Perceptually-motivated, Interactive Rendering and Editing of Global Illumination


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

Vorgelegt durch

Tobias Ritschel
Max-Planck-Institut Informatik
Campus E1 4
66123 Saarbrücken
Germany

am 29. Okt 2009 in Saarbrücken

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

Gutachter – Reviewer
Prof. Dr. Hans-Peter Seidel, MPI Informatik, Saarbrücken, Germany
Prof. Dr. Jan Kautz, University College London, UK
Prof. Dr. Carsten Dachsbacher, VISUS / Universität Stuttgart, Germany
Dr. Habil. Karol Myszkowski, MPI Informatik, Saarbrücken, Germany

Dekan – Dean
Prof. Dr. Joachim Weickert, Universität des Saarlandes, Saarbrücken, Germany

Kolloquium – Examination

Datum – Date:
22. Dez. 2009

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

Prüfer – Examiners:
Prof. Dr. Hans-Peter Seidel, MPI Informatik, Saarbrücken, Germany
Prof. Dr. Jan Kautz, University College London, UK
Prof. Dr. Carsten Dachsbacher, VISUS / Universität Stuttgart, Germany
Dr. Habil. Karol Myszkowski, MPI Informatik, Saarbrücken, Germany

Protokoll – Reporter:
Dr. Thorsten Thormählen, MPI Informatik, Saarbrücken, Germany

Abstract

This thesis proposes several new perceptually-motivated techniques to synthesize, edit and enhance depiction of three-dimensional virtual scenes. Finding algorithms that fit the perceptually economic middle ground between artistic depiction and full physical simulation is the challenge taken in this work. First, we will present three interactive global illumination rendering approaches that are inspired by perception to efficiently depict important light transport. Those methods have in common to compute global illumination in large and fully dynamic scenes allowing for light, geometry, and material changes at interactive or real-time rates. Further, this thesis proposes a tool to edit reflections, that allows to bend physical laws to match artistic goals by exploiting perception. Finally, this work contributes a post-processing operator that depicts high contrast scenes in the same way as artists do, by simulating it "seen" through a dynamic virtual human eye in real-time.

Kurzzusammenfassung

Diese Arbeit stellt eine Anzahl von Algorithmen zur Synthese, Bearbeitung und verbesserten Darstellung von virtuellen drei-dimensionalen Szenen vor. Die Herausforderung liegt dabei in der Suche nach Ausgewogenheit zwischen korrekter physikalischer Berechnung und der künstlerischen, durch die Gesetze der menschlichen Wahrnehmung motivierten Praxis. Zunächst werden drei Verfahren zur Bild-Synthese mit globaler Beleuchtung vorgestellt, deren Gemeinsamkeit in der effizienten Handhabung großer und dynamischer virtueller Szenen liegt, in denen sich Geometrie, Materialen und Licht frei verändern lassen. Darauffolgend wird ein Werkzeug zum Editieren von Reflektionen in virtuellen Szenen das die menschliche Wahrnehmung ausnutzt um künstlerische Vorgaben umzusetzen, vorgestellt. Die Arbeit schließt mit einem Filter am Ende der Verarbeitungskette, der den wahrgenommen Kontrast in einem Bild erhöht, indem er die Entstehung von Glanzeffekten im menschlichen Auge nachbildet.

Summary

This thesis proposes several new perceptually-motivated techniques to synthesize, edit and enhance depiction of three-dimensional virtual scenes. Finding algorithms that fit the perceptually economic middle ground between artistic depiction and full physical simulation is the challenge taken in this work. First, we will present three interactive global illumination rendering approaches that are inspired by perception to efficiently depict important light transport. Those methods have in common to compute global illumination in large and fully dynamic scenes allowing for light, geometry, and material changes at interactive or real-time rates. Further, this thesis proposes a tool to edit reflections, that allows to bend physical laws to match artistic goals by exploiting perception. Finally, this work contributes a post-processing operator that depicts high contrast scenes in the same way as artists do by simulating it "seen" through a dynamic virtual human eye in real-time. This work starts with an introduction in Chapter 1, that motivates the subject, lists the contributions made and gives and outline of the thesis. Next a background is provided in Chapter 2. Chapter 3 reviews previous work, which is relevant for Chapter 4–8, where five novel techniques are presented in detail. The thesis is completed by a conclusion in Chapter 9 which also contains a discussion of future work.

Image-space Directional Occlusion Physically plausible illumination at real-time framerates is often achieved using approximations. One popular example is ambient occlusion (AO), for which very simple and efficient implementations are used extensively in production. Recent methods approximate AO between nearby geometry in screen space (SSAO). The key observation described in Chapter 4 is, that screen-space occlusion methods can be used to compute many more types of effects than just occlusion, such as directional shadows and indirect color bleeding. The proposed generalization has only a small overhead compared to classic SSAO, approximates direct and one-bounce light transport in screen space, can be combined with other methods that simulate transport for macro structures and is visually equivalent to SSAO in the worst case without introducing new artifacts. Since the method works in screen space, it does not depend on the

geometric complexity. Plausible directional occlusion and indirect lighting effects can be displayed for large and fully dynamic scenes at real-time frame rates.

Imperfect Shadow Maps Chapter 5 presents a method for interactive computation of indirect illumination in large and fully dynamic scenes based on approximate visibility queries. While the high-frequency nature of direct lighting requires accurate visibility, indirect illumination mostly consists of smooth gradations, which tend to mask errors due to incorrect visibility. The approach exploits this by approximating visibility for indirect illumination with imperfect shadow maps – low-resolution shadow maps rendered from a crude point-based representation of the scene. These are used in conjunction with a global illumination algorithm based on virtual point lights enabling indirect illumination of dynamic scenes at real-time frame rates. Finally, it is demonstrated that imperfect shadow maps are a valid approximation to visibility, which makes the simulation of global illumination an order of magnitude faster than using accurate visibility.

Micro-Rendering Recent approaches to global illumination for dynamic scenes achieve interactive frame rates by using coarse approximations to geometry, lighting, or both, which limits scene complexity and rendering quality. High-quality global illumination renderings of complex scenes are still limited to methods based on ray tracing. While conceptually simple, these techniques are computationally expensive. Chapter 6 presents an efficient and scalable method to compute global illumination solutions at interactive rates for complex and dynamic scenes. The method is based on parallel final gathering running entirely on the GPU. At each final gathering location micro-rendering is performed: the algorithm traverses and rasterizes a hierarchical point-based scene representation into an importance-warped micro-buffer, which allows for BRDF importance sampling. The final reflected radiance is computed at each gathering location using the micro-buffers and is then stored in image-space. The system can trade quality for speed by reducing the sampling rate of the gathering locations in conjunction with bilateral upsampling. The chapter demonstrates the applicability of the method to interactive global illumination, the simulation of multiple indirect bounces, and to final gathering from photon maps.

Interactive Reflection Editing Effective digital content creation tools must be both efficient in the interactions they provide but also allow full user control. There may be occasions, when art direction requires changes that contradict physical laws. In particular, it is known that physical correctness of reflections for the human observer is hard to assess. For many centuries, traditional artists have exploited

this fact to depict reflections that lie outside physical laws. However, a system that gives explicit control of this effect to digital artists has not yet been described. Chapter 7 introduces a system that transforms physically correct reflections into art-directed reflections, as specified by *reflection constraints*. The system introduces a taxonomy of reflection editing operations, using an intuitive user interface, that works directly on the reflecting surfaces with real-time visual feedback using a GPU. A user study shows how such a system can allow users to quickly manipulate reflections according to an art direction task.

Temporal Glare Glare is a consequence of light scattered within the human eye when looking at bright light sources. This effect can be exploited for tone mapping since adding glare to the depiction of high-dynamic range (HDR) imagery on a low-dynamic range (LDR) medium can dramatically increase perceived contrast. Even though most, if not all, subjects report perceiving glare as a bright pattern that fluctuates in time, up to now it has only been modeled as a static phenomenon. Chapter 8 argues, that the temporal properties of glare are a strong means to increase perceived brightness and to produce realistic and attractive renderings of bright light sources. Based on the anatomy of the human eye, a model is proposed, that enables real-time simulation of dynamic glare on a GPU. This allows an improved depiction of HDR images on LDR media for interactive applications like games, feature films, or even by adding movement to initially static HDR images. By conducting psychophysical studies, it is validated that the method improves perceived brightness and that dynamic glare-renderings are often perceived as more attractive depending on the chosen scene.

Zusammenfassung

Diese Arbeit stellt eine Anzahl von Algorithmen zur Synthese, Bearbeitung und verbesserten Darstellung von virtuellen drei-dimensionalen Szenen vor. Die Herausforderung liegt dabei in der Suche nach Ausgewogenheit zwischen korrekter physikalischer Berechnung und der künstlerischen, durch die Gesetze der menschlichen Wahrnehmung motivierten Praxis. Zunächst werden drei Verfahren zur Bild-Synthese mit globaler Beleuchtung vorgestellt, deren Gemeinsamkeit in der effizienten Handhabung großer und dynamischer virtueller Szenen liegt, in denen sich Geometrie, Materialen und Licht frei verändern lassen. Darauffolgend wird ein Werkzeug zum Editieren von Reflektionen in virtuellen Szenen das die menschliche Wahrnehmung ausnutzt um künstlerische Vorgaben umzusetzen, vorgestellt. Die Arbeit schließt mit einem Filter am Ende der Verarbeitungskette, der den wahrgenommen Kontrast in einem Bild erhöht, indem er die Entstehung von Glanzeffekten im menschlichen Auge nachbildet.

Die Arbeit beginnt mit einer Einführung in Kapitel 1, welche das Thema motiviert und die neuen Beiträge die diese Arbeit leistet beschreibt. Im Kapitel 2 wird der Hintergrund dieser Arbeit behandelt. Der Stand der Technik wird in Kapitel 3 dargestellt. Kapitel 4 bis Kapitel 8 stellen die vorgeschlagenen neuen Techniken im Detail dar. Diese Dissertation schließt mit einer Folgerung in Kapitel 9 die eine Anzahl weiterführende Ideen motiviert.

Screen-space Directional Occlusion Physikalisch plausible globale Beleuchtung in Echtzeit wird oft durch Näherungsverfahren ermöglicht. Ein populäres Beispiel einer solchen Technik ist “Ambient Occlusion” (AO), für das effiziente Verfahren verfügbar sind und das in der Praxis, z. B. in Computer-Spielen weite Verbreitung gefunden hat. Gängige Verfahren nähern AO mit Hilfe eines Teils der Geometrie im Bildraum an (“Screen-Space Ambient Occlusion” – SSAO). Die Idee die im Kapitel 4 entwickelt wird ist, daß SSAO dazu verwendet werden kann, wesentlich weitreichenderen Licht-Transport zu simulieren als bisher bekannt war: Gerichtete Schatten und farbige Inter-Reflektionen. Die beschriebene Verallgemeinerung von SSAO erfordert nur einen geringen Mehraufwand im Vergleich

zu klassischem SSAO und kann mit anderen Verfahren zur globalen Beleuchtung, auch aus dieser Arbeit, kombiniert werden. Da die Methode im Bildraum arbeitet, ist sie von der geometrischen Komplexität der Szene unabhängig. Ergebnis des Ansatzes sind – wenn auch angenähert - direktionale Schatten und farbige Interreflektionen im Bildraum in großen und dynamischen Szenen bei sehr hohen Bild-Wiederholraten.

Imperfect Shadow Maps Im Kapitel 5 wird eine Methode zur Berechnung von indirekter Beleuchtung in großen und dynamischen Szenen die auf "unvollständigen Sichtbarkeits-Tests" (Imperfect Shadow Maps - ISMs) basiert, vorgestellt. Während direkte Beleuchtung akkurate Sichtbarkeits-Test erfordert, erlauben die glatten Verläufe indirekter Beleuchtung die Verwendung angenäherter Sichtbarkeits-Tests, die durch direktes Licht visuell maskiert werden. Das Verfahren nutzt diesen Effekt indem es eine stark vereinfachte Szenen-Geometrie aus Punkten in eine Shadow Map zeichnet. Zusammen mit Algorithmen zur globalen Beleuchtung die auf virtuellen Punktlichtern bestehen, erlaubt der Ansatz die interaktive Darstellung großer dynamischer Szenen in einer Geschwindigkeit, die eine Größenordnung über der vormals bekannter Verfahren liegt.

Micro-Rendering Gängige Verfahren zur Darstellung globaler Beleuchtung verwenden oft starke Vereinfachungen der Beleuchtung, der Reflektanz oder der Geometrie, die aber in komplexen Szenen auftreten können. Die Darstellung solcher Szenen war bis jetzt Verfahren, die auf Ray-Tracing aufbauen, vorbehalten. Solche Verfahren aber sind zeitaufwendig und nicht explizit an das Problem globaler Beleuchtung angepasst. Die in Kapitel 6 vorgestellte Methode ist eine effektive und skalierbare Alternative um mit der globalen Beleuchtung solcher komplexen Szenen umzugehen. Der Ansatz basiert auf einer GPU-Implementierung von "Final Gathering" und traversiert eine hierarchische Punktwolke und rasterisiert diese in einen BRDF-adaptiven Framebuffer, der an wichtigen Stellen höhere Abtastraten erlaubt. Das Kapitel demonstriert die Anwendbarkeit der Methode anhand einer Anzahl von Beispielen, die von interaktiver globaler Beleuchtung, bis zum Final Gathering für Photon Maps reichen.

Interactive Reflection Editing In der compuer-basierten Gestaltung virtueller Szenen kommt es vor, das künstlerische Vorgaben im Wiederspruch zu physikalischen Gesetzten stehen. Im speziellen ist es bekannt, da§ die menschliche Wahrnehmung große künstlerische Freiheit bei der Darstellung von Spiegelungen zulässt, was von viele Künstlern seit Jahrhunderten genutzt wird. Diese Möglichkeit bieten heutige Editiermöglichkeiten für drei-dimensionale Szenen bis jetzt nicht.

In Kapitel 7 wird ein System vorgestellt, das physikalisch korrekte Spiegelungen in solche umsetzte die künstlerischen Vorgaben folgen. Das Kapitel schlägt ein interaktives System zur Manipulation von Randbedingungen vor, welche Reflektionen erfüllen müssen. Die GPU-basierende Implementierung erlaubt es, bestehende Spiegelungen mit direktem Feedback zu manipulieren. Eine Benutzerstudie zeigt, da§ die Interaktionsmetapher schnell zu erlernen ist und erlaubt, Reflektionen un-physikalisch zu bearbeiten obwohl diese physikalisch exakten Spiegelungen gleichen.

Temporal Glare Fällt ein Lichtstahl ins menschliche Auge, wird dieser nicht auf einen einzelnen Netzhaut-Punkt sondern auf eine komplexe Form (engl. “Glare”) abgebildet. Während Glare im Alltag unmerklich bleibt, kann er bei starkem Helligkeitskontrast, wie einer Lichtquelle bei Nacht, sichtbar werden. Dieser Effekt kann verwendet werden um empfundenen Kontrast in ein Bild einzubringen. Unsere Echtzeit-Simulation des Licht-Transports im menschlichen Augen erlaubt erstmals die Darstellung der zeitlichen Veränderung des Glare (“Temporal Glare”), hervorgerufen durch Bewegungen des Betrachters, Sakkaden und die Akkomodation der Linse. Das Kapitel schließt mit einer psycho-physikalischen Studie, die validiert, dass Temporal Glare den empfunden Kontrast erhöht und von vielen Beobachtern als attraktiver verglichen zu statischem Glare empfunden wird.

Contents

1 Introduction

This thesis proposes several new perceptually-motivated techniques to synthesize, edit and enhance the depiction of three-dimensional virtual scenes. In this first chapter we motivate our research, present our main contributions and outline the whole thesis.

