbf{x} \rightarrow \omega) + \int_{\mathbb{R}^3} f(\omega \leftarrow \mathbf{x} \leftarrow \omega')\hat{G}(\mathbf{x} \leftrightarrow \mathbf{y})T(\mathbf{x} \leftrightarrow \mathbf{y})V(\mathbf{x} \leftrightarrow \mathbf{y})L(\mathbf{y} \rightarrow \mathbf{x})\mathrm{d}\mu(\mathbf{y}), \quad \text{(A.6)}$$

obtaining the three-point formulation of the volumetric light transport equation.

A.2 Derivation of the Path Integral

In order to define the integrand of Equation (2.89) we first expand $L(\mathbf{x} \leftarrow \omega)$ in Equation (2.78) using Equation (2.76):

$$I = \int_{\mathcal{A}_p} \int_{\Omega} W(\mathbf{x} \leftarrow \omega) \left[\int_0^b T(\mathbf{x} \leftrightarrow \mathbf{x}_t)L(\mathbf{x}_t \rightarrow -\omega)\mathrm{d}t \ + \ T(\mathbf{x} \leftrightarrow \mathbf{x}_b)L(\mathbf{x}_b \rightarrow -\omega) \right] \mathrm{d}\omega\mathrm{d}\mathbf{x}. \quad \text{(A.7)}$$

Using the following generalization:

$$W(\mathbf{x} \leftarrow \mathbf{y}) = \begin{cases} W\left(\mathbf{x} \leftarrow \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|^2}\right) & \text{if } \mathbf{x} \in \mathcal{A}_p, \\ 0 & \text{otherwise,} \end{cases} \quad \text{(A.8)}$$

we can write Equation (A.7) as:

$$I = \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} W(\mathbf{x} \leftarrow \mathbf{y})G(\mathbf{x} \leftrightarrow \mathbf{y})T(\mathbf{x} \leftrightarrow \mathbf{y})V(\mathbf{x} \leftrightarrow \mathbf{y})L(\mathbf{y} \rightarrow \mathbf{x})\mathrm{d}\mu(\mathbf{y})\mathrm{d}\mu(\mathbf{x}). \quad \text{(A.9)}$$

This can be recursively expanded using the three point form of the volumetric LTE (cf. Equation (2.79)) with $\omega = \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|^2}$. Doing so k times yields light transport carried by paths with at most $k+2$ vertices. Radiance carried from the light sources to the sensor by paths with exactly k vertices can be written as:

$$I = \int_{\mathcal{P}_k} \hat{L}_e(\mathbf{x}_0 \rightarrow \mathbf{x}_1)$$
$$\prod_{i=1}^{k-1} \left[f(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i \rightarrow \mathbf{x}_{i+1})\hat{G}(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)V(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) \right]$$
$$\hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)W(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1}) \ \mathrm{d}\mu(\mathbf{x}_0) \cdots \mathrm{d}\mu(\mathbf{x}_k), \quad \text{(A.10)}$$

where $\mathbf{x}_0$ is on a light source and $\mathbf{x}_k$ is on the sensor. In order to account for the light transport of all possible lengths, we need to integrate over the entire path space $\mathcal{P}$ defined in Equation (2.88):

$$I = \sum_{k=1}^{\infty} \int_{\mathcal{P}_k} \hat{L}_e(\mathbf{x}_0 \rightarrow \mathbf{x}_1)$$
$$\prod_{i=1}^{k-1} \left[f(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i \rightarrow \mathbf{x}_{i+1})\hat{G}(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)V(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) \right]$$
$$\hat{G}(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)T(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)V(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)W(\mathbf{x}_k \leftarrow \mathbf{x}_{k-1}) \ \mathrm{d}\mu(\mathbf{x}_0) \cdots \mathrm{d}\mu(\mathbf{x}_k), \quad \text{(A.11)}$$

yielding the path integral.

A.3   Construction of the PDF for Sampling VRLs

We first evaluate f_p^{uv} at the two ends of the spherical arc to obtain vertices θ_1 and θ_M. We distribute the remaining $\theta_2 \dots \theta_{M-1}$ interior vertices non-uniformly. Most importantly, we must avoid missing the peak of the underlying function. Though we do not know the exact location of the peak, we do know that the function is a product of two phase functions and that, of these, the phase function along the camera ray must be monotonic between θ_1 and θ_M. Hence, the peak is primarily determined by the remaining phase function along the VRL. If the direction θ_{peak} maximizing the VRL's phase function lies on the arc (i.e. between θ_1 and θ_M), we can easily find it as:

$$\theta_{\text{peak}} = \cos^{-1}(\vec{a} \cdot \vec{e}), \tag{A.12}$$

$$\vec{e} = \text{normalize}((\vec{c} \times \vec{d}) \times \vec{c}), \tag{A.13}$$

$$\vec{c} = \text{normalize}(\vec{a} \times \vec{b}). \tag{A.14}$$

where $\vec{a}$ and $\vec{b}$ are the directions from $\mathbf{x}_{v_i}$ towards the origin and the end of the ray, and $\vec{d}$ is the direction of the VRL. If the peak does not fall within $[\theta_1, \theta_M]$, we invert $\vec{d}$ and repeat the above procedure to find the possible minimum of the VRL's phase function (negative peak). If neither the maximum, nor the minimum fall within $[\theta_1, \theta_M]$, we distribute the interior vertices with a cosine-warped uniform spacing[1]:

$$\theta_j = \frac{\theta_M - \theta_1}{2} \left(1 - \cos\left(\frac{\pi(j-1)}{M-1} \right) \right). \tag{A.15}$$

If $\theta_{\text{peak}} \in [\theta_1, \theta_M]$ then we place the vertex with index:

$$j_{\text{peak}} = \left\lfloor \frac{\theta_{\text{peak}} - \theta_1}{\theta_M - \theta_1}(M-1) + 0.5 \right\rfloor, \tag{A.16}$$

at θ_{peak}, and we distribute the remaining vertices to either side using a similar cosine-warped distribution as in Equation (A.15), but within each subinterval.

This procedure ensures that we always sample the expected location of the peak, and distribute the remaining vertices where the function is expected to be varying the most rapidly.

A.4   Analysis of Singularities

We analyze the singularities present in light transport simulated with VPLs and VRLs. There are three types of transfer we are interested in comparing: point-to-point (P2P), point-to-line (P2L), and line-to-line (L2L) as highlighted in the renderings in Figure 6.10 and illustrated in Figure A.1. Without loss of generality, we assume a canonical geometric configuration and eliminate terms which do not affect limit behavior as the distance between the elements approaches 0. This

---

[1]Applying a cosine to equally-spaced points pushes more points towards the boundary of the interval, where we expect the most variation.