1.1 Motivation

This work is motivated by physical laws, findings in perception as well as it is founded in anecdotal artistic practice. We are inspired by the idea, that art has developed over centuries the most efficient means to render information as perceived by humans. Also our cultural background has trained us to see images in a way that adapted to such means. When designing algorithms, one can learn from artists, even when aiming for naturalistic depiction. A current trend in visual computing is non-photorealistic rendering (NPR), that aims for non-naturalistic depiction. While this is efficient in conveying information in a compressed way, we will argue, how similar approaches can also lead to improved naturalistic depiction. The similarity between an artist and a machine that produce images is, that both will only have a limited amount of time and resources to convey the message, and physical bounds of human perception define what is important and worth to invest resources into. For examples, artists will carefully match the directions of shadows from a local light in a different way as they do for distant lights, but for indirect shadows they will depict them as diffuse darkening. Completely ignoring this darkening altogether is not an option, neither it is to physically simulate it in full detail. Finding algorithms that fit the artistically and perceptually economic middle ground, is the challenge taken in this work. First, we will present three

interactive global illumination rendering approaches that are inspired by perception to efficiently depict important light transport in the way it is perceived. Further, we propose a tool, that allows to adapt physical laws to artistic goals and describe a post-processing effects that reproduces high contrast as it is perceived in human eyes and depicted by artists.

1.2 Contributions

This section lists individual contributions made in those five publications, on which this thesis is based [Ritschel et. al. 2008b; Ritschel et. al. 2009; Ritschel et. al. 2009a; Ritschel et. al. 2009c; Ritschel et. al. 2009b].

The two contributions to interactive global illumination in Chapter 4, (published as [Ritschel et. al. 2009]) are:

- An approach to include directional blocker information in image space occlusion
- An extension that bounces light from image space blockers

While incurring only a performance overhead of a few percent compared to other image space methods, the contributions made substantially improve realism in rendered scenes, e. g. the perception of light and materials under natural illumination.

The main contributions of Chapter 5 (published as [Ritschel et. al. 2008b]) are:

- An approximate representation of visibility — Imperfect Shadow Maps — (ISMs) to facilitate the computation of indirect illumination.
- An instant-radiosity based technique that uses ISMs to compute (multi-bounce) indirect illumination in large, dynamic scenes in real-time on modern GPUs.
- An analysis of the influence of ISMs on the resulting indirect illumination.
- Imperfect Reflective Shadow Maps, that generalize reflective shadow maps [Dachsbacher and Stamminger 2005] to multiple bounces.

The contributions made advance the state of the art by removing restrictions of previous work limited to static scenes, or outperforming previous methods that handle dynamic scenes by one order of magnitude.

Main contributions in Chapter 6 (published as [Ritschel et. al. 2009a]) are:

- A novel, scalable GPU-based micro-rendering technique to efficiently gather incident radiance in large and dynamic scenes.
- A method to perform BRDF-based importance warping for rasterization of a point-based hierarchical scene representation.
- Techniques for the efficient computation of multiple-bounce indirect illumination and photon mapping walkthroughs.
- A point-based scene representation optimized for GPU cache efficiency and data locality.

Micro-Rendering advances the accuracy of interactive global illumination while supporting larger scenes and more materials compared to previous work. The resulting quality level has previously only been found in (offline, non-real-time) movie production.

Specific contributions of Chapter 7 (published as [Ritschel et. al. 2009c]) are as follows:

- A formalization of artist-directed edits to reflections.
- A user interface to manipulate reflection edits.
- A real-time GPU implementation to propagate reflection edits over three-dimensional surfaces.
- A study of preference and task performance when using the system.

Compared to previous appearance editing approaches, the novel perspective of this work is, to permit new artistic degrees of freedom by exploiting perception to allow physically impossible edits that still achieve perceptually plausible results.

The contributions of Chapter 8 (published as [Ritschel et. al. 2009b]) are:

- A model for light scattering in the human eye based on wave-optics, that includes temporal characteristics of major anatomical structures.
- A GPU implementation of this model, that allows to see an HDR image “through” a simulated dynamic eye in real-time.
- A psychophysical study that measures the perceived brightness and preference for static and temporal glare models.

State of the art glare models, even outside computer graphics, were either substantially more ad-hoc or limited to static eyes and non-real-time performance.

[Ritschel et. al. 2007] and [Ritschel 2007] cover interactive global illumination but both are largely superseded by Chapter 4, Chapter 5 and Chapter 6. A full list of related own publications is found on Page I of the Appendix.

1.3 Outline

This thesis is structured as follows. After this introduction, a background on perception and rendering is given in Chapter 2 before we review previous work in Chapter 3. From Chapter 4 to Chapter 8, five novel techniques are presented in detail. We propose three new approaches to interactive global illumination, namely “Image Space Directional Occlusion” in Chapter 4, “Imperfect Shadow Maps” in Chapter 5 and “Micro-Rendering” in Chapter 6. Further, Chapter 7 introduces a novel interaction metaphor to edit reflection in computer generated scenes. A post processing technique called “Temporal Glare”, which improves the depiction of contrast is presented in Chapter 8. This order also presents the way the techniques would be included in an existing graphics pipeline, ranging from image synthesis, to editing and post-processing. The thesis is completed by a conclusion in Chapter 9 which also contains a discussion of future work.

2

Background

In this chapter we will introduce some conventions and basic phenomena that provide a background used in depiction and manipulation of global illumination with a perceptual motivation. Starting in physics (Section 2.1) and algebra (Section 2.2) a background for rendering (Section 2.3) and perception Section 2.4 is provided.

2.1 Physics

2.1.1 Radiometry

Radiometric units are used to describe electromagnetic waves. As physically-based rendering simulates light which is a visible electromagnetic wave, we review the relevant radiometric units here and use "light" as a synonym for visibile electromagnetic waves.

Radiant Energy Radiant energy describes the energy of light (Symbol Q). Its SI unit is the *Joule* (J). In rendering, one is mostly considered with resolving light at a single point in time, i. e. per unit time. Differentiating radiant energy in time will lead to radiant flux.

Radiant energy $Q(\lambda)$, as all other radiometric quantities depends on wavelength λ (i. e. color) but we drop this dependency until Section 2.1.2.

Radiant Flux Radiant flux is radiant energy per unit time (Symbol Φ). Its unit is *Watt* (W), $1\,\mathrm{W} = 1\,\mathrm{J}/\mathrm{s}$. Integrating radiant flux over time leads back to radiant

energy. For image formation, one is often considered with resolving light per unit time and unit area. Differentiating radiant flux in area will lead to irradiance, while differentiating in solid angle will lead to intensity.

Irradiance Irradiance is radiant flux per unit area (Symbol E). Its unit is Watt per square meter ($\mathrm{W/m^2}$). Integrating irradiance over area leads back to radiant flux. For image formation, one is often considered with resolving light per unit time and unit area in a unit solid angle. Differentiating irradiance in direction will lead to radiance.

Radiance Radiance is irradiance per unit solid angle (Symbol L). Its unit is Watt per square meter per steradian ($\mathrm{W/m^2\,sr}$). Integrating radiance over all directions leads back to irradiance. Radiance is often a useful unit for rendering and also closest to the common conception of "light", both in everyday life and naïve, non-physically-based rendering.

Intensity Intensity is radiant flux per unit solid angle (Symbol I). Its unit is Watt per solid angle ($\mathrm{W/sr}$). At the same time intensity is radiance integrated over area, which is used for point lights, as it allows to abstract away the area. Intensity distribution functions describe the intensity as a function of direction. In physically-based rendering and light engineering such functions are sometimes called a "*luminaire*".

This thesis ignores some physical phenomena that are not encountered for most applications of computer graphics. First, we only consider an infinitesimal period of time. We model light to spread instantaneous and do not consider the time it takes to propagate in reality. We base our work on geometrical opitcs without diffraction and interference, with the exception of Chapter 8. We assume that no energy-transfer between bands (fluoresecence) takes place, no participating media is present and do not consider polarization. It assumes that all objects in the scene are big compared to the wavelength of light, which allows to exclude diffraction.

To some extent sound waves also fit the framework of electromagnetic waves and computer graphics techniques have been successfully applied to sound simulation [Funkhouser, Jot and Tsingos 2002]. However, the propagation time of sound is significant, while it was neglected for rendering light. Further, effects such as diffraction have to be simulated because occluder sizes are significant compared to the wavelength of sound. This argument of scale applies in the opposite direction to rendering of microscopic structures as found inside the dynamic human eye (Chapter 8): for sufficient "zooming" light behaves similar to sound and becomes

subject to e. g., diffraction.

2.1.2 Photometry

The human visual system (HVS) (Section 2.4) is sensitive to a limited wavelength range of electromagnetic waves. Some wavelengths are perceived more accurate than others. The range of visible light begins at ca. 400 nm (blue), ends at ca. 700 nm (red) with a peek around 555 nm (green) [Palmer 1999].

Luminosity function More precisely, the sensitivity of the HVS to a certain wavelength can be described using a luminosity function (Symbol $\bar{y}$). The luminosity function maps wavelength λ to sensitivity. The function varies between individuals and between day and night vision. In practice an average model such as CIE 1951 [Palmer 1999] is used. To convert a radiometric into a photometric quantity relative to a certain luminosity function, the spectrum of the radiometric quantity is convolved with the luminosity function. For example, radiant flux Φ (radiometric) can be converted into luminous flux F (photometric) as

$$F = 683\,{}^{\mathrm{lm}}/_{\mathrm{W}} \cdot \int \bar{y}(\lambda)\Phi(\lambda)\mathrm{d}\lambda.$$

This leads to the following photometric units.

Luminous Energy Perceived energy of light (Symbol Q_{v}). Its unit is Lumen seconds (lm s).

Luminous Flux Describes perceived light power (Symbol F). Its SI unit is Lumen (lm).

Illuminance Perceived light power per unit area (Symbol E_{v}). Its SI unit is the *Lux* (lx) ($1\,\mathrm{lx} = 1\,{}^{\mathrm{lm}}/_{\mathrm{m}^2}$).

Luminous Intensity Perceived light power per unit solid angle (Symbol I_{v}). Its SI unit is the *Candela* ($1\,\mathrm{cd} = 1\,{}^{\mathrm{lm}}/_{\mathrm{sr}}$).

Luminance Perceived light power per unit solid angle per unit area (Symbol L_{v}). Its unit is candela per square meter (${}^{\mathrm{cd}}/_{\mathrm{m}^2}$).

2.1.3 Reflectance

Reflectance at location $\mathbf{x}$ is the ratio between outgoing (Φ_{out}) and incoming (Φ_{in}) flux

$$\rho(\mathbf{x}) = \frac{\mathrm{d}\Phi_{\text{out}}}{\mathrm{d}\Phi_{\text{in}}}.$$

It is a two-dimensional function, that varies over the surface. As we did for radiometric qualities, we dropped the wavelength-dependency here, although it is strong and responsible for many colors we perceive.

Often, the reflectance of a surface varies over directions, as modeled by the *bidirectional reflectance distribution function* (BRDF):

$$f_{\text{r}}(\mathbf{x}, \omega_{\text{in}} \to \omega_{\text{out}}) = \frac{L(\mathbf{x}, \omega_{\text{out}})}{L(\mathbf{x}, \omega_{\text{in}})\mathbf{n} \cdot \mathrm{d}\theta},$$

where L is scene radiance, $\mathbf{x}$ a differential patch with orientation $\mathbf{n}$, and ω_{in} resp. ω_{out} are the in- and out-going directions. The BRDF is a six-dimensional function, with four directional dimensions (two incoming and two outgoing) and two spatial dimensions over a surface. The range of f_{r} is $\mathbb{R}_0^+$, but it holds for all physically correct BRDFs, that outgoing flux is less then incoming flux,

$$\int_{\Omega^+} f_{\text{r}}(\mathbf{x}, \omega_{\text{in}} \to \omega_{\text{out}}) \mathrm{d}\omega_{\text{in}} \leq 1.$$

Further, all physically plausible BRDFs are *reciprocal*, that is, in- and out-direction can be exchanged:

$$f_{\text{r}}(\mathbf{x}, \omega_{\text{in}} \to \omega_{\text{out}}) = f_{\text{r}}(\mathbf{x}, \omega_{\text{out}} \to \omega_{\text{in}}).$$

There are several classes of BRDFs. Often the spatial variation is dropped and the BRDF is assumed to be constant (*shift-invariant*) over the surface, making it a 4D function. Orthogonally, some materials allow to drop dependence on one incoming directional dimension, leading to a 3D function. Such *isotropic* BRDFs are re-parametrized to only depend on the angle between ω_{in} and $\mathbf{n}$. A very simple form of BRDF is found on *Lambertian diffuse* surfaces. Here, no directional dependence is found, leading to a constant. The other extreme is a perfect *mirror*, that is zero almost everywhere, and only a Dirac for those ω_{out} that are a mirror of ω_{in} over the normal $\mathbf{n}$ at $\mathbf{x}$. All BRDFs between diffuse and mirror are called *glossy*, and a plethora of other names for in-between phenomena [Hunter and Harold 1987].

There is one simplifying assumption made when using a BRDF: light that leaves the surface at a location $\mathbf{x}$ enters the surface also at $\mathbf{x}$. However, there are some materials such as skin or wax where a fraction of light leaves at a position $\mathbf{x}_{\text{out}}$

different from $\mathbf{x}$. The more general *bidirectional subsurface scattering function* (BSSRDF) [Jensen et al. 2001], describes, how light that enters a surface at an arbitrary location $\mathbf{x}_{\text{in}}$ from an arbitary direction ω_{in} leaves this surface at another location $\mathbf{x}_{\text{out}}$ in a direction ω_{out}, as in $f_{\text{ssr}}(\mathbf{x}_{\text{in}}, \mathbf{x}_{\text{out}}, \omega_{\text{in}} \rightarrow \omega_{\text{out}})$. We do not consider subsurface light transport in this work.

2.2 Algebra

In this section we give a very basic introduction of linear operators, followed by discussion of some basis functions used in rendering.

2.2.1 Linear Operators

Linear operators are a generalization from a finite-dimensional vector space with linear transformations to infinite-dimensional vector spaces. Elements of such a space are functions and classic linear transformations based on matrices are replaced by operators and kernels. In simple words, an operator is a function that turns a function into another function. Let X and Y be spaces of functions, and a function $f : T \rightarrow T'$ be a mapping from T to T' which is a vector in X. An operator $\mathbf{T} : X \rightarrow Y$ is a mapping from elements f in X to Y, defined as

$$(\mathbf{T}f)(s) = \int_T f(t)k(s,t)\mathrm{d}t,$$

where $k : T \times T \rightarrow T'$ is called a *kernel*. Light transport can be understood in terms of such operators [Arvo, Torrance and Smits 1994]. For example the light in a scene is a function from a manifold surface and a directional domain to radiance, i. e. an element of a vector space and light transport is an operator called the *transport operator*. We will make use of such operators in Section 2.3.1.

2.2.2 Basis Functions

Rendering operates on several signals which are continuous functions in high dimensions. One such signal is radiance, a two-dimensional function of direction and a one-dimensional function of wavelength. Another signal, is the four-dimensional BRDF. To handle such signals computational, several techniques for their digital representation are available. Formally, they are all projections from the continuous space of continuous functions onto some finite basis. Popular bases are the common piecewise constant functions as used in screen pixels (in the extreme

case a single constant), wavelets, spherical harmonics, or point-based irregular representations (cf. Figure 2.1).

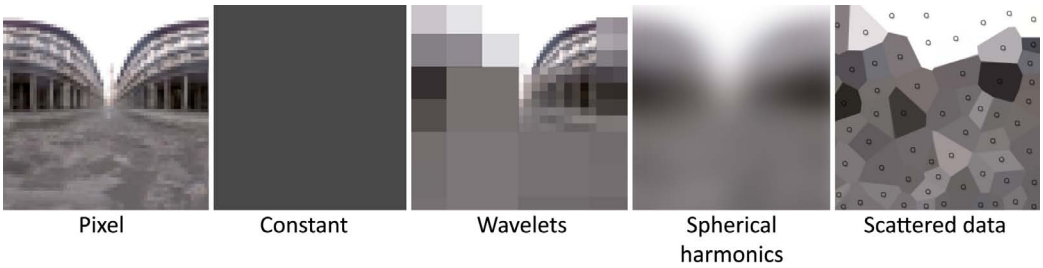

Figure 2.1: Different ways to represent a signal (here incoming light) using basis functions.

Pixels Traditionally, elements of digital images are called "pixels". Such a scheme corresponds to a basis of translated unit-box functions. We will call this basis simply the *pixel* basis. In the context of texture (reflectance signals) or volumes (geometry), pixels are called texels or voxels. One drawback is, that because of their simplicity, they store much redundant information, if the signal is smooth. This wastes space, and also makes their processing slow, i. e. convolving a BRDF slice and incoming light with N pixels each is $O(N^2)$. The simple, regular pixel structure fits well to parallel processors and reading and writing pixels is very efficient on such hardware.

Constants In the extreme case, the entire function is flattened into a constant. In terms of operators, this considers the function to have a constant value and moves it outside of the integral. If this is appropriate, depends on the function, but can be very useful as a convolution by a constant is a plain multiplication and very fast.

Wavelets Wavelets store the signals using a different, more complex basis. Without going into too much details here (they are not used in this work, but in concurrent approaches they are), this basis is *hierarchical* [Stollnitz, Rose and Salesin 1995]. That is, some basis vectors have a large support, and average large areas, while others have a small support, but only a difference relative to a wider basis is stored. This can be seen as a n-ary *wavelet coefficient* tree where the average is stored in the root and children store differences. This allows several optimizations. When thresholding the wavelet coefficients, only less relevant details will get lost. Often more than ninety percent of the coefficients can be skipped. This makes wavelets a good choice for compression. Also convolution of two signals of size N in wavelet space is faster ($O(N)$), resembling a sparse matrix

multiplication, while the usual basis corresponds to a full matrix multiplication. Even convolution of more than two factors, like triple-products is still efficient ($O(N)$) [Ng et. al. 2004 ACM Trans. on Graphics (Proc. ACM SIGGRAPH)]. One shortcoming of wavelets is, that they are difficult to rotate [Wang et. al. 2006]. In practice, rendering signals are represented using several hundreds of coefficients.

Spherical Harmonics Another basis is the spherical harmonics (SH) basis, a family of smooth functions. In rendering they are used to represent smooth spherical, two-dimensional functions. One SH space is not hierarchical, that is, the required frequency detail must be decided beforehand. However, there are different SH spaces that are able to represent increasingly more detail but also using increasingly more coefficients. The basis' smoothness allows SH to excellently support smooth functions, but fails to adapt to sharp details. Computing the convolution of two functions given as SHs is a simple dot product of the SH coefficients. Working with SH rotations is easier than wavelets, but still involved [Sloan et. al. 2002]. In practice, rendering signals are often represented using 25 coefficients only.

Scattered Data In a scattered data representation, a function in $\mathbb{R}^n \to \mathbb{R}^m$ is stored as a list of points with a location in $\mathbb{R}^n$ and a value in $\mathbb{R}^m$. As other representation cover the full domain $\mathbb{R}^n$ the domain information becomes implicit: The first pixel is at the first location, the second at the second, and so forth. Scattered data representations can be very useful, because the point density can freely adapt to features where required. Evaluating the function value in-between points in the range $\mathbb{R}^n$ requires a *reconstruction* function. In the example in Fig. 2.1, a piecewise constant reconstruction was used. High-quality reconstruction can be computationally expensive, as it often requires to find multiple neighboring points in the range and to blend their domain value.

Scattered data is not to be mistaken with density estimation [Jensen 1996] as used in photon mapping. The density of the points is not the signal itself. Higher density only means that the signal is better reproduced.

2.3 Rendering

The synthesis of naturalistic images can also be considered as the simulation of light. Based on perception, computer graphics developed some practical means to efficiently simulate exactly such light transport phenomena that are important. For example, direct light is clearly more important than indirect light, and has

achieved much more attention. Also some special phenomena have special solution, i. e. planar mirrors. In this section, we review some models and computational techniques that are used in rendering and editing of physically-based illumination, such as the rendering equation (Section 2.3.1), shading models (Section 2.3.2), as well as some fundamental approaches to solve the rendering equation (Section 2.3.3 and Section 2.3.4). We also describe some basic image formation (Section 2.3.5) and tone reproduction (Section 2.3.6) assumptions and conventions.

2.3.1 Rendering Equation

The rendering equation (RE) by Kajiya [1986] states, that the radiance L leaving a differential surface patch at location $\mathbf{x}$ with normal $\mathbf{n}$ in direction ω_{out} is

$$L(\mathbf{x}, \omega_{\text{out}}) = E(\mathbf{x}, \omega_{\text{out}}) + \int_{\Omega^+} L(\mathbf{x}, \omega_{\text{in}}) f_{\text{r}}(\mathbf{x}, \omega_{\text{in}} \to \omega_{\text{out}}) \cos\theta \, \mathrm{d}\omega_{\text{in}}, \qquad (2.1)$$

where E is emitted radiance, f_{r} is the BRDF at location $\mathbf{x}$, Ω^+ the upper heimsphere above $\mathbf{x}$ and θ the angle between ω_{in} and $\mathbf{n}$.

The difficulty when trying to solve this equation is, that L appears on both sides. Two prominent approaches to solve the equation are presented later: Monte Carlo rendering in Section 2.3.3 and finite element methods in Section 2.3.4. When L is known on the right side, we call the solution of this equation *local* illumination, if it is unknown, we call it *global* illumination. In other words, evaluating the RE requires to evaluate the RE at many other points recursively.

Light reflection can be understood as a convolution of incoming light L_{in} with the BRDF f_{r} producing outgoing light L_{out} and we can rewrite this in operator form [Arvo, Torrance and Smits 1994] (cf. Section 2.2.1), using a *reflection operator* $\mathbf{K}$:

$$\begin{aligned} L_{\text{out}}(\mathbf{x}, \omega_{\text{out}}) &= \int_{\Omega^+} L_{\text{in}}(\mathbf{x}, \omega_{\text{in}}) f_{\text{r}}(\mathbf{x}, \omega_{\text{in}} \to \omega_{\text{out}}) \cos\theta \, \mathrm{d}\omega_{\text{in}} \\ &= (\mathbf{K} L_{\text{in}})(\mathbf{x}, \omega_{\text{out}}) \end{aligned}$$

or shorter

$$L_{\text{out}} = \mathbf{K} L_{\text{in}}. \qquad (2.2)$$

Next, we define a *geometry* operator $\mathbf{G}$:

$$(\mathbf{G}L)(\mathbf{x}, \omega) = L(x'(\mathbf{x}, \omega), \omega),$$

where $x'(\mathbf{x}, \omega)$ is the closest point from $\mathbf{x}$ in direction ω. This operator includes the visiblity and turns distant surface radiance into local incident radiance. This allows to rewrite the RE as

$$L = E + \mathbf{K}\mathbf{G}L$$

or finally, using a *transport operator* $\mathbf{T} = \mathbf{KG}$:

$$L = E + \mathbf{T}L.$$

Arvo, Torrance and Smits [1994] show, that such equations can indeed be solved using an infinite Neumann series where each summand represents one bounce:

$$L = E + \mathbf{T}E + \mathbf{T}^2E + \mathbf{T}^3E + \ldots.$$

A simplification often made in direct illumination, is the use of *point lights*, which turns the integral into a finite sum:

$$L(\mathbf{x}, \omega_{\text{out}}) = E(\mathbf{x}, \omega_{\text{out}}) + \sum_{i=1}^{N} I_i(\omega_i) f_{\text{r}}(\mathbf{x}, \omega_i \to \omega_{\text{out}}) \cos\theta_i, \tag{2.3}$$

where N is he number of point lights, ω_i is the direction of the i-th point light from $\mathbf{x}$, $I_i(\omega)$ is the directional intensity function for each point light and θ_i is the angle between the surface normal and ω_i.

If a function inside an operator can be written as a product or a sum, it can be useful to *split* this function. By doing so, each part can be compressed, e. g. flattened individually. For example ambient occlusion, splits L into light and visibility. Visibility is flattened into a constant, and light computed without visibility. Sometimes, after splitting, one factor or summand can be placed outside the integral, facilitating the solution. Another popular split is to separate direct and indirect illumination as a sum and simulate each one indepdentently [Stokes et al. 2004].

2.3.2 Shading Models

A BRDF is a six-dimensional function. For efficient and practical evaluation and storage of reflectance information, several so-called *shading models* were proposed.

One early model is the Phong [1975] model, that was later corrected to become energy-conserving by Lewis [1993]:

$$f_{\text{r}}(\omega_{\text{in}} \to \omega_{\text{out}}) = \frac{k_{\text{d}}}{\pi} + k_{\text{s}} \frac{n+2}{2\pi} \cos^n \alpha_{\text{Phong}},$$

where α_{Phong} is the angle between the incoming direction ω_{in} reflected around the normal and the outgoing direction ω_{out}. This model has three parameters: A specular exponent n together with a diffuse color k_{d} and a specular color k_{s}. Alternatively, the angle α_{Blinn} between the half-vector of ω_{in} and ω_{out} and the normal $\mathbf{n}$ can be used [Blinn and Newell 1976]. Note, that the same specular

exponent n will result if different highlight shapes when used in either Phong or Blinn-Phong shading.

Later, Lafortune et al. [1997] proposed a different popular model, which is sometimes used in this work. They use a generalization of the Phong model to N lobes, each with its own orientation $\mathbf{o}_i$, specular strength k_{si} and specular exponent n_i.

$$f_r(\omega_{in} \to \omega_{out}) = \frac{k_d}{\pi} + \sum_{i=0}^{N} k_{s_i}(\omega_{out} \cdot (\omega_{ix}\mathbf{o}_{ix}, \omega_{iy}\mathbf{o}_{iy}, \omega_{iz}\mathbf{o}_{iz},))^{n_i}.$$

This model can be used to approximate measured BRDFs with a few lobes already.

2.3.3 Monte Carlo

Monte Carlo integration is a method to numerically solve integrals, such as the RE (Section 2.3.1). Numerical techniques are required, when the integral is too costly to evaluate in analytic form, there is no analytic form available or it does not exist. Instead of the RE let us assume we want to solve the integral

$$\int_\Omega f(x)\mathrm{d}x.$$

For rendering the integrand f is the rendering integrand as described before in Section 2.3.1. Monte Carlo makes use of random numbers. The expected value $E[x]$ for a random variable $x \in \Omega$ with probability density function $p(x)$ is

$$E[x] = \int_\Omega xp(x)\mathrm{d}x.$$

The law of large numbers, states that

$$P(E[x] = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} x_i) = 1,$$

which means that averaging a large number of samples from f gives the expected value. We can use this to solve the integral. First, write the integrand $f = gp$, where p is a probability density function. It follows, that for random samples x_i

$$\int_\Omega f(x)\mathrm{d}x = \int_\Omega g(x)p(x)\mathrm{d}x = E[g(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \frac{g(x_i)}{p(x_i)}$$

is an estimator for the integral using n samples. Any distribution $p(x)$ can be used here, as long as $p(x)$ is non-zero for any x for which $f(x)$ is also nonzero. An estimator is said to have low *variance* if p closely matches f.

Monte Carlo is *unbiased*, which means, that with enough samples, it always converges to the exact result. A difference between the true result and an estimator

$$\beta = E[F] - F$$

is called *bias* of that estimator. It is introduced by systematical errors in the estimator, e. g by not sampling x, although $p(x)$ is non-zero.

Importance Sampling Monte Carlo allows to use *any* distribution $p(x)$. A distribution $p(x)$ that is similar in shape to $f(x)$ results in less variance, because samples that have a higher contribution to the integral have a higher probability. Drawing samples proportional to f is difficult for the rendering equation, because it is a product of several functions and all factors potentially contain high frequencies that are costly to evaluate. To draw samples from the product of several distributions *multiple importance sampling* (MIS) can be used [Veach and Guibas 1995].

To sample according to p, an inverse *cumulative density functions* (CDF) can be used. A CDF stores the cumulated probability density

$$F(x) = P(x \leq X) = \int_0^X p(x)\mathrm{d}x$$

that a random variable $f(x)$ with distribution $p(x)$ takes on a value less then X. This function is increasing strictly monotone because p is positive and can be inverted to $F^{-1}(X)$. When inserting a uniform random number ξ into F^{-1} the result is distributed according to p. Inverting F, can be done analytically or using a table.

Randomness Numerical integration can also be done using non-random, regular cubature: One could divide the d-dimensional unit-cube into elements of equal size. This is not done for two reasons.

First, with this approach the number of samples is exponential in the dimensionality d of the integrand (*curse of dimensionality*). For random numbers, the number of samples is *independent* of the dimension of the integrand.

Second, random numbers reduce perceived aliasing. This is not the same, as *removing* aliasing which can only be done by either increasing the sample rate or low pass filtering of the signal which is not practical for rendering as there is no known way to band-limit the RE signal and increasing the sample rate increases the rendering time. Transforming a regular low-frequency aliasing pattern into irregular high-frequency noise, is still as wrong as before, but artifacts are perceived less by a human observer [Yellott 1983] (Section 2.4.1).

Therefore instead of random numbers, any other pattern could be used, as long as it fills the entire n-dimensional space without aliasing. Some example patterns are shown in Figure 2.2.

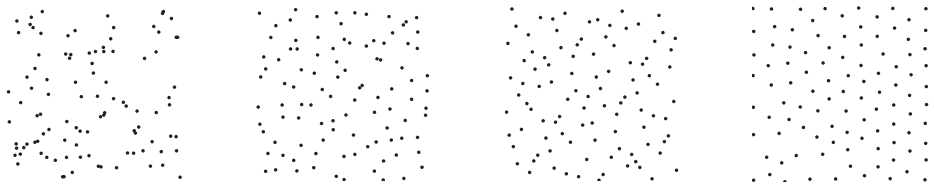

Figure 2.2: Some sample patterns, from Left to Right: random, jittered, Halton and Poisson (100 samples). Random samples have large clumps with many samples and large free areas. The jittered pattern guarantees that there is at least one sample in every stratum. Here 10×10 strata are used. However this does not prevent samples to clump across the borders of strata. The Halton pattern is more evenly spaced. The Poisson (blue noise) pattern shows no clumping, without forming regular patterns.

2.3.4 Finite Element Methods

A different way to solve the rendering equation is based on finite elements (FEM). Such methods turn the continuous RE into a discrete interaction between individual *finite elements*. The discretization is called *meshing*, and can be challenging for complex and dynamic geometry. In the simplest form, a finite element is a *patch* (sometimes called a cell or a simplex in FEM). FEM assumes that light variation across a patch is described by a finite basis. In the simplest case, the basis is a constant, and light is denoted in terms of view-independent radiosity. This configuration is called *radiosity* [Goral et al. 1984; Nishita and Nakamae 1985; Cohen, Wallace and Hanrahan 1993], which is limited to diffuse surfaces. Solutions including view dependent lighting are called *view dependent radiosity* [Aupperle and Hanrahan 1993]. Orthogonally, methods using higher order functions are called Galerkin methods [Zatz 1993], where the hierarchical wavelet basis (e. g. Haar) is a special case [Hanrahan, Salzman and Aupperle 1991; Gortler et al. 1993]. FEMs solve the RE by solving a large matrix of interaction between patches.