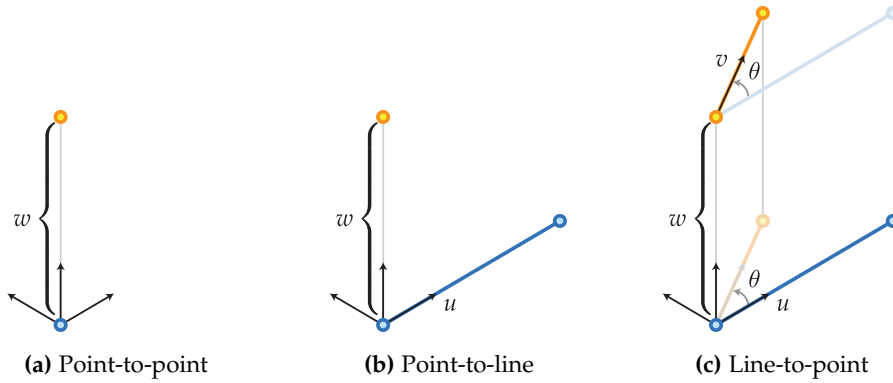**(a)** Point-to-point   **(b)** Point-to-line   **(c)** Line-to-point

Figure A.1: The canonical geometric configuration for the three different transport types.

corresponds to the following three equations:

$$f_{\text{P2P}}(w) = \frac{1}{w^2} \tag{A.17}$$

$$f_{\text{P2L}}(w) = \int_0^1 \frac{1}{u^2 + w^2} du = \frac{1}{w} \tan^{-1}\left(\frac{1}{w}\right) \tag{A.18}$$

$$f_{\text{L2L}}(w, \theta) = \int_0^1 \int_0^1 \frac{1}{(v \cos \theta - u)^2 + (v \sin \theta)^2 + w^2} du dv \tag{A.19}$$

In order, these functions correspond to: the contribution of a VPL to a *point* on a surface/volume; the contribution of a VRL to a *point* on a surface/volume, or the contribution of a VPL integrated along a camera *ray*; and, the contribution of a VRL integrated along a camera *ray*. Note that each successive function integrates over an additional linear domain. We plot these functions in Figure A.2.a.

We will prove that $f_{\text{L2L}}(w) \in o(f_{\text{P2L}}(w))$, and $f_{\text{P2L}}(w) \in o(f_{\text{P2P}}(w))$ as $w \to 0$; where for non-zero functions f and g:

$$f(x) \in o(g(x)) \text{ as } x \to 0 \quad \Longleftrightarrow \quad \lim_{x \to 0} \frac{f(x)}{g(x)} = 0. \tag{A.20}$$

More informally, we will show that the singularity at $w = 0$ for VPLs on surfaces dominates that of VRLs at surfaces, which dominates that of VRLs integrated along a camera ray in a volume.

A.4.1 Point-to-Line vs Point-to-Point Transfer

Plugging in Equations (A.17) and (A.18) into Equation (A.20) we have:

$$\lim_{w \to 0} \frac{f_{\text{P2L}}(w)}{f_{\text{P2P}}(w)} = \lim_{w \to 0} \left[w \tan^{-1}\left(\frac{1}{w}\right) \right] = 0. \tag{A.21}$$

Hence, $f_{\text{P2L}}(w) \in o(f_{\text{P2P}}(w))$: the point-to-line contribution is dominated by the point-to-point contribution as the distance w goes to 0. This limit is illustrated as the blue curve in Figure A.2.b.

Furthermore, since

$$\lim_{w \to 0} \frac{f_{\text{P2L}}(w)}{w} = \frac{\pi}{2},\tag{A.22}$$

we know that $f_{\text{P2L}}(w) \in \Theta(1/w)$ as $w \to 0$; or, in other words, the point-to-line singularity is bounded from above and below by $1/w$ asymptotically. Note that this is in contrast to the point-to-point singularity, which is of order $\Theta(1/w^2)$.

A.4.2 Line-to-Line vs Point-to-Line Transfer

The inner integral of Equation (A.19) has the closed form solution:

$$\frac{\sqrt{2} \left(\tan^{-1} \left(\frac{\sqrt{2}v \cos(\theta)}{d(v,w,\theta)} \right) - \tan^{-1} \left(\frac{\sqrt{2}(v \cos(\theta) - 1)}{d(v,w,\theta)} \right) \right)}{d(v, w, \theta)},\tag{A.23}$$

where $d(v, w, \theta) = \sqrt{v^2(1 - \cos 2\theta) + 2w^2}$. Since the numerator is bounded from above by $\sqrt{2}\pi$, we have

$$f_{\text{L2L}}(w, \theta) \leq \sqrt{2}\pi \int_0^1 \frac{1}{d(v, w, \theta)} dv \tag{A.24}$$

$$= \pi \csc \theta \operatorname{csch}^{-1}(w \csc \theta), \tag{A.25}$$

and, for any fixed $\theta \in (0, \frac{\pi}{2}\rangle$, we can obtain

$$\lim_{w \to 0} \frac{f_{\text{L2L}}(w, \theta)}{f_{\text{P2L}}(w)} \leq \lim_{w \to 0} \frac{\pi \csc \theta \operatorname{csch}^{-1}(w \csc \theta)}{\frac{1}{w} \tan^{-1} \left(\frac{1}{w} \right)} = 0. \tag{A.26}$$

Hence, $f_{\text{L2L}}(w, \theta) \in o(f_{\text{P2L}}(w))$: the line-to-line contribution is dominated by the point-to-line contribution as the distance w goes to 0. This limit is illustrated as the red curve in Figure A.2.b.

Following the same simplification, we can see that

$$\lim_{w \to 0} \frac{f_{\text{L2L}}(w, \theta)}{-\ln(w)} \leq \frac{\pi}{\sin \theta},\tag{A.27}$$

which is non-negative and finite for any $\theta \in (0, \frac{\pi}{2}\rangle$ and hence, $f_{\text{L2L}}(w, \theta) \in O(-\ln(w))$ as $w \to 0$; or, in other words, the line-to-line singularity is bounded from above by $-\ln(w)$.

For the special case where the two lines approach being parallel, the term in the numerator of Equation (A.26) is bounded by a constant: $\lim_{\theta \to 0} f_{\text{L2L}}(w, \theta) \leq \frac{\pi}{w}$. Hence, in this special case it can be shown that $f_{\text{L2L}}(w, 0) \in \Theta(f_{\text{P2L}}(w))$ as $w \to 0$: the singularity of f_{L2L} becomes asymptotically equivalent to that of f_{P2L}.

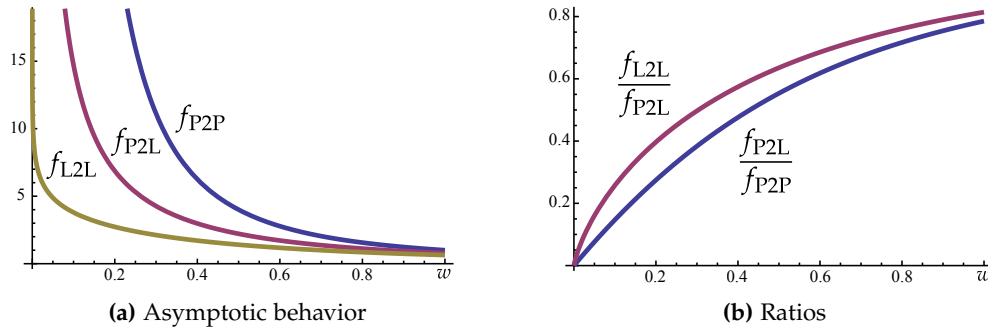**(a)** Asymptotic behavior   **(b)** Ratios

Figure A.2: Asymptotic behavior of the point-to-point, point-to-line and line-to-line contributions, and ratios showing that the singularities become successively weaker.

A.5   Interpretations of VBLs

In this section, we provide further intuition for interpreting virtual beam lights. As mentioned in Section 7.2, a VBL can be seen as a VSL swept along a line (e.g. a VRL). The original *surface* VSLs by Hašan et al. [2009] distribute power of the corresponding VPL over surfaces inside the sphere. In the case of *volumetric* VPLs (or points on a VRL), we need to formulate the energy redistribution differently.