For view-independent radiosity, the RE that works on radiance can be rewritten as the view-independent *radiosity equation* that states the radiosity leaving a differential element $\mathbf{x}$ in all directions:

$$B(\mathbf{x}) = E(\mathbf{x}) + \rho(\mathbf{x}) \int_{\mathcal{S}} B(\mathbf{x}') \frac{\cos(\phi_x)\cos(\phi_{x'})}{\pi\|\mathbf{x}-\mathbf{x}'\|^2} V(\mathbf{x},\mathbf{x}') \mathrm{d}\mathcal{S}, \tag{2.4}$$

where $B(\mathbf{x})$ is the radiosity at point $\mathbf{x}$, $E(\mathbf{x})$ is emittance at $\mathbf{x}$, ρ the constant reflectance (diffuse material), $\mathcal{S}$ the entire scene surface, $B(\mathbf{x}')$ the radiance leaving

a different differential element at $\mathbf{x}'$, ϕ_x is the angle between the normal at $\mathbf{x}$ and the direction between $\mathbf{x}$ and $\mathbf{x}'$, $\phi_{x'}$ the angle between the normal at $\mathbf{x}'$ and the direction between $\mathbf{x}'$ and $\mathbf{x}$, and $V(\mathbf{x},\mathbf{x}')$ the visibility between $\mathbf{x}$ and $\mathbf{x}'$. In case of a piecewise constant basis (non-Galerkin radiosity), Equation 2.4 can be discretized into n patches, where radiosity B_i at the i-th patch is

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}, \tag{2.5}$$

where F_{ij} is called a *form factor* between patch i and patch j, or shorter in its matrix form:

$$\mathsf{B} = \mathsf{E} + \rho \mathsf{F}\mathsf{B} \tag{2.6}$$

where F is a $n \times n$ matrix, called the *form factor matrix*. This matrix stores in column i how much patch i contributes to all other patches and in row j how much patch j receives from all other patches.

The form factor F_{ij} includes distance, angular relation and visibility between the two patches $\mathcal{P}_i$ and $\mathcal{P}_j$. It can be calculated as

$$F_{ij} = \frac{1}{A_i} \int_{\mathbf{x}\in\mathcal{P}_i} \int_{\mathbf{x}'\in\mathcal{P}_j} \frac{\cos(\phi_x)\cos(\phi_{x'})}{\pi\|\mathbf{x}-\mathbf{x}'\|^2} V(\mathbf{x},\mathbf{x}') \mathrm{d}\mathcal{P}_j \mathrm{d}\mathcal{P}_i.$$

There are several ways to compute from factors. The first one is using Monte Carlo sampling by picking random points on $\mathcal{P}_j$ and $\mathcal{P}_i$ and using ray-tracing for visibility. The second one called the *hemicube* method [Cohen and Greenberg 1985]. Here, one patch (the smaller one) is considered a point, and the world is simply rasterized from that position, including depth buffering. By doing so, all form factors between patch i and all other patches (entire rows in F) are computed at once.

To solve the discrete radiosity Equation 2.6, several methods are used. The *direct method* computes a complete form factor matrix F and inverts it (including ρ). However, this requires to compute and store F, as well as inverting it, which is only practical for a low number of patches n.

A better approaches to solve Equation 2.6 for large n is *gathering*. In gathering, the complete form factor matrix F is still required but the equation is solved iteratively: First, all patches B^0 are initialized with E in iteration 0. Next, in the i-th iteration, all patches B^i gathers from all other patches of the previous iteration B^{i-1} using F. In each iteration F is the same, and it needs only to be computed once, which takes most of the time. Intuitively, each iteration corresponds to the i-th bounce of light. Usually after a low number of bounces remaining bounces can be ignored. One drawback of this approach is, that – while avoiding the inversion – the entire form factor matrix F needs to be computed once, which needs much storage and computation time.

Another approach to solve Equation 2.6 is progressive *shooting*, which again is iterative. Here, in each step a single sender patch (row) with the biggest unshot radiosity is chosen and only one row in F is computed. This avoids computing, or even inverting, the entire matrix F. Naturally, shooting is used together with the hemi-cube method, which calculates the required row in F [Cohen and Greenberg 1985].

Classic radiosity computes a view-independent solution for the entire scene. When the solution is computed, the scene can be inspected from an arbitrary viewpoint, as long as it is static. This has been used for architectural walk-troughs or to store global illumination in textures for computer games. None withstanding, it has to be acknowledged that radiosity does not receive much attention in current research anymore and is seldom used in practice.

2.3.5 Image Formation

Image formation in rendering usually makes two simplification: infinitely short exposure and a pinhole projection. This is different from a physical film and a physical lens. A physical film is exposed for a period of time, and contains the average over time. If objects moves, this leads to *motion blur*, which is not considered in this work. The difference between a physical lens and a pinhole model where a point is mapped onto a point is, that it spreads a point to a two-dimensional function, the *point spread function* (PSF). The effect of point spreading is most pronounced, when a small bright feature is spread over a large area, an effect called *glare*. Human eyes also have a PSF, and this works contributes a novel PSF model for dynamic human eyes in Chapter 8. The PSF also depends on wavelength, e. g. the PSF can be bigger for higher (red) wavelengths. Point spreading is also different for different distances, an effect called *depth of field*, which is also not considered in this work.

2.3.6 Tone Reproduction

The results of physically-based light simulation are radiance values. For color images, the wavelength-dependent effects need to be included. This is achieved by rendering the same image for a number of channels, usually in some RGB space. To convert a continuous spectral quality (i. e. reflectance) into an RGB tuple, the spectral function gets projected onto the three basis functions that span the RGB space. Output devices, such as screens, can not display all colors, the range they can display is called their *gamut*.

Besides gamut, another important limitation of any display device is the maximum luminance. While in reality, humans can deal with a *high dynamic range* (HDR) of up to five orders of magnitude [Reinhard et al. 2005], a screen can only display two to three orders of magnitude, which is hence called a *limited dynamic range* (LDR). Also cameras have a limited dynamic range to acquire images. Further, screens and cameras, traditionally use a gamma curve to compress resp. decompress luminance in a way that contributes more quantization steps to low values and store them using 8 bits. In both cases, very low and very high values are lost (clamped to white resp. black) or at least heavily quantized. Recently, hardware to display and capture HDR information became available. To present HDR images on a standard display, a mapping from HDR to LDR has to be used which is called *tone mapping*. Here, both the nonlinear display response and its limited dynamic range need to be considered. All approaches described in this thesis utilize HDR information. They are reproduced in LDR using a simple linear tone mapper, which works as follows. First, RGB values are converted to XYZ. Second, this triple is scaled such that the highest Y to be reproduced maps to 1. Nonlinearity is accounted for using a gamma mapping ($\gamma \approx 2.2$) that fits most current displays. Finally the result is converted to LDR by clamping and used for display.

2.4 Perception

The human visual system (HVS) is a fascinating topic. When rendering images, it is important to consider the HVS, because it is the last stage and an inevitable part of the rendering pipeline, which has important consequences. In this section we give a short introduction to a perceptual background. A lot of optical phenomena are not perceived by humans (e. g. absolute physical qualities like luminance or shape, or more obvious: polarization with some exceptions as noted by Haidinger [1844]). Consequently it is uneconomical to render them given limited resources. At the same time, the HVS is sensitive to details e. g. to edges, or temporal coherence which should be modeled faithfully.

Besides learning what is perceptually important to improve rendering algorithms, the opposite way has shown fruitful: Using computer graphics techniques has allowed to study perception more successfully by easily producing a wider variety of more naturalistic stimuli.

2.4.1 Human Vision

Light arrives at the human eye and is projected onto the retina which is covered with photoreceptors. The range of light the receptors can process starts at ca. 400 nm (blue), ends at ca. 700 nm (red) with a peek around 555 nm (green) [Palmer 1999] (Section 2.1.2). The human eye is able to adapt to fifteen orders of magnitude (10^{-6}–10^{9}) in luminance and four to five orders can be seen simultaneously. However, at a single point in time, the perceived contrast range is much smaller. This adjustment is called *adaptation* and takes between several seconds and many minutes. For different levels of luminance, different receptors are used [Palmer 1999]. In normal daylight, vision is called *photopic*, includes good color perception and is based on *cone* cells. Under dim conditions such as in closed rooms without direct light *mesopic* vision is active, where cones and *rods* are active. Finally, *scotopic* vision is found in dark conditions and is based on rods. The adaptation to scotopic vision can take several minutes. If the eye is exposed to a luminance level that does not fit the current adaptation state, its receptive performance decreases, a situation called *maladaptation*.

After light is received by photoreceptors on the retina, it undergoes a number of neural stages that produce increasingly abstract representations. *Receptive fields* transform absolute values into relative qualities and later, the "*what*" and the "*where*" pathway process information independently. The "what" system is related to recognition and categorization, while the "where" system is responsible for spatial localization. For computer graphics, two findings are important.

First, there is agreement that perception rather operates on contrast than on absolute luminance values. While retinal cells capture absolute values, it was shown by Kuffler [1953] that in the next higher level of processing, the strongest neural response is found for center-surround patterns, i. e. edges and gradients. For this reason, much attention has to be paid to discontinuities and dealing with them carefully in image manipulation can improve quality substantially [Tomasi and Manduchi 1998; Eisemann and Durand 2004; Kopf et al. 2007]. In the same sense, adding or removing gradients artificially can be a more powerful tool to change an image in a natural way ([Ritschel et. al. 2008c] and Chapter 8).

Second, information is processed in a multi-resolution fashion [Palmer 1999]. To this end, small spatial as well as broad spatial details are represented at the same time in a similar way and all scales should be considered. They form a context for *spatial vision* that deals with structured spatial areas instead of single point luminance. Spatial vision was modeled in computer graphics by Pattanaik et al. [1998]. Similar to the HVS itself, such models use a digital filter bank to find contextual features localized in space and frequency. It will be more

difficult to see quantization noise in an image which is cluttered with details, compared to an image with smooth gradations only, an effect called *visual masking* and first modeled computationally by Ferwerda et al. [1997]. To assess if humans are able to tell the difference between two images including masking, so called *image difference metrics*, such as the visual difference predictor [Daly 1993] were developed.

The size of details that the HVS can resolve, varies with contrast. A *contrast sensitivity function* (CSF) determines detection thresholds for spatial frequency. Not surprising, higher frequencies (smaller features) need more threshold to be detectable. However, CSFs were evaluated in an experiment using artificial stimuli such as Gabor patches and it is know that effects like masking can increase thresholds tremendously. Further, the thresholds vary with luminance adaptation. Under very low levels, more contrast is required. For stimuli that vary over time, a *temporal* contrast sensitivity function that determines thresholds for frequencies in space as well as in time, exists. From such a mapping it can be seen that certain lower frequencies also have lower thresholds and temporal changes can be strong attractors for visual attention. Depending on the situation, this can be desired or not. Noise can be more acceptable when it stays consistent over time, instead of introducing a flickering or pumping distraction. In other cases, such as glare (Chapter 8), temporal variation might be desired to attract attention to a point light, emphasizing its contrast.

2.4.2 Light, Material and Shape Inference

Helmholtz [1867] first proposed that the HVS uses an *unconscious inference* mechanism to construct a three-dimensional understanding from a two-dimensional image. It is not the only perception model, but as it fits rendering quite well, we will consider it in more detail here. Other models include *Gestalt* theory, *ecological* optics after Gibson [1950] [Palmer 1999] and Marr's [1982] *computational approach*. In Helmholtz' sense, perception is similar to *inverse rendering* as it involves estimating geometry, light and reflectance at the same time. This inference is an ill-posed problem, similar to *blind de-convolution*. Often blind de-convolution is concerned with images, while for three-dimensional signals the de-convolution is done in the directional domain. In general, reflected light after a bounce has less information than before the bounce, because the reflection operator is a directional convolution [Miller and Hoffman 1984; Ramamoorthi and Hanrahan 2001], and a convolution removes information. Inverse rendering performance in the HVS varies and mechanics besides inference also play an important role [Blake and Bülthoff 1990; Bloj et al. 2004; Fleming and Bülthoff 2005; Pellacini, Ferwerda

and Greenberg 2000].

Any further bounce – as in global illumination – is another convolution and provides less and less information, resp. it becomes more difficult to de-convolve the operator in an inverse rendering fashion. Therefore, indirect light is less important – and indirect shadows even less – in terms of localizing information. Still, indirect illumination (sometimes called *mutual* lighting in perception literature) is perceived in a specific way. Perception of shape influences color perception in indirect illumination [Bloj, Kersten and Hurlbert 1999]. The laws of diffuse light transport are to some extent understood by the HVS [Dörschner 2004]: In a study, naïve observers were able to correctly adjust light bounced from colorful paper to white paper, depending on angle. Also, indirect light has enough information to be considered in computer vision shape reconstruction [Forsyth and Zisserman 1990; Nayar, Ikeuchi and Kanade 1991]. Similar to direct light, the color of indirect light is more related to reflectance, whereas the intensity is associated with shape [Bloj and Ruppertsberg 2006]. All those findings are made for artificial stimuli however. In computer graphics indirect light in complex natural images is often reproduced in reduced quality [Stokes et al. 2004; Debattista et al. 2005]. In practical computer graphics, global illumination is mainly a cue for realism and is more perceived as a whole, setting the mood of a scene.

The human ability of canceling out the light color completely by inferring what constitutes the illuminant and what is material reflectance, is called *color constancy* [Palmer 1999; Maloney 1999]. What is called "color" in everyday life is reflectance, a material property, which is a much stronger perceptual cue than e. g. the chromaticity of the illuminant. Indirect illumination is believed to assist color constancy [Bloj and Hurlbert 1995]: Light bouncing off colorful surfaces becomes more saturated, whereas colorful light is unaffected chromatically.

Humans have impressive capabilities in telling apart different *materials* [Fleming, Dror and Adelson 2003]. Perception of materials is important, because materials often reveal key information about the state of an object, e. g. food. Similar to shape and light, material can in theory be recovered in an inverse rendering fashion. For simple object albedo, if a key light color and direction are known, the reflectance can be precisely calculated and infered by humans.

Texture refers to a reflectance pattern, is often part of a material concept and a visual cue on its own that plays a role e. g. in depth perception [Palmer 1999]. Observing a texture, we do not perceive individual elements, but more of a spectral property instead. As textures are by definition of high frequency, all rendering operations that apply even the slightest blur to a textured image can reduce their effect. To this end, whenever possible the approaches proposed in this thesis apply textures *after* a required blur.

2.4.3 Shadow

Shadow is a useful geometric cue that deserves accurate representation in an image. Observers deduce position and size [Wanger 1992; Mamassian, Knill and Kersten 1998; Hubona et al. 1999; Kersten, Mamassian and Knill 1997] of the occluder as well as geometry of occluder and receiver from a shadow [Wanger 1992]. However, some simplifications can be made, that humans are not able to detect [Sattler et al. 2005], especially in indirect shadows [Yu et al. 2009]. Observers estimate object motion and shape from the trajectory of shadows [Kersten, Mamassian and Knill 1997] and task performance can be improved by displaying shadows [Hubona et al. 1999]. One key shadow effect is, that objects in contact also exhibit an explicit *contact shadow* which can be challenging to reproduce in rendering (Section 4.5) as it is prone to singularities (i. e. for virtual point lights) and aliasing (i. e. shadow maps). Such a contact shadow can be sharp or even completely smooth, in any case it will make an object appear grounded.

Furthermore, soft shadows from area lights are a cue for the distance between occluder and occluded surface [Hasenfratz et al. 2003]: on a close distance a shadow is sharper than a shadow from a far-away object. This cue is strong enough to induce the perception of occluder movement by changing the shadow softness, as shown by Kersten et al. [1996]. The indirect shadows proposed in this work are inherently shadows cast by area lights (light-bouncing surfaces) and therefore produce soft shadows.

2.4.4 Reflections

Mirror reflections are a unique effect: They expose a copy of a scene to an observer inside the scene. However, this copy is flipped and smaller than its real-world counterpart [Corballis and McLaren 1984] and while observers identify objects, they might not perceive them as the original [Ittelson, Mowafy and Magid 1991]. Mirroring across a perceived axis of symmetry (including canonical axes, defined by social convention) is more acceptable then mirroring across an arbitrary axis [Ittelson, Mowafy and Magid 1991]. The laws of reflection are “understood” by the HVS which makes use of this understanding to infer shape from highlights Blake and Bülthoff [1990]. Nonetheless, human observers have difficulties in assessing the physical correctness of reflections on complex surfaces [Fleming, Dror and Adelson 2003; Khan et al. 2006; Ramanarayanan et al. 2007], which is exploited in art [Braham 1976] and contributes to the effectiveness of reflection maps [Miller and Hoffman 1984]. Observers expect to see smooth reflections of smooth features as an indicator for smooth geometry [Hagen et al. 1992; Theisel 2001; To-

sun et al. 2007]. To assess the quality (e. g. smoothness or continuity) of a physical surface, reflections of linear features, so called *reflection* lines, are used.

3

Previous Work

In this section we review the related work in global illumination rendering and editing as well as glare. Our discussion of interactive global illumination rendering in Section 3.1 will start with theoretical and non-real-time work, we will later focus on real-time techniques using efficient visibility, especially those that are based on perception. Thereafter, we review interactive illumination editing, as well as some related reflectance and shape-editing approaches in Section 3.2. Previous work to render glare effects is mentioned in Section 3.3.

3.1 Interactive Global Illumination

In the following we will discuss relevant interactive global illumination methods with a focus on their use of visibility. We argue, that efficient and perception-based computation of visibility is a key ingredient to efficient global illumination.

Several books give an introduction to global illumination. A classic, that focuses on radiosity, is Cohen, Wallace and Hanrahan's [1993] book. Advanced global illumination is covered in Dutré, Bekaert and Bala's [2003] work. Pharr and Humphreys's "PBRT" from [2004b] serves as an excellent textbook while providing a full reference implementation of a physically-based rendering software. "PBRT" was also used as a reference implementation for results produced by techniques presented in this thesis. Further, real-time rendering was recently covered [2008] in a book by Akenine-Möller, Haines and Hoffman.

While an excellent survey on interactive (soft) shadow rendering exists [Hasenfratz et al. 2003], there is no up-to-date survey on interactive global illumination rendering or editing. A survey by Durand [2000] covers visibility also beyond

computer graphics and recent surveys by Bittner and Wonka [2003] and Cohen-Or et al. [2003] cover exact visibility.

Durand et al. [2005] proposes a high-level, formal analysis of light transport in terms of spatial and angular frequencies, including visibility. Similar in spirit Ramamoorthi, Mahajan and Belhumeur [2007] analyze gradients in direct lighting, shading and shadows.

3.1.1 Exact Visibility

Exact visibility is 1 if two points in space can see each other and 0 otherwise (Section 2.3.1). In computer graphics this is mainly resolved using either ray-tracing [Appel 1968] in a forward mapping or using a backward mapping based on rasterization and z-buffering [Straßer 1974] as in shadow mapping [Williams 1978] or the hemi-cube [Cohen and Greenberg 1985]. Between two areas in space, fractional visibility, a continuum ranging from 0 to 1, is defined.

In local illumination, visibility is used to determine if a path between a surface location and a point light source is blocked, resulting in shadow [Whitted 1980]. In global illumination, the light coming from the blocking geometry is also taking into consideration.

Finite element global illumination methods (Section 2.3.4), such as radiosity [Cohen, Wallace and Hanrahan 1993], use form factors between pairs of finite elements ("patches"), including fractional visibility. In early work, Goral et al. [1984] assumed a scene without occlusion while Nishita and Nakamae [1985] already used shadow volumes to resolve visibility. The hemi-cube [Cohen and Greenberg 1985; Max 1995] explicitly handles visibility by introducing the idea of patches that "see" each other, an intuition also used in Chapter 6. Later, Baum, Rushmeier and Winget [1989] subdivide patches until visibility between them becomes either 0 or 1, imposing a high geometric complexity. Monte Carlo sampling of visibility between two patches can be done using ray-tracing [Wallace, Elmquist and Haines 1989]. Analytical form factors between two arbitrarily oriented polygons were presented by Schröder and Hanrahan [1993], but did not include visibility.

Monte Carlo methods, such as path tracing [Kajiya 1986] use ray-tracing to resolve visibility and modern physically-based ray-tracing systems established in production, do as well [Dutré, Bekaert and Bala 2003; Pharr and Humphreys 2004b]. Also Photon Mapping [Jensen 1996; Purcell et al. 2003] traces exact rays to distribute photons as well as to compute final gathering.

3.1.2 Approximate Visibility

Methods exist that make more explicit use of visibility approximations. They consider visibility a signal as any other, and compute it efficiently, without a focus on perception, as in Section 3.1.6.

Sillion [1995] proposed a radiosity method that approximates visibility such that a user-defined feature-size can still be resolved while blurring out smaller features. No distinction was made between direct and indirect illumination.

Arikan, Forsyth and O'Brien [2005] propose an approximate global illumination method where irradiance arriving from nearby and distant geometry is decoupled: all incident lighting from nearby geometry is simply integrated without computing visibility at all. Arvo, Torrance and Smits [1994] analyze error in global illumination algorithms, including error from visibility discretization, but do not propose algorithms to make use of this information.

Methods have been proposed to reduce the geometric complexity when computing global illumination solutions [Rushmeier, Patterson and Veerasamy 1993; Christensen et al. 2003; Tabellion and Lamorlette 2004]. In particular, geometric approximations are used when computing shading.

3.1.3 Virtual Point Lights

Instant radiosity [Keller 1997] and instant global illumination [Wald et al. 2002] use *virtual point lights* to represent indirect light. Both methods use accurate visibility either through shadow volumes, shadow maps or ray-casting.

Point lights are easier and more efficient to handle, i. e. compared to finite element meshes, while having many desirable sampling properties. Besides indirect light, other continuous lights, such as linear lights, area lights, environment maps [Agarwal et al. 2003] or even many direct lights can be discretized into virtual point lights, leading to a single, effective *many-lights* [Hašan, Pellacini and Bala 2007] representation of scene lighting.

Lightcuts [Walter et. al. 2005] is one option to render efficiently from such a many-lights representation by grouping all lights into a hierarchy. For every visible surface point, a cut through the hierarchy is computed, and the contributions of all clustered point lights are summed up; visibility is resolved by ray-casting towards the center of each cluster. However, the algorithm bounds the resulting error, such that no perceptible artifacts appear. Multidimensional Lightcuts [Walter et. al. 2006] proposes to reduce banding artifacts in the original approach by using multiple representatives. Recently, lightcuts were also extended to com-

bine several cuts through a hierarchy of lights, visibility, and BRDFs [Cheslack-Postava et al. 2008]. While this extension allows for interactive performance, it does assume static geometry.

Matrix row-column sampling [Hašan, Pellacini and Bala 2007] views the lighting of many pixel samples by many lights as a large matrix of sample-light interaction. They computes individual important rows (pixels, samples) or columns (lights) using graphics hardware and shadow mapping instead of varying it per image location. While the most important rows and columns can be found for some scenes, in other scenes with much occlusion the important lights change drastically and individual rows (samples, pixels) are lit only by a few columns (lights) resp. individual columns (lights) contribute to only a few rows (samples, pixels).

Hašan et al. [2009] generalize point to sphere lights, to avoid several singularity problems of VPLs. In concurrent – but somewhat orthogonal – work, Dong et al. [2009] generalize virtual point lights to virtual area lights with a focus on visibility. Binary, ray-tracing-based visibility is replaced by an efficient soft shadow query, yielding fractional visibility. A combination of Hašan et al.'s light and Dong et al.'s visibility and clustering might be worth exploration.

Real-time global illumination for dynamic models is possible if visibility for indirect illumination is completely neglected. Dachsbacher and Stamminger [2005] and Dachsbacher and Stamminger [2006] use a variant of shadow maps, called *reflective shadow maps* to efficiently compute the first indirect bounce: rendering the scene from the direct light's point of view into a cube map, results in pixels that represent all first-bounce light emitters. Such pixels are equivalent to one-bounce VPLs, and indirect lighting can be sampled from them efficiently e. g. by picking a random subset of pixels, eventually with importance sampling [Dachsbacher and Stamminger 2006]. A second bounce remains difficult, as it would required to render the scene again, but this time from many new first-bounce view-points instead of a single direct light viewpoint. Reflective Imperfect Shadow Maps (Section 5.3.1) generalize the idea to multiple bounces.

Real-time rendering of static scenes with one-bounce indirect illumination is possible with a variant of instant radiosity that reuses VPLs and their respective shadow maps over multiple frames [Laine et al. 2007]. For scenes with much animation, the indirect lighting and indirect shadows can get out-of-sync.

Nichols and Wyman [2009] improve on splatting-based VPL techniques, by splatting into a multi-resolution frame-buffer, exploiting the smooth nature of many indirect lighting phenomena.

3.1.4 Precomputed Radiance Transfer

In *Precomputed Radiance Transfer*, (PRT) rendering is done in two steps. First, a precomputation is run on the scene (taking between some seconds and many hours) that outputs an intermediate result. When interacting with the scene, the intermediate result is used for rendering. Usually, visibility evaluation is done in the precomputation and stored in a compressed form as the intermediate result.

In classic radiosity, diffuse lighting was stored for vertices, or later in textures. For such scenes, geometry was fixed while lighting could be manipulated to an extent.

Sloan et. al. [2002] used spherical harmonics to store the radiance transfer according to low-frequency distant lights, including arbitrary low-frequency BRDFs and even subsurface scattering. Their approach allows to change viewpoint and environment lighting at high frame rates, but keeps reflectance and geometry fixed. They store spherical harmonics at vertices, but textures could be used as well. Clustering of SH-based PRT information [Sloan et. al. 2003a] can further reduce storage. A limited number of precomputable deformations can also be supported using Zonal Harmonics [Sloan et. al. 2005]. An extension to local illumination is possible using spherical harmonic gradients [Annen et. al. 2004].

In a recent PRT approach Lehtinen et al. [2008] uses a hierarchical, point-based basis that is well-suited to complex scenes. It allows local, dynamic lights, but requires static geometry.

In wavelet PRT [Ng et. al. 2003 ACM Trans. on Graphics (Proc. ACM SIGGRAPH); Ng et. al. 2004 ACM Trans. on Graphics (Proc. ACM SIGGRAPH); Wang et. al. 2004], a wavelet representation of radiance transport is used. It allows for more high-frequency BRDFs and shadows, but is slightly slower. Still geometry and BRDF remain fixed and lighting is distant. Also dealing with rotations in a wavelet basis is difficult [Wang et. al. 2006]. Ma et al. [2006] propose several improvements for wavelet-based PRT. They store BRDFs in a local frame using a spherical parametrization to ease the handling of spatial variation and bump mapping. Also, they store directional visibility information as wavelets in a texture, which allows per-pixel visibility, but requires a high-quality parametrization. While many PRT approaches are limited to distant illumination, Sun and Ramamoorthi [2009] propose an extension to near field (local) lights. In this framework, factors (i. e. a local area light) are allowed to scale and translate, but not rotate. However, not all desirable properties of distant wavelet PRT are preserved, e. g. the triple-product becomes more involved.

PRT that allows movement of rigid objects was investigated by Iwasaki et al. [2007] and Wang, Zhu and Humphreys [2007].

The technique described in [Ritschel et. al. 2008a] proposes to replace the precomputed lighting response used in PRT by precomputed depth. Precomputing depth has the same cost as precomputing visibility, but allows visibility tests for moving objects at runtime using simple shadow mapping. For this purpose, a compression scheme for a high number of coherent surface shadow maps (CSSMs) covering the entire scene surface is developed. CSSMs allow visibility tests between all surface points against all points in the scene. The effectiveness of CSSM-based visibility is demonstrated using a novel combination of the lightcuts algorithm and hierarchical radiosity, which can be efficiently implemented on the GPU.

Nowrouzezahrai, Kalogerakis and Fiume [2009] combine reflectance and visibility in a compact representation that allows efficient rendering. However, the representation is limited to low-frequency (spatially and angular) visibility and low-frequency (angular) BRDFs.

A mixture – relevant for architectural visualization in practice – between local and distant lights is proposed by Yue et al. [2009], where distant illumination can be animated but local lights (windows) are used to light interior scenes using PRT.

Wang et al. [2009a] address static scenes with dynamic high-frequency BRDFs and distant lights, which in turn requires high-frequency visibility to be effective. As their scenes are static, they pre-compute visibility coarsely, and use signed distance functions to interpolate it across the surface.

In [Ritschel 2007], an efficient way to use PRT for volume data is proposed. While PRT has been used to render volumetric data under distant low-frequency illumination at real-time rates, including natural illumination, soft shadows, attenuation from semi-transparent occluders and multiple scattering, it requires a lengthy preprocess, which is acceptable only for static volume data. However, in practical volume rendering, general transfer functions are used. Manipulating such a transfer function will result in a dynamic radiance transfer which has to be re-computed. Also some scenes used in interactive applications like games might be dynamic, e˙ g. moving clouds. While other work has used CPU Monte Carlo ray-tracing for precomputation and requires time in the order of many minutes, the GPU technique proposed in [Ritschel 2007] uses a hierarchical visibility approximation that requires only a few seconds for typical scenes.

3.1.5 Finite-element Methods

An approximate FRM method based on disk-elements taking into accoutn approximate visibility was presented by Bunnell [2005]. It uses a hierarchical link structure, that can deform but never changes topology, which is suitable, i. e. for

characters. It is an approximation, as it can result in multiple shadows or multiple bounces where shadows or bounced light is not accurately blocked. Instead, [Bunnell 2005] proposes to use multiple iterations to approximately remove doubled (or tripled and so forth) shadows or bounces.

A variant of hierarchical radiosity, which replaces explicit visibility queries with an iterative process using anti-radiance (negative light), has been shown to enable interactive, dynamic global illumination [Dachsbacher et al. 2007]. However, the use of dynamic objects is restricted due to the need for highly tessellated surfaces near indirect shadows. This also requires using a directionally discretized data structure, akin to omni-directional shadow mapping. Only limited object movements were supported, since the hierarchical link structure was computed beforehand. Extra work is needed to update links [Meyer et al. 2009].

Dong et al. [2007] proposed an interactive global illumination method for small scenes, where visibility is implicitly evaluated on the hierarchical finite element structure.

Schmitz, Tavenrath and Kobbelt [2008] present a CUDA-based approach to interactive global illumination. They use explicit visibility. The results reported are difficult to assess, as they only update global illumination in a lazy fashion.

Low-resolution dynamic scenes are possible, assuming low-frequency illumination [Ren et al. 2006]. Similar, Sloan et. al. [2007] demonstrate real-time indirect illumination for large and dynamic scenes, but only low-frequency incident radiance as well as low-frequency visibility are supported as these are represented with a small number of spherical harmonics.

Wang et. al. [2009b] demonstrate interactive global illumination using GPU-based final gathering with ray-tracing. It enables complex lighting effects but relies on sparse gathering locations for efficiency.

3.1.6 Perceptual Visibility

Since computing global illumination solutions is very expensive, perceptually-based rendering methods have been explored. Their goal is to speed up the process by taking the limits of the human visual system into account. Using the visual differences predictor (VDP) [Daly 1993] to determine if an approximation is indistinguishable from a reference image is a common approach [Volevich et al. 2000]. A spatio-temporal extension can be used for efficient rendering of dynamic environments [Yee, Pattanaik and Greenberg 2001]. Perceptually-based error metrics have also been used to efficiently compute animation sequences [Myszkowski et al. 2001]. Stokes et al. [2004] separate the rendering process

into individual illumination components and identify perceptually important ones, which can be used to speed up the computation [Debattista et al. 2005]. The work by Vangorp, Laurijssen and Dutré [2007] predicts how a change in incident illumination affects the appearance of an object, depending on its geometry and material, which can be used to speed up rendering.