Our goal is to preserve the energy, so that irradiance due to the spherical light at an arbitrary receiving surface point (or the fluence at an arbitrary volumetric point), which is outside the spherical light, is the same as due to the original point emitter. This constrain does not necessarily need to hold for receiving points that are inside the spherical light. The irradiance due to a single, omni-directional point light reads:

$$E(\mathbf{x}) = \Phi \frac{n(\mathbf{x}) \cdot \omega}{4\pi \left\| \mathbf{p} - \mathbf{x} \right\|^2}, \tag{A.28}$$

where Φ is the power emitted by the point light placed at $\mathbf{p}$, $n(\mathbf{x})$ is the normal at the shading point $\mathbf{x}$, and ω is the direction from $\mathbf{x}$ towards $\mathbf{p}$. In the following sections, we present several interpretations that satisfy these constraints.

A.5.1   Volumetric Sphere Light

In order to redistribute the energy of a point emitter, we can uniformly spread it through a spherical, volumetric region $\mathcal{S}$ centered at the emitter. To compute the irradiance at $\mathbf{x}$, we integrate the radiance emitted by $\mathcal{S}$ along rays, which are confined to the solid angle $\Omega_{\mathcal{S}}(\mathbf{x})$ subtended by $\mathcal{S}$:

$$E(\mathbf{x}) = \int_{\Omega_{\mathcal{S}}(\mathbf{x})} (n(\mathbf{x}) \cdot \omega) \int_{t_{\text{near}}}^{t_{\text{far}}} L_e(\mathbf{x}_t \to -\omega) dt d\omega, \tag{A.29}$$

(a) Volumetric sphere light

(b) Encapsulating transmissive sphere

(c) Beam radiance estimation

(d) Emissive spherical cap
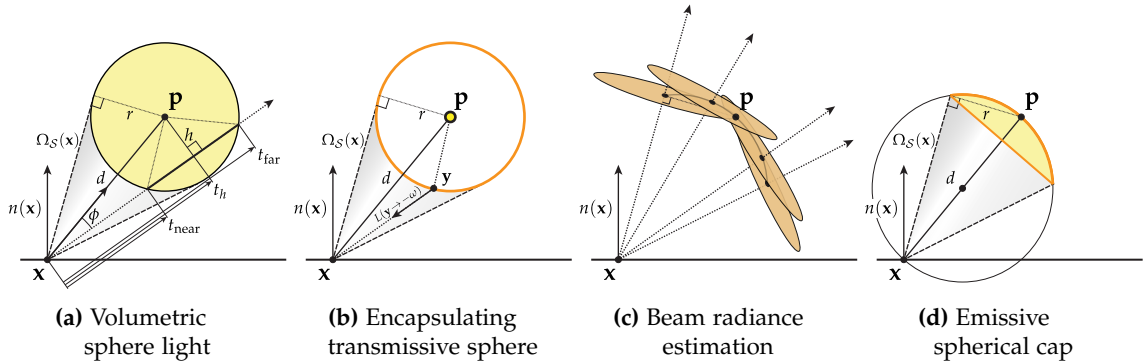
Figure A.3: Four different options for spreading the energy of a volumetric point emitter. The energy can be (a) uniformly spread throughout a spherical region, (b) spread by transmission through an encapsulating sphere, (c) queried using a density estimation technique with a finite kernel, or spread over the surface of a spherical cap shown in (d).

where $\mathbf{x}_t = \mathbf{x} + t\omega$ and:

$$t_{\text{near}} = t_h - \sqrt{r^2 - h^2} \tag{A.30}$$

$$= d\cos\phi - \sqrt{r^2 - d^2\sin^2\phi}, \tag{A.31}$$

$$t_{\text{far}} = d\cos\phi + \sqrt{r^2 - d^2\sin^2\phi}; \tag{A.32}$$

see Figure A.3.a for an illustration of the individual terms.

In order to express the emitted radiance, we first differentiate the power w.r.t the differential solid angle; since the emission is directionally invariant, this amounts to dividing by 4π. Then we uniformly distribute the intensity over the volume $\mathcal{S}$ of the spherical region, resulting in a quantity that we refer to as the *volumetric intensity density* $\left[\text{W}\cdot\text{sr}^{-1}\cdot\text{m}^{-3}\right]$. By integrating this quantity along the segments of rays that originate at $\mathbf{x}$ and overlap with $\mathcal{S}$, we obtain the irradiance as:

$$E(\mathbf{x}) = \int_{\Omega_{\mathcal{S}}(\mathbf{x})} (n(\mathbf{x})\cdot\omega) \int_{t_{\text{near}}}^{t_{\text{far}}} \frac{\Phi}{4\pi}\frac{3}{4\pi r^3}dtd\omega \tag{A.33}$$

$$= \frac{3\Phi}{16\pi^2 r^3} \int_{\Omega_{\mathcal{S}}(\mathbf{x})} (n(\mathbf{x})\cdot\omega)(t_{\text{far}} - t_{\text{near}})d\omega \tag{A.34}$$

$$= \frac{3\Phi}{8\pi^2 r^3} \int_{\Omega_{\mathcal{S}}(\mathbf{x})} (n(\mathbf{x})\cdot\omega)\sqrt{r^2 - d^2(1-(\omega_{\mathbf{p}}\cdot\omega)^2)}d\omega. \tag{A.35}$$

While this equation is not quite the same as Equation (A.28), we empirically verified that the irradiance due to the volumetric sphere light is the same as due to a point light, as long as $\mathbf{x}$ is outside $\mathcal{S}$.

A.5.2 Encapsulating Transmissive Sphere Light

Another option is to encapsulate the point emitter with a diffuse, perfectly transmissive sphere $\mathcal{S}$, see Figure A.3.b for an illustration. In order to compute $E(\mathbf{x})$, we will integrate the incident

radiance over the solid angle $\Omega_S(\mathbf{x})$ subtended by the sphere:

$$E(\mathbf{x}) = \int_{\Omega_S(\mathbf{x})} L(\mathbf{x} \leftarrow \omega)(n(\mathbf{x}) \cdot \omega)d\omega. \tag{A.36}$$

The incident radiance $L(\mathbf{x} \leftarrow \omega)$ can be expressed as radiance exitant from point $\mathbf{y}$ on the surface of the sphere in direction $-\omega$: $L(\mathbf{x} \leftarrow \omega) = L(\mathbf{y} \rightarrow \omega)$, where $\mathbf{y} = r(\mathbf{x}, \omega)$. Threating the surface transmission of S as diffuse, we can factor the BTDF $f_s(\omega \leftarrow \mathbf{y} \leftarrow \omega')$ out of the integral and replace it with the hemispherical transmittance (i.e. the analog of albedo for transmission), which for a perfectly transmissive surface equals $1/\pi$. The remaining integral then expresses the irradiance at the inner side of the sphere, which can be computed by dividing the power of the point light by the surface area of the sphere:

$$\begin{aligned} L(\mathbf{y} \rightarrow -\omega) &= \frac{1}{\pi} \int_{\mathcal{H}^2} L(\mathbf{y} \leftarrow \omega')(-n(\mathbf{y}) \cdot \omega')d\omega' & \text{(A.37)} \\ &= \frac{1}{\pi} E(\mathbf{y}) & \text{(A.38)} \\ &= \frac{\Phi}{4\pi^2 r^2}, & \text{(A.39)} \end{aligned}$$

where r is the radius of S. Inserting this expression into Equation (A.36) yields the final formula for computing the irradiance at $\mathbf{x}$:

$$E(\mathbf{x}) = \frac{\Phi}{4\pi^2 r^2} \int_{\Omega_S(\mathbf{x})} (n(\mathbf{x}) \cdot \omega)d\omega. \tag{A.40}$$

One advantage of Equation (A.40) over Equation (A.35) is that the integrand consists of only a dot product; therefore, a numeric evaluation will be less expensive.

A.5.3 Blurring via Density Estimation

We can also gather the energy from a point emitter using a density estimation approach, e.g. the beam radiance estimate [Jarosz et al. 2008b]. We shoot rays through the upper hemisphere at $\mathbf{x}$, find the closest point on the ray to $\mathbf{p}$ and perform 2D density estimation using a radius r. Using a constant estimation kernel $1/\pi r^2$, the irradiance computed by a BRE can be written as:

$$E(\mathbf{x}) = \frac{\Phi}{4\pi^2 r^2} \int_{\Omega_S(\mathbf{x})} (n(\mathbf{x}) \cdot \omega)d\omega, \tag{A.41}$$

which yields the same equation as the encapsulating transmissive sphere.

Notice that the set of rays' closest points to $\mathbf{p}$ form a spherical cap (see Figure A.3.d). Another option for spreading the energy is to distribute it over a virtual, emissive surface of the spherical cap. It can be shown that the cap has the area πr^2 and subtends solid angle $\Omega_S(\mathbf{x})$. Tracing rays confined to $\Omega_S(\mathbf{x})$ and gathering emitted radiance from the virtual, emissive surface of the cap yields again the same equation as the BRE or the encapsulating sphere formulations.

Despite being conceptually different, all these formulations yield visually equivalent results. However, using the BRE approach, we can rely on previous publications that provide converging results with limited memory footprint. We can apply this theory to our case and progressively reduce the size of the spreading region so that the inherent bias diminishes in the limit.

Bibliography

AILA, T. and LAINE, S. [2009]. Understanding the efficiency of ray traversal on GPUs. In *Proc. of High Performance Graphics*, pp. 145–149. ACM, New York, NY, USA.

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., and KAUTZ, J. [2008]. Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 27(3):34:1–34:8.

APPEL, A. [1968]. Some techniques for shading machine renderings of solids. In *Proc. of the Spring Joint Computer Conference*, pp. 37–45. ACM, New York, NY, USA.

ARBREE, A., WALTER, B., and BALA, K. [2008]. Single-pass scalable subsurface rendering with lightcuts. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):507–516.

ARIKAN, O., FORSYTH, D. A., and O'BRIEN, J. F. [2005]. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1108–1114.

ARVO, J. [1986]. Backward ray tracing. In *In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, pp. 259–263.

ARVO, J. R. [1995]. *Analytic methods for simulated light transport*. Ph.D. thesis, Yale University, New Haven, CT, USA.

ARVO, J. R., TORRANCE, K., and SMITS, B. [1994]. A framework for the analysis of error in global illumination algorithms. In *Proc. of SIGGRAPH 94*, Annual Conference Series, pp. 75–84. ACM, New York, NY, USA.

AUPPERLE, L. and HANRAHAN, P. [1993]. A hierarchical illumination algorithm for surfaces with glossy reflection. In *Proc. of SIGGRAPH 93*, Annual Conference Series, pp. 155–162. ACM, New York, NY, USA.

BABOUD, L. and DÉCORET, X. [2006]. Rendering geometry with relief textures. In *Proc. of Graphics Interface*, pp. 195–201. Canadian Information Processing Society, Toronto, Ont., Canada.

BARAN, I., CHEN, J., RAGAN-KELLEY, J., DURAND, F., and LEHTINEN, J. [2010]. A hierarchical volumetric shadow algorithm for single scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 29(6):178:1–178:10.

BARANOSKI, G. V. G. and KRISHNASWAMY, A. [2010]. *Light & Skin Interactions: Simulations for Computer Graphics Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

BARRINGER, R., GRIBEL, C. J., and AKENINE-MÖLLER, T. [2012]. High-quality curve rendering using line sampled visibility. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 31(6):162:1–162:10.

BAVOIL, L., SAINZ, M., and DIMITROV, R. [2008]. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 talks*.

BIRI, V., ARQUÈS, D., and MICHELIN, S. [2006]. Real time rendering of atmospheric scattering and volumetric shadows. *Journal of WSCG*, 14(1):65–72.

BLASI, P., LE SAEC, B., and SCHLICK, C. [1993]. A rendering algorithm for discrete volume density objects. *Computer Graphics Forum (Proc. of Eurographics '93)*, 12(3):201–210.

BORN, M. and WOLF, E. [1999]. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press.

BOUBEKEUR, T., HEIDRICH, W., GRANIER, X., and SCHLICK, C. [2006]. Volume-surface trees. *Computer Graphics Forum (Proc. of Eurographics)*, 25(3):399–406.

BOUDET, A., PITOT, P., PRATMARTY, D., and PAULIN, M. [2005]. Photon splatting for participating media. In *Proc. of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pp. 197–204. ACM, New York, NY, USA.

BOUTHORS, A. [2008]. *Realistic rendering of clouds in real-time / Rendu réaliste de nuages en temps-réel*. Ph.D. thesis, Université Joseph Fourier, Grenoble, France.

CARR, N. A., HOBEROCK, J., CRANE, K., and HART, J. C. [2006]. Fast GPU ray tracing of dynamic meshes using geometry images. In *Proc. of Graphics Interface*, pp. 203–209. Canadian Information Processing Society, Toronto, Ont., Canada.

CEREZO, E., PÉREZ, F., PUEYO, X., SERON, F. J., and SILLION, F. X. [2005]. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328.

CHANDRASEKHAR, S. [1960]. *Radiative Transfer*. Dover Publications.

CHOI, B., KOMURAVELLI, R., LU, V., SUNG, H., BOCCHINO, R. L., ADVE, S. V., and HART, J. C. [2010]. Parallel SAH k-D tree construction. In *Proc. of High Performance Graphics*, pp. 77–86. Eurographics Association, Aire-la-Ville, Switzerland.

CHRISTENSEN, P. H. [2003]. Adjoints and importance in rendering: An overview. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):329–340.

CIE [1987]. *CIE International Lighting Vocabulary : IEC International Electrotechnical Vocabulary*. CIE, 4th edn.

CLINE, D., TALBOT, J., and EGBERT, P. [2005]. Energy redistribution path tracing. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1186–1195.

COHEN, M. F. and GREENBERG, D. P. [1985]. The hemi-cube: a radiosity solution for complex environments. *Computer Graphics (Proc. of SIGGRAPH)*, 19(3):31–40.

COHEN-OR, D., CHRYSANTHOU, Y. L., SILVA, C. T., and DURAND, F. [2003]. A survey of visibility for walk-through applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):412–431.

COLEMAN, W. A. [1968]. Mathematical verification of a certain Monte Carlo sampling technique and applications of the technique to radiation transport problems. *Nuclear Science and Engineering*, 32(1):76–81.

COOK, R. L., PORTER, T., and CARPENTER, L. [1984]. Distributed ray tracing. *Computer Graphics (Proc. of SIGGRAPH)*, 18(3):137–145.

CRASSIN, C. [2011]. *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes*. Ph.D. thesis, Université de Grenoble.

CRASSIN, C., NEYRET, F., LEFEBVRE, S., and EISEMANN, E. [2009]. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 15–22. ACM, New York, NY, USA.

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., and EISEMANN, E. [2011]. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 30(7):207–207.

DACHSBACHER, C. [2010]. Analyzing visibility configurations. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):475–486.

DACHSBACHER, C., KŘIVÁNEK, J., HAŠAN, M., ARBREE, A., WALTER, B., and NOVÁK, J. [2014]. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33(1):88–104.

DACHSBACHER, C. and STAMMINGER, M. [2003]. Translucent shadow maps. In *Proc. of Eurographics Symposium on Rendering*, pp. 197–201. Eurographics Association, Aire-la-Ville, Switzerland.