Recently, it has been shown that accurate occlusion for glossy reflections is not always necessary [Kozlowski and Kautz 2007] — an observation that has already been exploited in rendering glossy reflections from environment maps [Green, Kautz and Durand 2007].

Yu et al. [2009] evaluate the use of approximate visibility for efficient diffuse global illumination in a perceptual study. It is assessed, how accurate visibility is psychophysical necessary in techniques such as AO (Section 3.1.7), screen-space directional occlusion (Chapter 4), imperfect visibility (Chapter 5), ray-tracing, as well as techniques that ignore visibility completely. The study revealed, that accurate visibility is not required and that certain approximations may be introduced.

As shown by Herzog et al. [2008], global illumination rendering quality can be adapted to account for perception of MPEG encoding of rendered results. They use lightcuts [Walter et. al. 2005] in order to adapt rendering quality (i. e. cut depth) to an MPEG stream in one direction and producing exact motion vectors in the other direction. Consequently, they render only details that will survive MPEG encoding and help MPEG encoding to encode details they render. Still, visibility is computed using shadow maps for all lights, which remains the bottleneck in geometrically complex scenes.

3.1.7 Ambient Occlusion

The term *Ambient Occlusion* (AO) describes the remaining surface shading that is perceived on a cloudy day [Langer and Bülthoff 2000]: Cavities on a surface appear dark, non-concave regions remain unaffected. Langer and Bülthoff [2000] study how this effect is an essential part of human shape perception, which also manifests as the principle that “dark means deeper” [Khan et al. 2006]. Despite the perceptual importance, AO was not explicitly named or received interest in computer graphics for a remarkable time before “Accessibility Shading” [Miller 1994] — an effect very similar to AO — was proposed. Earlier, it was only understood as a by-product of general global illumination, which tries to solve a much wider class of problems.

By now, AO is used in computer graphics production extensively [Zhukov, Iones and Kronin 1998; Landis 2002] because of its speed, simplicity and ease of im-

plementation. While physically correct illumination computes the integral over a product of visibility and illumination for every direction, AO computes a product of two individual integrals: one for visibility and one for illumination. For static scenes, AO allows to precompute visibility and store it as a scalar field over the surface (using vertices or textures). Combining static AO and dynamic lighting using a simple multiplication gives perceptually plausible results at high frame rates. An efficient way to compute static AO was proposed by Sattler et al. [2004]. To account for dynamic scenes, Kontkanen and Laine [2005] introduced AO Fields, which allow rigid translation and rotation of objects and specialized solutions for animated characters exist [Kontkanen and Aila 2006; Kirk and Arikan 2007]. A special solution for trees was explored by [Hegeman et al. 2006]. Deforming surfaces and bounces of indirect light are addressed by Bunnell [Bunnell 2005] using a set of disks to approximate the geometry. A more robust version was presented by Hoberock and Jia [2007], which was further extended to point-based ambient occlusion and interreflections by Christensen [2008]. Méndez, Sbert and Catá [2003] compute simple color bleeding effects using the average albedo of the surrounding geometry.

These methods either use a discretization of the surface or rely on ray-tracing, which both do not scale well to the amount of dynamic geometry used in current interactive applications like games. Therefore, instead of computing occlusion over surfaces, recent methods approximate AO in *screen space* (SSAO) [Franklin 2006; Shanmugam and Arikan 2007; Mittring 2007; Bavoil, Sainz and Dimitrov 2008; Filion and McNaughton 2008]. The popularity of SSAO is due to its simple implementation and high performance: It is output-sensitive, applied as a post-process, requires no additional data (e. g. surface description, spatial acceleration structures for visibility like BVH, kD-trees or shadow maps) and works with many types of geometry (e. g. displacement / normal maps, vertex / geometry shaders, iso-surface ray-casting). Image-space methods can also be used to efficiently simulate subsurface scattering [Mertens et al. 2005; Dachsbacher and Stamminger 2003]. At the same time, SSAO is an approximation with many limitations, that also apply to this work, as we will detail in the following sections.

Light transport in meso-structure, which is similar to image space was addressed previously by Sloan et. al. [2003b]. While it is not limited to AO, the similarity is, to exploit surface meso-structure.

Reinbothe, Boubekeur and Alexa [2009] uses a fast scene voxelizsation [Eisemann and Décoret 2006] to compute fast AO. They avoid limitations of screen-space methods, but require a scene voxelization. In their results, the voxel grid has a lower resolution than the screen resolution, which ignores AO from very fine details. Also details from displacement and bump maps are not considered in the

voxelization and do not appear in the AO. A combination of this approach for far-field and image-space approaches for near-field blocking should capture most AO.

Later in [2009], Nowrouzezahrai and Snyder proposed a method to compute global illumination based on dynamic height fields. They use a multi-resolution approach to search for blockers resp. bouncers in a local neighborhood at different levels. They only demonstrate results for height-fields although an interpretation of a deferred shading frame-buffer as a height-field might be possible with some extensions.

McGuire and Luebke [2009] propose Image Space Photon Mapping. They use techniques similar to the aforementioned VPL-based approaches, but based on photon mapping. In photon mapping, a photon contributes only to a local neighborhood via splatting instead of the gathering that needs to be done from all VPLs. While they accelerate the first bounce as well as final gathering, they still use ray-tracing to bounce photons in-between.

Image space gathering [Robison and Shirley 2009] is based on the idea of blurring ray-traced results in image space. It can be used to compute "blurry" effects in screen space, instead of sending multiple rays per pixel. Applications include glossy reflections and soft shadows, but would also allow blurred indirect lighting or AO.

3.1.8 Final Gathering

Final gathering refers to the process of calculating the amount of indirect illumination at a surface point. For instance, it is an integral part of photon mapping [Jensen 1996]. It is usually the most time consuming part in GI and, not surprisingly, many algorithms aim at reducing the final gathering cost.

Irradiance caching [Ward, Rubinstein and Clear 1988] performs final gathering only at a sparse set of locations in the image, and interpolates irradiance values for all other pixels. The irradiance caching locations have to be chosen carefully to ensure good results. However, gathering and sparse sampling in image space are two orthogonal concepts. The irregular and – in the original work by Ward, Rubinstein and Clear [1988] strictly sequential – placement to of samples does not fit and might not pay off for GPUs that require coherent memory access and instruction streams.

Irradiance gradients [Ward and Heckbert 1992] additionally compute the gradient of the irradiance to enable better interpolation. Křivánek et al. [2005] extended irradiance caching to glossy surfaces by storing incident radiance instead of irradi-

ance. Another extension [Gassenbauer, Křivánek and Bouatouch 2009] allows for shiny surfaces with high-frequency BRDFs.

GPU-based gathering using environment maps has been used in the context of irradiance volumes [Mantiuk, Pattanaik and Myszkowski 2002]. However, importance sampling is not possible and rendering quality is limited due to the low number of gathering samples used.

Recently, Christensen [2008] proposed a CPU-based method to speed up final gathering for diffuse and moderately glossy scenes using a point-based representation of direct illumination stored in an octree. At each irradiance cache location, distant points are rasterized into a cube map and nearby points are ray-cast, but since no importance warping is used, glossiness is directly limited by the buffer's resolution.

3.2 Interactive Editing

Several methods have been proposed to manipulate lighting effects in a rendered 3D scene, as an alternative to manually setting raw lighting parameters.

3.2.1 Light Editing

Most of them use sketching or painting interfaces that allow users to directly draw lighting effects [Schoeneman et al. 1993; Poulin, Ratib and Jacques 1997; Pellacini et al. 2007; Todo et al. 2007; Okabe et al. 2007; Obert et al. 2008; Kerr and Pellacini 2009]. While these methods are effective in controlling smooth (global) illumination and low-frequency BRDFs, we argue that the view-dependent, high-frequency complexity of reflections and refractions is better addressed by reflection editing (Chapter 7), which deforms the existing reflections and refractions. Light painting techniques become prohibitively work-intensive, if the view-dependent light pattern (in our case a reflected object) is complex. Considering the reflection of a car, a paint-based interface would require to re-paint the reflection in a different place to move it across a surface. Anjyo and Hiramitsu [2003] proposed the interactive control of highlight shapes by painting, but this technique is limited to the area of cartoon animations.

Gingold and Zorin [2008] use painted changes in light to define a change in shape. A user is provided with image manipulation tools, to change shading, that is used in an optimization approach to fit a surface. They only address diffuse surfaces.

3.2.2 Appearance Editing

Automated lighting design systems have also been proposed [Shacked and Lischinski 2001; Rusinkiewicz, Burns and DeCarlo 2006]. In theory, reflections outside physical bounds could also be realized using existing BRDF [Colbert, Pattanaik and Krivánek 2006] or BTF [Kautz, Boulos and Durand 2007] editing approaches. However, manually specifying an artist-directed, optimal and smooth per-pixel mirror direction (as resulting from the approach proposed in this work in Chapter 7) would be a prohibitively laborious task.

3.2.3 Perception of Reflections

Reflections of real-world illumination are important for human perception of material appearance and shape, however, it is difficult for human observers to assess illumination consistencies [Fleming, Dror and Adelson 2003; Ostrovsky, Cavanagh and Sinha 2005] and the correctness of a given reflection [Ramanarayanan et al. 2007]. This fact was exploited by Khan et al. [2006], who used parts of a photograph to approximate glossy environment maps. The human difficulties in understanding reflection patterns can be generalized to caustics [Gutierrez et al. 2008].

Reflection lines are reflections of linear features on a mirroring surface, similar to isophotes [Theisel 2001]. Tosun et al. [2007] use reflection lines to optimize surfaces in an offline process, which is a common practice in the automotive industry. Observers (which are also customers in the case of automotive design) expect to see smooth reflections lines as an indicator for smooth geometry [Hagen et al. 1992].

3.2.4 Intuitive Deformation

Editing reflections can be understood as deforming the field of reflection directions on a surface. Many techniques for the intuitive deformation of images [Igarashi, Moscovich and Hughes 2005] or surfaces [Sorkine and Alexa 2007] now exist. Our approach adapts a Moving Least Squares cost function, which has previously been successfully used in the domain of shape deformation [Müller et al. 2005; Schaefer, McPhail and Warren 2006].

3.2.5 Manual Solutions

Although unpublished, it can be assumed that some production houses have in-house tools to manipulate reflections. Such manipulations can be added to a professional rendering pipeline, e. g., by applying a global linear transformation within a programmable shader [Kopra 2007]. However, we are not aware of any specific software that explicitly addresses reflections or is similar to our interactive user-constraint-driven system, which allows smoothly blended, local, non-linear reflection edits.

3.3 Temporal Glare

An introduction to wave optics – the background for "Temporal Glare" (Chapter 8) – is given by Hecht [1998], an introduction to Fourier Optics in Goodman [2005]. However, both are physics text books and often beyond the scope of computer graphics. More background on the depiction of bright lights is found in Reinhard et al.'s HDR book [2005].

The modeling of glare effects has been used to improve image realism and to convey an impression of high intensity of luminaires in the context of realistic rendering [Spencer et al. 1995], driving simulation [Nakamae et al. 1990], computer games [Kawase 2005], image post-processing [Rokita 1993], and tone mapping [Larson, Rushmeier and Piatko 1997; Durand and Dorsey 2000]. A recent perceptual study [Yoshida et al. 2008] demonstrates that the impression of displayed image brightness can be increased by over 20% by convolving high intensity pixels in the image with relatively simple filters used in glare models [Spencer et al. 1995; Kawase 2005]. Precise information about the intensity values of light sources and highlights represented by such pixels is immediately available in 3D image synthesis and HDR photography [Reinhard et al. 2005]. For physically based rendering and photometrically calibrated cameras, such pixel intensity can even be properly scaled in cd/m^2 units, which is important for faithful modeling of light scattering in the eye.

The majority of existing approaches to computer-generated glare, while inspired by knowledge about human eye anatomy and physiology, are based on phenomenological results rather than explicit modeling of the underlying physical mechanisms. A common approach is to design convolution filters, which reduce image contrast in the proximity of glare sources up to full image saturation in the glare center. Nakamae et al. [1990] derive such a filter to model the light diffraction on the eye pupil and eyelashes for various wavelengths. Spencer et al. [1995] base their

filter on the point-spread function (PSF) measured for the optics of the human eye. Glare solutions used in tone mapping [Larson, Rushmeier and Piatko 1997; Durand and Dorsey 2000] are mostly based on Spencer et al.'s approach. A set of Gaussian filters with different spatial extent, when skillfully applied, may lead to very convincing visual results. This approach is commonly used in computer games [Kawase 2005] and rendering post-production [Reinhard et al. 2005 Section 9.2.5]. Other glare effects such as the ciliary corona and the lenticular halo are often designed off-line, and placed in the location of the brightest pixel for each glare source as a billboard (image sprite) [Rokita 1993; Spencer et al. 1995]. In the designing of such billboards, seminal ophthalmology references are used such as [Simpson 1953]. The resulting appearance is very realistic for small point-like glare sources. However, using billboards, it is difficult to realistically render glare for glare sources of arbitrary shape and non-negligible spatial extent.

Recently, there have been some successful attempts to model glare based on the principles of wave optics. Kakimoto et al. [2004] propose a practical model to simulate scattering from a single plane rigid aperture. Three diffraction-causing obstacles: the eyelashes, the eyelids, and the pupil edge are placed in this plane. The Fraunhofer diffraction formula is then used to determine the diffraction pattern of the obstacle-plane on the retina. This pattern is stored as a billboard, placed at high-intensity pixels and blended with the rendered image. Similarly, van den Berg, Hagenouw and Coppens [2005] describe glare as diffraction by particles in the lens. By computing this pattern for multiple wavelengths, they achieve the first physical simulation of the ciliary corona.

A CG model for iris deformation was proposed by Pamplona, Oliveira and Baranoski [2009]. They measure individual iris deformation behavior, reacting to environment lighting. Their output could also be included in the dynamic aperture which is proposed in this work to simulate individualized glare.

4

Image Space Directional Occlusion

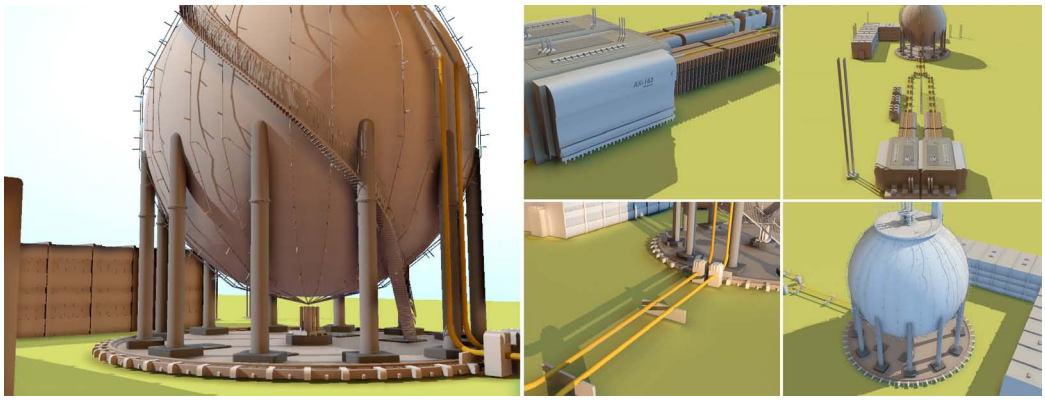

Figure 4.1: This work generalize screen-space ambient occlusion (SSAO) to directional occlusion (SSDO) and one additional diffuse indirect bounce of light. The scene presented contains 537 k polygons and runs at 20.4 fps at 1600×1200 pixels. Both geometry and lighting can be fully dynamic.