DACHSBACHER, C. and STAMMINGER, M. [2005]. Reflective shadow maps. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 203–208. ACM, New York, NY, USA.

DACHSBACHER, C. and STAMMINGER, M. [2006]. Splatting indirect illumination. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 93–100. ACM, New York, NY, USA.

DAMMERTZ, H., HANIKA, J., and KELLER, A. [2008]. Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):1225–1233.

DAMMERTZ, H. and KELLER, A. [2008]. Edge volume heuristic - robust triangle subdivision for improved BVH performance. In *Proc. of IEEE Symposium on Interactive Ray Tracing*, pp. 155–158.

DAMMERTZ, H., KELLER, A., and LENSCH, H. [2010]. Progressive point-light-based global illumination. *Computer Graphics Forum*, 29(8):2504–2515.

DAVIDOVIČ, T., GEORGIEV, I., and SLUSALLEK, P. [2012]. Progressive lightcuts for GPU. In *ACM SIGGRAPH 2012 Talks*.

DAVIDOVIČ, T., KŘIVÁNEK, J., HAŠAN, M., SLUSALLEK, P., and BALA, K. [2010]. Combining global and local virtual lights for detailed glossy illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 29(6):143:1–143:8.

DE TOLEDO, R., WANG, B., and LÉVY, B. [2008]. Geometry textures and applications. *Computer Graphics Forum*, 27(8):2053–2065.

DEERING, M., WINNER, S., SCHEDIWY, B., DUFFY, C., and HUNT, N. [1988]. The triangle processor and normal vector shader: a VLSI system for high performance graphics. *Computer Graphics (Proc. of SIGGRAPH)*, 22(4):21–30.

D'EON, E. and IRVING, G. [2011]. A quantized-diffusion model for rendering translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 30(4):56:1–56:14.

D'EON, E., LUEBKE, D., and ENDERTON, E. [2007]. Efficient rendering of human skin. In *Proc. of Eurographics Symposium on Rendering*, pp. 147–157. Eurographics Association, Aire-la-Ville, Switzerland.

DONG, Z., GROSCH, T., RITSCHEL, T., and SEIDEL, H.-P. [2009]. Real-time indirect illumination with clustered visibility. In *Proc. of Vision, Modeling and Visualization*, pp. 187–196. Otto-Von-Guericke-Universität, Braunschweig, Germany.

DONNER, C. and JENSEN, H. W. [2005]. Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1032–1039.

DUTRÉ, P., BALA, K., and BEKAERT, P. [2006]. *Advanced Global Illumination*. A K Peters, 2nd edn.

ENGELHARDT, T. and DACHSBACHER, C. [2010]. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 119–125. ACM, New York, NY, USA.

ENGELHARDT, T., NOVÁK, J., and DACHSBACHER, C. [2010]. Instant multiple scattering for interactive rendering of heterogeneous participating media. Tech. rep., Karlsruhe Institute of Technology, Germany.

ENGELHARDT, T., NOVÁK, J., SCHMIDT, T.-W., and DACHSBACHER, C. [2012]. Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 31(7):2145–2154.

ERNST, M. and GREINER, G. [2007]. Early split clipping for bounding volume hierarchies. In *Proc. of IEEE Symposium on Interactive Ray Tracing*, pp. 73–78. IEEE Computer Society, Washington, DC, USA.

FEYNMAN, R. and HIBBS, A. [1965]. *Quantum Mechanics and Path Integrals*. International series in pure and applied physics. McGraw-Hill.

FICK, A. [1855]. Über Diffusion. *Annalen der Physik*, 170(1):59–86.

FRISVAD, J. R., CHRISTENSEN, N. J., and JENSEN, H. W. [2007]. Computing the scattering properties of participating media using lorenz-mie theory. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 26(3).

GARANZHA, K., PANTALEONI, J., and MCALLISTER, D. [2011]. Simpler and faster HLBVH with work queues. In *Proc. of High Performance Graphics*, pp. 59–64. ACM, New York, NY, USA.

GELBARD, E. M., ONDIS, II, L. A., and SPANIER, J. [1966]. A new class of Monte Carlo estimators. *SIAM Journal on Applied Mathematics*, 14(4):697–701.

GEORGIEV, I., KŘIVÁNEK, J., POPOV, S., and SLUSALLEK, P. [2012]. Importance caching for complex illumination. *Computer Graphics Forum (Proc. of Eurographics)*, 31(3):701–710.

GEORGIEV, I., KŘIVÁNEK, J., HACHISUKA, T., NOWROUZEZAHRAI, D., and JAROSZ, W. [2013]. Joint importance sampling of low-order volumetric scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 32(6):164:1–164:14.

GEORGIEV, I. and SLUSALLEK, P. [2010]. Simple and robust iterative importance sampling of virtual point lights. In *Eurographics 2010 - Short Papers*, pp. 57–60. Eurographics Association, Norrköping, Sweden.

GIBBSONS, M. G. [1958]. Radiation received by an uncollimated receiver from a 4π source. *Journal of the Optical Society of America*, 48(8):550–555.

GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., and BATTAILE, B. [1984]. Modeling the interaction of light between diffuse surfaces. *Computer Graphics (Proc. of SIGGRAPH)*, 18(3):213–222.

GRADSHTEYN, I. S. and RYZHIK, I. M. [2007]. *Table of Integrals, Series, and Products*. Academic Press, Amsterdam, seventh edn.

GRASSMANN, A. and PETERS, F. [2004]. Size measurement of very small spherical particles by mie scattering imaging (MSI). *Particle & Particle Systems Characterization*, 21(5):379–389.

GRIBEL, C. J., DOGGETT, M., and AKENINE-MÖLLER, T. [2010]. Analytical motion blur rasterization with compression. In *Proc. of High Performance Graphics*, pp. 163–172. Eurographics Association, Aire-la-Ville, Switzerland.

GROSS, M. and PFISTER, H.-P. [2007]. *Point-Based Graphics*. Elsevier.

HABEL, R., CHRISTENSEN, P. H., and JAROSZ, W. [2013]. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 32(4).

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., and JENSEN, H. W. [2008a]. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 27(3):33:1–33:10.

Hachisuka, T. and Jensen, H. W. [2009]. Stochastic progressive photon mapping. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5):141:1–141:8.

Hachisuka, T., Ogaki, S., and Jensen, H. W. [2008b]. Progressive photon mapping. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 27(5):130:1–130:8.

Hammersley, J. M. and Handscomb, D. C. [1964]. *Monte Carlo Methods*. Monographs on statistics and applied probability. Chapman and Hall, London, New York.

Hanika, J., Keller, A., and Lensch, H. P. A. [2010]. Two-level ray tracing with reordering for highly complex scenes. In *Proc. of Graphics Interface*, pp. 145–152. Canadian Information Processing Society, Toronto, Ont., Canada.

Havran, V. [2000]. *Heuristic Ray Shooting Algorithms*. Ph.D. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague.

Havran, V., Bittner, J., Herzog, R., and Seidel, H.-P. [2005]. Ray maps for global illumination. In *Proc. of Eurographics Symposium on Rendering*, pp. 43–54. Eurographics Association, Aire-la-Ville, Switzerland.

Hašan, M., Křivánek, J., Walter, B., and Bala, K. [2009]. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5):1–6.

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

Hašan, M., Velázquez-Armendariz, E., Pellacini, F., and Bala, K. [2008]. Tensor clustering for rendering many-light animations. In *Proc. of Eurographics Symposium on Rendering*, pp. 1105–1114. Eurographics Association, Aire-la-Ville, Switzerland.

Heckbert, P. S. [1991]. *Simulating Global Illumination using Adaptive Meshing*. Ph.D. thesis, EECS Department, University of California, Berkeley.

Heidrich, W., Brabec, S., and Seidel, H.-P. [2000]. Soft shadow maps for linear lights. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 269–280. Springer-Verlag, London, UK.

Henyey, L. G. and Greenstein, J. L. [1941]. Diffuse radiation in the galaxy. *Astrophysical Journal*, 93:70–83.

Herzog, R., Havran, V., Kinuwaki, S., Myszkowski, K., and Seidel, H.-P. [2007]. Global illumination using photon ray splatting. *Computer Graphics Forum (Proc. of Eurographics)*, 26(3):503–513.

Hoberock, J. and Bell, N. [2010]. Thrust: A parallel template library. Version 1.7.0.

Holländer, M., Ritschel, T., Eisemann, E., and Boubekeur, T. [2011]. Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 30(4).

Hoppe, H. [1997]. View-dependent refinement of progressive meshes. In *Proc. of SIGGRAPH 97*, Annual Conference Series, pp. 189–198. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Howell, J. R., Siegel, R., and Menguc, M. P. [2010]. *Thermal Radiation Heat Transfer*. CRC Press, 5th edn.

Immel, D. S., Cohen, M. F., and Greenberg, D. P. [1986]. A radiosity method for non-diffuse environments. *Computer Graphics (Proc. of SIGGRAPH)*, 20(4):133–142.

Jakob, W. and Marschner, S. [2012]. Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):58:1–58:13.

JAROSZ, W., DONNER, C., ZWICKER, M., and JENSEN, H. W. [2008a]. Radiance caching for participating media. *ACM Transactions on Graphics*, 27(1):7:1–7:11.

JAROSZ, W., NOWROUZEZAHRAI, D., SADEGHI, I., and JENSEN, H. W. [2011a]. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics*, 30(1):5:1–5:19.

JAROSZ, W., NOWROUZEZAHRAI, D., THOMAS, R., SLOAN, P.-P., and ZWICKER, M. [2011b]. Progressive photon beams. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 30(6):181:1–181:12.

JAROSZ, W., ZWICKER, M., and JENSEN, H. W. [2008b]. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):557–566.

JENSEN, H. W. [1996]. Global illumination using photon maps. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 21–30. Springer-Verlag, London, UK.

JENSEN, H. W. and BUHLER, J. [2002]. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):576–581.

JENSEN, H. W. and CHRISTENSEN, P. H. [1998]. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proc. of SIGGRAPH 98*, Annual Conference Series, pp. 311–320. ACM, New York, NY, USA.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., and HANRAHAN, P. [2001]. A practical model for subsurface light transport. In *Proc. of SIGGRAPH 01*, Annual Conference Series, pp. 511–518. ACM, New York, NY, USA.

JIMENEZ, J., SUNDSTEDT, V., and GUTIERREZ, D. [2009]. Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception*, 6(4):23:1–23:15.

JONES, T. R. and PERRY, R. N. [2000]. Antialiasing with line samples. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 197–206. Springer-Verlag, London, UK.

KAJIYA, J. T. [1986]. The rendering equation. *Computer Graphics (Proc. of SIGGRAPH)*, pp. 143–150.

KAJIYA, J. T. and VON HERZEN, B. P. [1984]. Ray tracing volume densities. *Computer Graphics (Proc. of SIGGRAPH)*, 18(3):165–174.

KALOJANOV, J., BILLETER, M., and SLUSALLEK, P. [2011]. Two-level grids for ray tracing on GPUs. *Computer Graphics Forum (Proc. of Eurographics)*, 30(2):307–314.

KALOS, M. H. [1963]. On the estimation of flux at a point by Monte Carlo. *Nuclear Science and Engineering*, 16:111–117.

KALOS, M. H. and WHITLOCK, P. A. [1986]. *Monte Carlo methods. Vol. 1: basics*. Wiley-Interscience, New York, NY, USA.

KAPLANYAN, A. S. and DACHSBACHER, C. [2013a]. Adaptive progressive photon mapping. *ACM Transactions on Graphics*, 32(2):16:1–16:13.

KAPLANYAN, A. S. and DACHSBACHER, C. [2013b]. Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Proc. of Eurographics)*, 32(2):63–72.

KARRAS, T. [2012]. Maximizing parallelism in the construction of BVHs, octrees, and kD-trees. In *Proc. of High Performance Graphics*, pp. 33–37. Eurographics Association, Aire-la-Ville, Switzerland.

KARRAS, T. and AILA, T. [2013]. Fast parallel construction of high-quality bounding volume hierarchies. In *Proc. of High Performance Graphics*, pp. 89–99. ACM, New York, NY, USA.

KELLER, A. [1997]. Instant radiosity. In *Proc. of SIGGRAPH 97*, Annual Conference Series, pp. 49–56. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

KELLER, A. and HEIDRICH, W. [2001]. Interleaved sampling. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 269–276. Springer-Verlag, London, UK.

KNAUS, C. and ZWICKER, M. [2011]. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics*, 30(3):25:1–25:13.

KOLLIG, T. and KELLER, A. [2006]. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pp. 245–257. Springer.

KŘIVÁNEK, J., GAUTRON, P., PATTANAIK, S., and BOUATOUCH, K. [2005]. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550 –561.

KULLA, C. and FAJARDO, M. [2012]. Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 31(4):1519–1528.

KŘIVÁNEK, J., FERWERDA, J. A., and BALA, K. [2010]. Effects of global illumination approximations on material appearance. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29(4):112:1–112:10.

LAFORTUNE, E. P. and WILLEMS, Y. D. [1993]. Bi-directional path tracing. In *Compugraphics '93*, pp. 145–153.

LAFORTUNE, E. P. and WILLEMS, Y. D. [1996]. Rendering participating media with bidirectional path tracing. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 91–100. Springer-Verlag, London, UK.

LAINE, S. and KARRAS, T. [2010]. Efficient sparse voxel octrees. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 55–63. ACM, New York, NY, USA.

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J., and AILA, T. [2007]. Incremental instant radiosity for real-time indirect illumination. In *Proc. of Eurographics Symposium on Rendering*, pp. 277–286. Eurographics Association, Aire-la-Ville, Switzerland.

LASTRA, M., UREÑA, C., REVELLES, J., and MONTES, R. [2002]. A particle-path based method for monte carlo density estimation. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 7–14. Eurographics Association, Aire-la-Ville, Switzerland.

LAUTERBACH, C., GARLAND, M., SENGUPTA, S., LUEBKE, D., and MANOCHA, D. [2009]. Fast BVH construction on GPUs. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):375–384.

LECOCQ, P., KEMENY, A., MICHELIN, S., and ARQUÈS, D. [2000]. Mathematical approximation for real-time lighting rendering through participating media. In *Proc. of the 8th Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, Washington, DC, USA.

LEFEBVRE, S. and DACHSBACHER, C. [2007]. Tiletrees. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 25–31. ACM, New York, NY, USA.

LEHTINEN, J., KARRAS, T., LAINE, S., AITTALA, M., DURAND, F., and AILA, T. [2013]. Gradient-domain metropolis light transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4):95:1–95:12.

LEPPÄNEN, J. [2010]. Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code. *Annals of Nuclear Energy*, 37(5):715–722.

LISCHINSKI, D., TAMPIERI, F., and GREENBERG, D. P. [1992]. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39.

LORENZ, L. [1890]. Lysbevægelsen i og uden for en af plane lysbølger belyst kugle. In *Det Kongelige Danske Videnskabernes Selskabs Skrifter (trykt utg.): Naturvidenskabelig og Mathematisk Afdeling*.