4.1 Introduction

Real-time global illumination is still an unsolved problem for large and dynamic scenes. Currently, sufficient frame rates are only achieved through approximations. One such approximation is ambient occlusion (AO) (cf. Section 3.1.7), which is often used in feature films and computer games, because of its high speed and simple implementation. However, AO decouples visibility and illumination,

allowing only for a coarse approximation of the actual illumination. AO typically displays darkening of cavities, but all *directional* information of the incoming light is ignored. This work extends recent developments in screen-space AO towards a more realistic illumination called *screen-space directional occlusion* (SSDO).

AO is a coarse approximation to general light transport as in PRT Section 3.1.4, which also supports *directional occlusion* (DO) and interreflections. The proposed approach allows to both resolve very small surface details and all angular resolutions: "no-frequency" AO, all-frequency image-based lighting and sharp shadows from point lights. While PRT works well with distant lighting and static geometry of low to moderate complexity, its adaptation to real applications can be difficult, while SSAO is of uncompromised simplicity.

4.2 Near-field Light Transport in Image Space

To compute light transport in image space, SSDO uses a framebuffer with positions and normals [Saito and Takahashi 1990] as input, and outputs a framebuffer with illuminated pixels using two rendering passes: One for *direct light* and another one for *indirect bounces*.

4.2.1 Direct Lighting using DO

Standard SSAO illuminates a pixel by first computing an average visibility value from a set of neighboring pixels. This occlusion value is then multiplied with the un-occluded illumination from all incoming directions. This chapter's contribution is, to *remove this decoupling* of occlusion and illumination in the following way:

For every pixel at 3D position $\mathbf{P}$ with normal $\mathbf{n}$, the direct radiance L_{dir} is computed from N sampling directions ω_i, uniformly distributed over the hemisphere, each covering a solid angle of $\Delta\omega = 2\pi/N$:

$$L_{\text{dir}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\text{in}}(\omega_i) V(\omega_i) \cos\theta_i \Delta\omega.$$

Each sample computes the product of incoming radiance L_{in}, visibility V and the diffuse BRDF ρ/π. We assume that L_{in} can be efficiently computed from point lights or an environment map. To avoid the use of ray-tracing to compute the visibility V, occluders are approximated in *screen space* instead. For every sample, a step of random length $\lambda_i \in [0 \dots r_{\max}]$ from $\mathbf{P}$ in direction ω_i is taken, where $r_{\max}$ is a user-defined radius. This results in a set of sampling points $\mathbf{P} + \lambda_i \omega_i$, located in

a hemisphere, centered at **P**, and oriented around **n**. Since the sampling points are generated as three-dimensional positions in the local frame around **P**, some of them will be above and some of them will be below the surface. In SSDO, approximate visibility test, all the sampling points *below* the surface of the nearby geometry are treated as *occluders*. Figure 4.2 (left) shows an example with $N = 4$ sampling

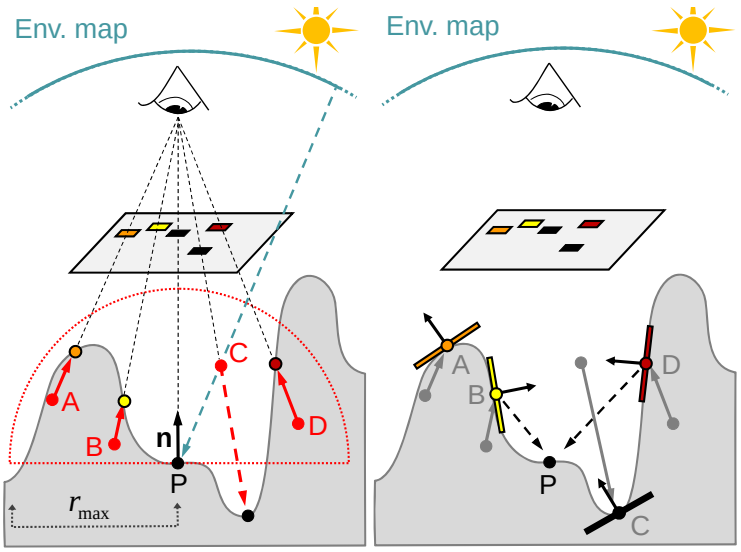

Figure 4.2: Left: For direct lighting with directional occlusion, each sample is tested as an occluder. In the example, point P is only illuminated from direction C. Right: For indirect light, a small patch is placed on the surface for each occluder and the direct light stored in the framebuffer is used as sender radiance.

points **A**, **B**, **C** and **D**: The points **A**, **B** and **D** are below the surface, therefore they are classified as *occluders* for **P**, while sample **C** is above the surface and classified as *visible*. To test if a sampling point is below the surface, the sampling points are back-projected to the image. Now, the 3D position can be read from the position buffer and the point can be projected onto the surface (red arrows). A sampling point is classified as *below* the surface if its distance to the viewer *decreases* by this projection to the surface. In the example in Figure 4.2, the samples **A**, **B** and **D** are below the surface because they move towards the viewer, while sample **C** moves away from the viewer. In contrast to SSAO, SSDO approach does not compute the illumination from all samples, but only from the visible directions (Sample **C**). Including this directional information can improve the result significantly, especially in case of incoming illumination with different colors from different directions. As shown in Figure 4.3, the technique can correctly display the resulting colored shadows, whereas SSAO simply displays a grey shadow at each location.

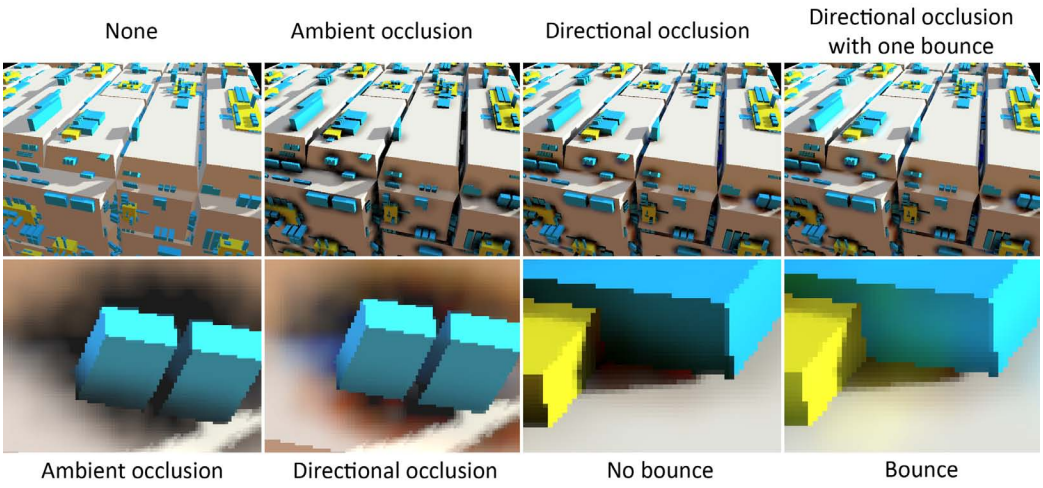

Figure 4.3: The top row shows the difference between no AO, standard SSAO, SSDO method with directional occlusion (SSDO) and one additional bounce. In this scene an environment map and an additional point light with a shadow map are used for illumination. The insets in the bottom row show the differences in detail. With SSDO, red and blue shadows are visible, whereas AO shadows are completely grey (bottom left). The images on the bottom right show the indirect bounce. Note the yellow light, bouncing from the box to the ground. The effect of dynamic lighting is seen best in the supplemental video.

4.2.2 Indirect Bounces

To include one indirect bounce of light, the direct light stored in the framebuffer from the previous pass can be used: For each sampling point which is treated as an occluder (**A**, **B**, **D**), the corresponding pixel color L_{pixel} is used as the *sender radiance* of a small patch, oriented at the surface (see Figure 4.2, right). Considering the sender normal here allows to avoid color bleeding from back-facing sender patches. The additional radiance from the surrounding geometry can be approximated as

$$L_{\text{ind}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\text{pixel}}(1 - V(\omega_i)) \frac{A_{\text{s}} \cos\theta_{\text{s}_i} \cos\theta_{\text{r}_i}}{d_i^2}$$

where d_i is the distance between $\mathbf{P}$ and occluder i (d_i is clamped to 1 to avoid singularity problems), θ_{s_i} and θ_{r_i} are the angles between the sender / receiver normal and the transmittance direction. A_{s} is the area associated to a sender patch. As an inital value for the patch area a flat surface inside the hemisphere is assumed. So the base circle is subdivided into N regions, each covering an area of $A_{\text{s}} = \pi r_{\text{max}}^2 / N$. Depending on the slope distribution inside the hemisphere, the actual value can be higher, so the algorithm uses this parameter to control the strength of the color bleeding manually. In the example in Figure 4.2, no indirect

lighting contribution is calculated for patch **A**, because it is back-facing. Patch **C** is in the negative half-space of **P**, so it does not contribute, too. Patches **B** and **D** are senders for indirect light towards **P**. Figure 4.3 shows bounces of indirect light.

4.2.3 Implementation Details

Note, that classic SSAO [Shanmugam and Arikan 2007] has similar steps and computational costs. The proposed method requires more computation to evaluate the shading model, but a similar visibility test. In the results presented, additional samples for known important light sources are used (e. g. the sun), by applying shadow maps that capture shadows from distant geometry instead of screen-space visibility. The approach uses an $M \times N$-texture to store M sets of N pre-computed low-discrepancy samples $\lambda_i \omega_i$. At runtime, every pixel uses one out of the M sets. In a final pass a geometry-sensitive blur [Segovia et al. 2006] is applied, to remove the noise which is introduced by this reduction of samples per pixel.

4.3 Multiple Pixel Values

Since SSDO are working in screen space, not every blocker or source of indirect light is visible. Figure 4.4 shows an example where the color bleeding is smoothly fading out when the source of indirect illumination becomes occluded. There are no visually disturbing artifacts, as shown in the accompanying video, but the results are biased. For a less biased solution, this section presents two approaches overcoming such limitations: *Depth peeling* (Section 4.3.2) and *additional cameras* (Section 4.3.3).

4.3.1 Single-depth Limitations

The blocker test described in the previous section is an approximation, since only the first depth value is known and information about occluded geometry is lost in a single framebuffer. Sampling points can therefore be misclassified, as shown in Figure 4.5 (left). In certain situations, the algorithm can miss a gap of incoming light or classify a blocked direction as visible. While a missed blocker (sample B) can be corrected by simply increasing the number of samples for this direction, the gap at sample A (and the indirect light coming from the vicinity of A) can not be detected from the single viewpoint, because no information about the scene behind the first depth value z_1 is available.

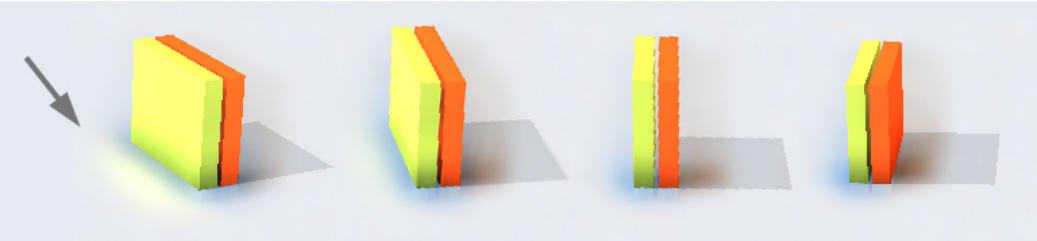

Figure 4.4: Four frames from an animation. In the first frame light is bounced from the yellow surface to the floor (see arrow). While the yellow surface becomes smaller in screen space the effect fades out smoothly.

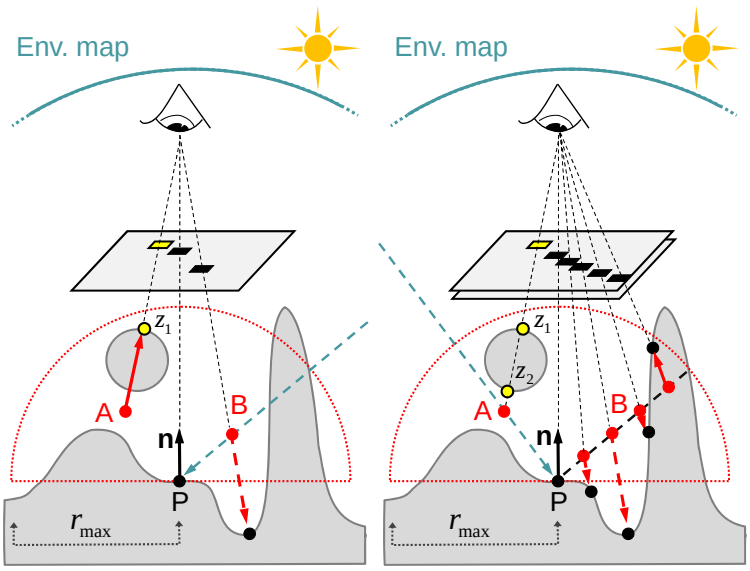

Figure 4.5: Problems with screen-space visibility (left): The visible sample **A** is classified as an occluder because its projected position is closer to the viewer. Sample **B** is above the surface, but the corresponding direction is blocked, so **P** is incorrectly illuminated from this direction. Solutions (right): Using depth peeling with two layers, sample **A** is classified as visible, because it is not between the first and second depth value. When using more samples for the direction of **B**, the occluder can be found.

4.3.2 Depth Peeling

When using depth peeling [Everitt 2001], the first n depth values are stored after n render passes for each pixel in the framebuffer. This allows us to improve the blocker test, since more information about the structure of the scene is available. Instead of just testing if a sampling point is *behind* the first depth value z_1, the approach additionally tests if the sampling point is *in front* of the second depth value z_2. When using two-manifold geometry, the first and second depth value correspond to a front - and a backface of a solid object, so a sampling point between the two faces must be inside this object (see Figure 4.5 right). To reconstruct all shadows for scenes with higher depth complexity, all pairs of consecutive depth values (third and forth depth value and so on) must be evaluated in the same way [Lischinski and Rappoport 1998]. Figure 4.6 shows screen-space shadows of a point light. For a single point light, the N samples are uniformly distributed on a line segment of length $r_{\max}$, directed towards the light source. Since there is no information about the actual *width* of the blocker from a single depth buffer, the shape of the shadow depends on $r_{\max}$. A correct shadow can only be displayed with depth peeling (avg. overhead $+30\,\%$). In addition to the visibility, color bleeding effects from backfacing or occluded geometry can be displayed with multiple layers, since we can compute the direct lighting for each depth layer.

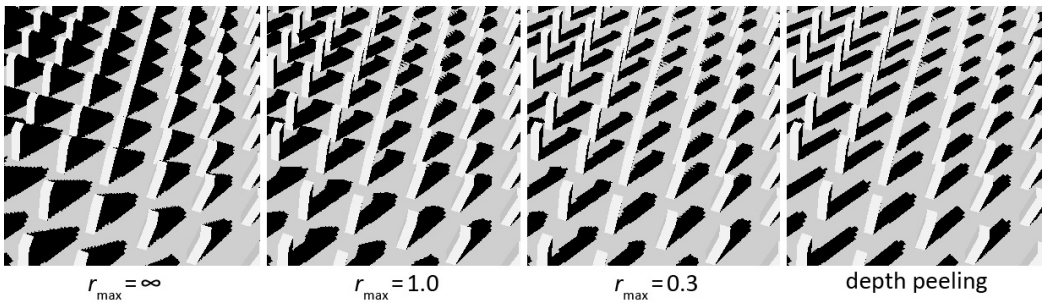

Figure 4.6: Screen-space shadows for different values of $r_{\max}$.