Lux, I. and Koblinger, L. [1991]. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC Press.

Metropolis, N. and Ulam, S. M. [1949]. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

Mie, G. [1908]. Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Annalen der Physik*, 330:377–445.

Moon, J. T. and Marschner, S. R. [2006]. Simulating multiple scattering in hair using a photon mapping approach. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 25(3):1067–1074.

Nichols, G., Shopf, J., and Wyman, C. [2009]. Hierarchical image-space radiosity for interactive global illumination. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 28(4):1141–1149.

Nichols, G. and Wyman, C. [2009]. Multiresolution splatting for indirect illumination. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 83–90. ACM, New York, NY, USA.

Nichols, G. and Wyman, C. [2010]. Interactive indirect illumination using adaptive multiresolution splatting. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):729–741.

Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., and Limperis, T. [1977]. *Geometrical Considerations and Nomenclature for Reflectance*. Monograph 161. National Bureau of Standards (US).

Nishita, T., Miyawaki, Y., and Nakamae, E. [1987]. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. *Computer Graphics (Proc. of SIGGRAPH)*, pp. 303–310.

Nishita, T. and Nakamae, E. [1985]. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. *Computer Graphics (Proc. of SIGGRAPH)*, 19(3):23–30.

Novák, J. and Dachsbacher, C. [2012]. Rasterized bounding volume hierarchies. *Computer Graphics Forum (Proc. of Eurographics)*, 31(3):403–412.

Novák, J., Engelhardt, T., and Dachsbacher, C. [2011a]. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 119–124. ACM.

Novák, J., Havran, V., and Dachsbacher, C. [2010]. Path regeneration for interactive path tracing. In *Eurographics 2010 - Short Papers*, pp. 61–64. Eurographics Association.

Novák, J., Havran, V., and Dachsbacher, C. [2011b]. Path regeneration for random walks. In *GPU Computing Gems*, edited by W. mei W. Hwu, chapter 26, pp. 401–411. Morgan Kaufmann.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. [2012a]. Progressive virtual beam lights. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 31(4):1407–1413.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. [2012b]. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):60:1–60:11.

Olsson, O. and Assarsson, U. [2011]. Tiled shading. *Journal of Graphics, GPU, and Game Tools*, 15(4):235–251.

Olsson, O., Billeter, M., and Assarsson, U. [2012]. Clustered deferred and forward shading. In *Proc. of High Performance Graphics*, pp. 87–96. Eurographics Association, Aire-la-Ville, Switzerland.

Ou, J. and Pellacini, F. [2011]. Lightslice: Matrix slice sampling for the many-lights problem. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 30(6):179:1–179:8.

PANTALEONI, J., FASCIONE, L., HILL, M., and AILA, T. [2010]. PantaRay: fast ray-traced occlusion caching of massive scenes. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29(4):37:1–37:10.

PANTALEONI, J. and LUEBKE, D. [2010]. HLBVH: hierarchical LBVH construction for real-time ray tracing of dynamic geometry. In *Proc. of High Performance Graphics*, pp. 87–95. Eurographics Association, Aire-la-Ville, Switzerland.

PATTANAIK, S. N. and MUDUR, S. P. [1993]. Efficient potential equation solutions for global illumination computation. *Computers & Graphics*, 17(4):387–396.

PAULY, M., KOLLIG, T., and KELLER, A. [2000]. Metropolis light transport for participating media. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 11–22. Springer-Verlag, London, UK.

PEGORARO, V. [2009]. *Efficient Physically-Based Simulation of Light Transport in Participating Media*. Ph.D. thesis, University of Utah.

PEGORARO, V. and PARKER, S. G. [2009]. An analytical solution to single scattering in homogeneous participating media. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):329–335.

PEGORARO, V., SCHOTT, M., and PARKER, S. G. [2009]. An analytical approach to single scattering for anisotropic media and light distributions. In *Proc. of Graphics Interface*, pp. 71–77. Canadian Information Processing Society, Toronto, Ont., Canada.

PEGORARO, V., SCHOTT, M., and PARKER, S. G. [2010]. A closed-form solution to single scattering for general phase functions and light distributions. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 29(4):1365–1374.

PEGORARO, V., SCHOTT, M., and SLUSALLEK, P. [2011]. A mathematical framework for efficient closed-form single scattering. In *Proc. of the 37th Graphics Interface Conference*, pp. 151–158.

PERLIN, K. H. and HOFFERT, E. M. [1989]. Hypertexture. *Computer Graphics (Proc. of SIGGRAPH)*, 23(3):253–262.

PHARR, M. and HUMPHREYS, G. [2010]. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edn.

POPOV, S., GEORGIEV, I., DIMOV, R., and SLUSALLEK, P. [2009]. Object partitioning considered harmful: Space subdivision for BVHs. In *Proc. of High Performance Graphics*, pp. 15–22. ACM, New York, NY, USA.

PREMOŽE, S., ASHIKHMIN, M., and SHIRLEY, P. [2003]. Path integration for light transport in volumes. In *Proc. of Eurographics Symposium on Rendering*, pp. 52–63. Eurographics Association, Aire-la-Ville, Switzerland.

PREMOŽE, S., ASHIKHMIN, M., TESSENDORF, J., RAMAMOORTHI, R., and NAYAR, S. [2004]. Practical rendering of multiple scattering effects in participating media. In *Proc. of Eurographics Symposium on Rendering*, pp. 363–374. Eurographics Association, Aire-la-Ville, Switzerland.

PRUTKIN, R., KAPLANYAN, A., and DACHSBACHER, C. [2012]. Reflective shadow map clustering for real-time global illumination. In *Eurographics 2012 - Short Papers*. Eurographics Association, Cagliari, Italy.

RAAB, M., SEIBERT, D., and KELLER, A. [2008]. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pp. 591–606. Springer.

RAYLEIGH, J. W. S. L. [1871]. On the scattering of light by small particles. *Philosophical Magazine*, 64:447–454.

RITSCHEL, T., EISEMANN, E., HA, I., KIM, J., and SEIDEL, H.-P. [2011]. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum*, 30(8):2258–2269.

Ritschel, T., Engelhardt, T., Grosch, T., Seidel, H.-P., Kautz, J., and Dachsbacher, C. [2009a]. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 28(5):132:1–132:8.

Ritschel, T., Grosch, T., Kim, M. H., Seidel, H. P., Dachsbacher, C., and Kautz, J. [2008]. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 27(5):129:1–129:8.

Ritschel, T., Grosch, T., and Seidel, H.-P. [2009b]. Approximating dynamic global illumination in image space. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pp. 75–82. ACM, New York, NY, USA.

Rushmeier, H. [1995]. Rendering participating media: Problems and solutions from application areas. In *Photorealistic Rendering Techniques*, Focus on Computer Graphics, pp. 37–59. Springer Berlin Heidelberg.

Rushmeier, H. E. and Torrance, K. E. [1987]. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics (Proc. of SIGGRAPH)*, 21(4):293–302.

Saito, T. and Takahashi, T. [1990]. Comprehensible rendering of 3-D shapes. *Computer Graphics (Proc. of SIGGRAPH)*, 24(4):197–206.

Saleh, B. and Teich, M. [2007]. *Fundamentals of Photonics*. Wiley Series in Pure and Applied Optics. John Wiley & Sons.

Salvi, M., Vidimce, K., Lauritzen, A., and Lefohn, A. [2010]. Adaptive volumetric shadow maps. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 29(4):1289–1296.

Scherzer, D., Nguyen, C. H., Ritschel, T., and Seidel, H.-P. [2012]. Pre-convolved radiance caching. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 4(31):1391–1397.

Schmidt, T.-W., Novák, J., Meng, J., Kaplanyan, A. S., Reiner, T., Nowrouzezahrai, D., and Dachsbacher, C. [2013]. Path-space manipulation of physically-based light transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4):129:1–129:11.

Segovia, B., Iehl, J. C., Mitanchey, R., and Péroche, B. [2006a]. Bidirectional instant radiosity. In *Proc. of Eurographics Symposium on Rendering*, pp. 389–398.

Segovia, B., Iehl, J. C., Mitanchey, R., and Péroche, B. [2006b]. Non-interleaved deferred shading of interleaved sample patterns. In *Proc. of the 21st Symposium on Graphics Hardware*, pp. 53–60. ACM, New York, NY, USA.

Segovia, B., Iehl, J. C., and Péroche, B. [2007]. Metropolis instant radiosity. *Computer Graphics Forum*, 26(3):425–434.

Shirley, P. [1990]. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proc. of Graphics Interface '90*, pp. 205–212. Canadian Information Processing Society, Toronto, Ont., Canada.

Shirley, P., Wade, B., Hubbard, P. M., Zareski, D., Walter, B., and Greenberg, D. P. [1995]. Global illumination via density-estimation. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 219–230. Springer-Verlag.

Shirman, L. A. and Abi-Ezzi, S. S. [1993]. The cone of normals technique for fast processing of curved patches. *Computer Graphics Forum (Proc. of Eurographics '93)*, 12(3):261–272.

Sillion, F. X., Arvo, J. R., Westin, S. H., and Greenberg, D. P. [1991]. A global illumination solution for general reflectance distributions. *Computer Graphics (Proc. of SIGGRAPH)*, 25(4):187–196.

Smits, B., Arvo, J., and Greenberg, D. [1994]. A clustering algorithm for radiosity in complex environments. In *Proc. of SIGGRAPH 94*, Annual Conference Series, pp. 435–442. ACM, New York, NY, USA.

Smits, B. E., Arvo, J. R., and Salesin, D. H. [1992]. An importance-driven radiosity algorithm. *Computer Graphics (Proc. of SIGGRAPH)*, 26(2):273–282.

Spanier, J. [1966]. Two pairs of families of estimators for transport problems. *SIAM Journal on Applied Mathematics*, 14(4):702–713.

Spanier, J. and Gelbard, E. M. [1969]. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley series in computer science and information processing. Addison-Wesley Pub. Co.

Stam, J. [1995]. Multiple scattering as a diffusion process. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 41–50. Springer-Verlag.

Steinberg, H. A. and Kalos, M. H. [1971]. Bounded estimators for flux at a point in Monte Carlo. *Nuclear Science and Engineering*, 44:406–412.

Stich, M., Friedrich, H., and Dietrich, A. [2009]. Spatial splits in bounding volume hierarchies. In *Proc. of High Performance Graphics*, pp. 7–13. ACM, New York, NY, USA.

Sun, B., Ramamoorthi, R., Narasimhan, S. G., and Nayar, S. K. [2005]. A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1040–1049.

Sun, X., Zhou, K., Guo, J., Xie, G., Pan, J., Wang, W., and Guo, B. [2013]. Line segment sampling with blue-noise properties. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4):127:1–127:14.

Sun, X., Zhou, K., Lin, S., and Guo, B. [2010]. Line space gathering for single scattering in large scenes. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 29(4):54:1–54:8.

Szirmay-Kalos, L., Tóth, B., and Magdics, M. [2011]. Free path sampling in high resolution inhomogeneous participating media. *Computer Graphics Forum*, 30(1):85–97.

Szirmay-Kalos, L. and Umenhoffer, T. [2008]. Displacement mapping on the GPU - state of the art. *Computer Graphics Forum*, 27(6):1567–1592.

Tipler, P. A. and Mosca, G. [2004]. *Physics for Scientists and Engineers*. W. H. Freeman, New York, 5th edn.

Tzeng, S., Patney, A., Davidson, A., Ebeida, M. S., Mitchell, S. A., and Owens, J. D. [2012]. High-quality parallel depth-of-field using line samples. In *Proc. of High Performance Graphics*, pp. 23–31. Eurographics Association, Aire-la-Ville, Switzerland.

Ulbrich, J., Novák, J., Rehfeld, H., and Dachsbacher, C. [2013]. Progressive visibility caching for fast indirect illumination. In *Proc. of Vision, Modeling and Visualization*, pp. 203–210. Eurographics Association.

Veach, E. [1996]. Non-symmetric scattering in light transport algorithms. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 81–90. Springer-Verlag, London, UK.

Veach, E. [1997]. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. thesis, Stanford University, Stanford, CA, USA.

Veach, E. and Guibas, L. J. [1994]. Bidirectional estimators for light transport. In *Proc. of Eurographics Rendering Workshop 1994*, pp. 147–162.

Veach, E. and Guibas, L. J. [1997]. Metropolis light transport. In *Proc. of SIGGRAPH 97*, Annual Conference Series, pp. 65–76. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

von Helmholtz, H. and Southall, J. [1924]. *Helmholtz's Treatise on Physiological Optics*. The Optical Society of America.

Wald, I., Kollig, T., Benthin, C., Keller, A., and Slusallek, P. [2002]. Interactive global illumination using fast ray tracing. In *Proc. of Eurographics Workshop on Rendering Techniques*, pp. 15–24. Eurographics Association, Aire-la-Ville, Switzerland.

Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., Parker, S. G., and Shirley, P. [2007]. State of the art in ray tracing animated scenes. In *STAR Proceedings of Eurographics 2007*, pp. 89–116. The Eurographics Association.

Wallace, J. R., Cohen, M. F., and Greenberg, D. P. [1987]. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (Proc. of SIGGRAPH)*, 21(4):311–320.

Walter, B., Arbree, A., Bala, K., and Greenberg, D. P. [2006]. Multidimensional lightcuts. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 25(3):1081–1088.

Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. [2005]. Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1098–1107.

Walter, B., Khungurn, P., and Bala, K. [2012]. Bidirectional lightcuts. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):59:1–59:11.

Ward, G. J., Rubinstein, F. M., and Clear, R. D. [1988]. A ray tracing solution for diffuse interreflection. *Computer Graphics (Proc. of SIGGRAPH)*, 22(4):85–92.

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

Williams, L. [1978]. Casting curved shadows on curved surfaces. *Computer Graphics (Proc. of SIGGRAPH)*, 12(3):270–274.

Woodcock, E., Murphy, T., Hemmings, P., and T.C., L. [1965]. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Computing Methods to Reactor Problems*. Argonne National Laboratory.

Wyman, C. and Ramsey, S. [2008]. Interactive volumetric shadows in participating media with single scattering. In *Proc. of IEEE Symposium on Interactive Ray Tracing*, pp. 87–92.

Yao, C., Wang, B., Chan, B., Yong, J., and Paul, J.-C. [2010]. Multi-image based photon tracing for interactive global illumination of dynamic scenes. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 29(4):1315–1324.

Yu, I., Cox, A., Kim, M. H., Ritschel, T., Grosch, T., Dachsbacher, C., and Kautz, J. [2009]. Perceptual influence of approximate visibility in indirect illumination. *ACM Transactions on Applied Perception*, 6(4):24:1–24:14.

Yue, Y., Iwasaki, K., Chen, B.-Y., Dobashi, Y., and Nishita, T. [2010]. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 29(6):177:1–177:8.

Zhou, K., Hou, Q., Wang, R., and Guo, B. [2008]. Real-time kD-tree construction on graphics hardware. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 27(5):126:1–126:11.

Index