4.3.3 Additional Cameras

Depth peeling removes many problems related to SSDO. Alternatively, different camera positions can be used instead of different depth layers to view hidden regions. Beside gaining information about *offscreen* blockers, a different camera position can be especially useful for polygons which are viewed under a *grazing angle*. These polygons cannot faithfully be reproduced, so color bleeding from such polygons vanishes. When using an additional camera, these sources of indirect

light can become visible. The best-possible viewpoint for an additional camera would be completely different from the viewer camera, e. g. rotated about 90 degrees around the object center to view the grazing-angle polygons from the front. However, this introduces the problem of occlusion: In a typical scene many of the polygons visible in the viewer camera will be occluded by other objects in the additional camera view. Of course, this can be solved by using depth peeling for the additional camera as well. A faster solution would be a large value for the near clipping plane, adjusted to the sample radius $r_{\max}$, but this is hard to control for the whole image. As a compromise, *four* additional cameras with a *standard* depth buffer are used, all directed to the same center-of-interest as the viewer camera. The relative position of an additional camera to the viewer camera is set manually with a normalized displacement vector. To adapt to the scene size, each displacement vector is scaled with the radius of the bounding sphere of the scene.

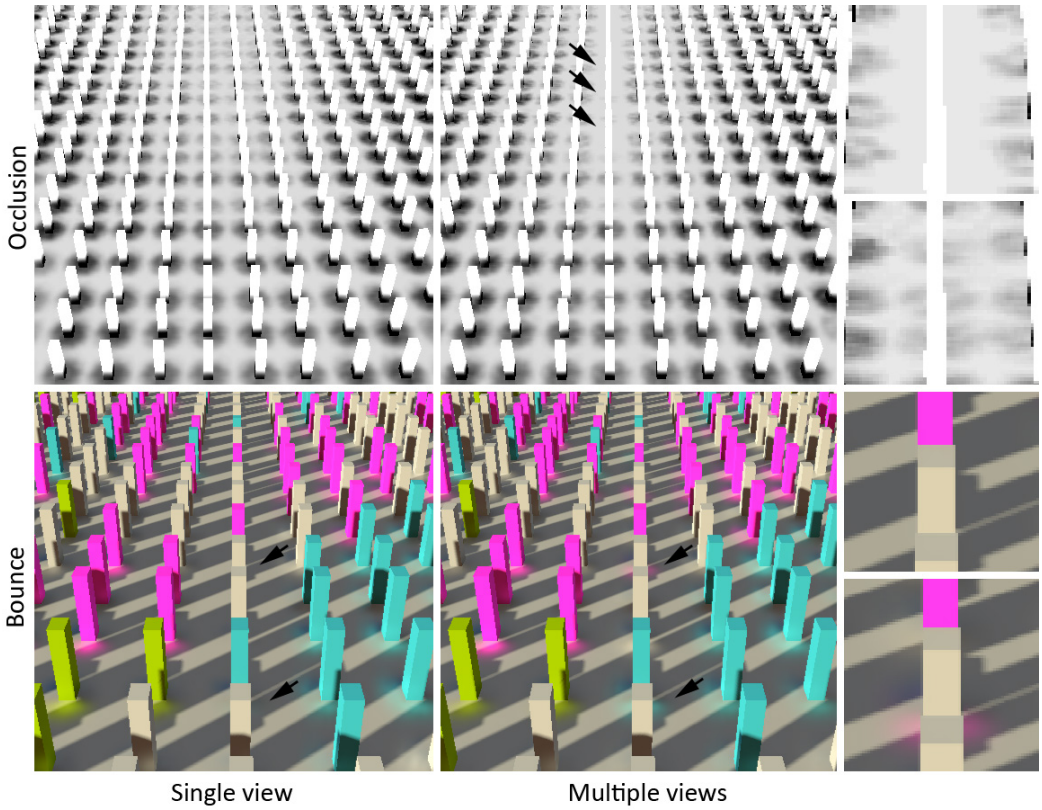

Figure 4.7: Comparing occlusion (top) and bounces (bottom) of a single view (left, 49.2 fps) with multiple views (middle, 31.5 fps). Multiple views allow surfaces occluded in the framebuffer to contribute shadows or bounces to un-occluded regions. Using multiple views, occluded objects still cast a shadow that is missed by a single view (top right). An occluded pink column behind a white column bounces light that is missed using a single view (lower right).

Using an additional camera can display color bleeding from occluded regions, offscreen objects and grazing-angle polygons (Figure 4.7 bottom). The approach uses frame-buffers with a lower resolution for the additional cameras. Therefore, memory usage and fill-rate remains unaffected. The drawback of this extension is that vertex transformation time grows linearly with the number of cameras. For a scene with simple geometry (e. g. Figure 4.7), the average overhead using four cameras is $+58\,\%$ while for large scenes (e. g. Figure 4.12, 1M polygons) it is $+160\,\%$. When using an alternative method like iso-surface ray-casting or point rendering, the number of rays resp. points per view can also be kept constant, giving multiple views the same performance as a single view.

4.4 Results

This section presents results, rendered with a 3 GHz CPU and an NVIDIA GeForce 8800 GTX.

4.4.1 Performance

SSDO works completely in image space, therefore SSDO can display directional occlusion and indirect bounces for large scenes at real-time frame-rates. Table 4.1 shows the timing values for the factory scene (Figure 4.1, 537 k polygons) using a single camera and a single depth layer. Without depth peeling or additional cameras, including directional occlusion adds only a modest amount of overhead to SSAO: $+3.6\,\%$ for DO and $+31.1\,\%$ for DO with one bounce. While adding DO effects incurs a negligible overhead, an additional diffuse bounce requires a little more computation time, assuming that the shader is bound by bandwidth and not by computation.

4.4.2 Time-Quality Tradeoff

Including depth peeling and additional cameras results in an overhead of $30\,\%$–$160\,\%$ for the test scenes used (2 depth layers or 4 cameras). The use of these extensions is a time-quality tradeoff. For environment map illumination, good results were achieved without them, since illumination comes from many directions and most of the visible errors are hidden. Figure 4.8 shows a comparison of SSDO with a ground truth path tracing image, generated with PBRT [Pharr and Humphreys 2004b].

Resolution	Samples	SSAO	SSDO	SSDO Overhead	Bounce	Bounce Overhead
1024×768	8	81.0 fps	81.3 fps	0.2 %	65.8 fps	19.3 %
	16	58.0 fps	56.5 fps	2.6 %	40.5 fps	30.2 %
1200×960	8	37.7 fps	37.5 fps	0.5 %	25.5 fps	32.4 %
	16	25.4 fps	23.6 fps	7.1 %	14.7 fps	22.2 %
1600×1200	8	24.8 fps	24.2 fps	2.4 %	15.9 fps	35.9 %
	16	16.8 fps	15.3 fps	8.9 %	9.0 fps	46.4 %

Table 4.1: Typical frame rates, using an Nvidia GeForce 8800 GTX. For SSAO a single pre-filtered environment map lookup is used, instead of one environment lookup per sample for DO. Overhead is relative to SSAO alone. The frame rate for pure rendering without any occlusion is 124 fps at 1600×1200. M is set to 4×4.

4.4.3 Animated Scenes

Since SSDO operates completely in image space, animated scenes can be displayed without any restrictions. The accompanying video shows moving colored blocks and animated objects without temporal flickering.

4.5 Integration in Global Illumination

In this section we argue, that screen-space approaches can improve the results of global illumination simulations in case of complex geometry. To avoid a time-consuming solution with the original geometry, the basic idea is to first compute the global illumination on a *coarse representation* of the geometry. Then, the lighting details can quickly be added in screen space at runtime. The idea is demonstrated for the two cases of environment map illumination and Instant Radiosity [Keller 1997], where indirect light is computed from a reflective shadow map [Dachsbacher and Stamminger 2005], resulting in a set of virtual point lights (VPLs). In contrast to previous applications (Section 4.2), in this section *all* occlusions are computed instead of only the small-scale shadows. Therefore, the indirect illumination is computed from each VPL and the indirect visibility is computed with a shadow map for each VPL. When using shadow mapping, a simplified, sampled representation of the scene (the depth map) is created for visibility queries. Additionally, a *depth bias* must be added to the depth values in the shadow map to avoid wrong self-occlusions. While this removes the self-occlusion artifacts, shadows of small surface details and contact shadows are lost,

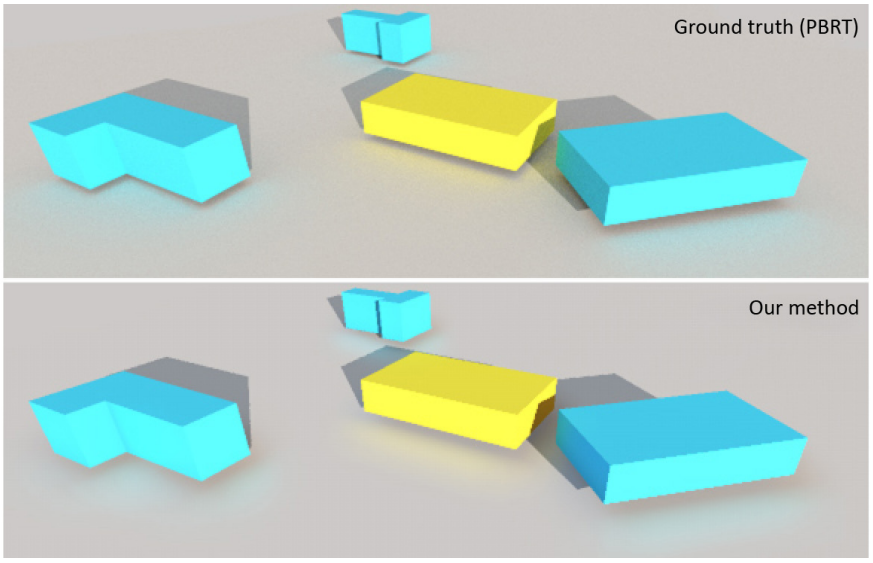

Figure 4.8: Comparison of SSDO with a one-bounce path tracing result of PBRT. The scene is illuminated by an environment map with one additional point light. Several lights and shadows are in the wrong place and appear two-dimensional. Although some haloing artifacts are introduced, the directional occlusion and indirect lighting effects present in the ground truth rendering are faithfully reproduced with SSDO.

Figure 4.9: Two animated animals. Directional occlusion and bounces are best seen on the animal's feet — without them, they would seem to float over the ground.

especially when the light source illuminates from a grazing angle. These contact shadows are perceptually important, without them, objects seem to float over the ground. Instead of applying high-resolution shadow maps in combination with sophisticated methods to eliminate the bias problems, a screen-space approach can be used to reconstruct the shadows of the small details and the contact shadows.

4.5.1 Shadow Mapping and Depth Bias

Shadow Mapping [Williams 1978] generates a depth texture from the point of view of the light source and compares the depth values stored in the texture with the real distance to the light source to decide whether a point is in shadow or not. Since each texel stores the depth value of the pixel center, we can think of a texel as a small patch, located on the surface of the geometry (see Figure 4.10). The orientation of the patches is parallel to the light source, so the upper half of each patch is above the surface. Since this part is closer to the light than the actual surface, self-occlusion artifacts appear (Figure 4.11 left). To remove this artifact, the patches must be moved completely *below* the surface. The standard solution for this is to add a depth bias $b = \frac{p}{2} \cdot \tan(\alpha)$, where p is the size of one patch in world coordinates and α is the angle between the light direction and the surface normal (see Figure 4.10). The patch size p can be computed from the distance to the light, the texture resolution and the opening angle of the spot.

Figure 4.10 shows that SSDO will not be able to display shadows of small details, e. g. at a point $\mathbf{P}$, with a shadow map. Therefore we test each framebuffer pixel, which is not occluded from the shadow map, for missing shadows in *screen space*. Since we know the amount of bias b we introduced, the basic idea is to place a few samples in this *undefined* region. More precisely, a user-defined number of samples is placed on a line segment between $\mathbf{P}$ and $\mathbf{P} + b \cdot l$, where l is a unit vector pointing to the light source. For each of the sampling points SSDO performs the same occlusion test as described in Section 4.2 and Section 4.3. If one of the sampling points is classified as an occluder for $\mathbf{P}$, the corresponding pixel in the framebuffer is shadowed. In this way, the approach adapts shadow precision to the precision of the visible occluders: As soon as an occluder becomes visible in screen space, SSDO can detect its shadow in screen space. Occluders smaller than the framebuffer resolution will not throw a shadow. Figure 4.11 shows how contact shadows can be filled in screen space.

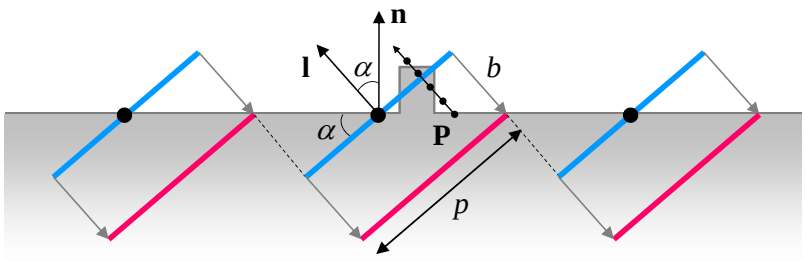

Figure 4.10: To remove the wrong self-occlusion, each shadow map texel (shown in blue) must be moved below the surface (red). This bias b depends on the slope α of the incoming light direction. Due to the coarse shadow map resolution and this bias, shadows of small scale geometry are lost in this way, for example at point $\mathbf{P}$. To reconstruct these shadows, a few samples are created on a line segment of length b, starting at $\mathbf{P}$, directed towards the light source. For each of the samples, the image-based blocker test is used.

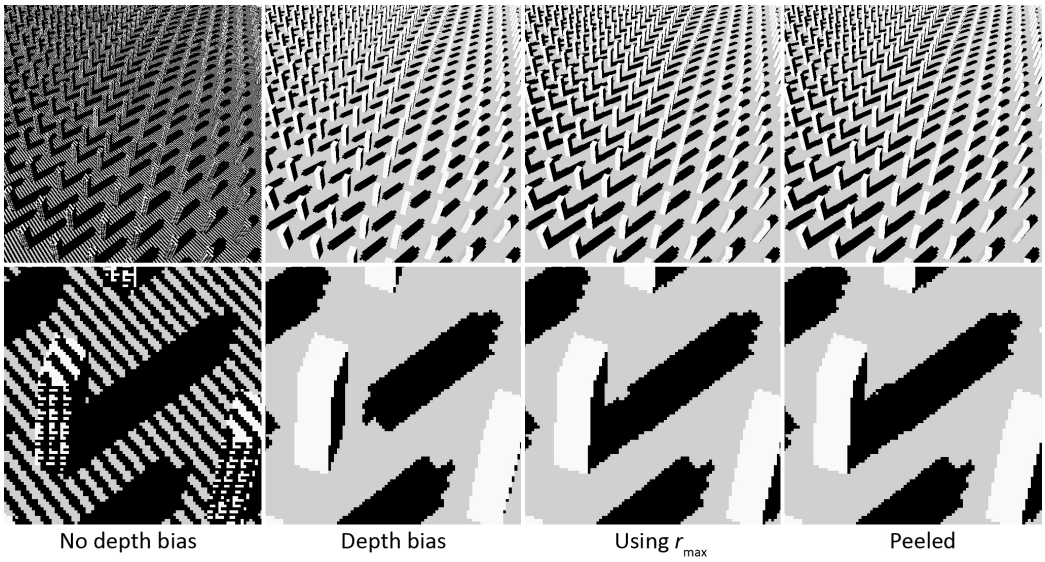

Figure 4.11: Shadows from small occluders (1024×768 framebuffer, 1024×1024 depth map), from left to right: Without depth bias, self-shadowing artifacts are visible. Classic depth bias removes the wrong self-occlusion, but shadows at contact points disappear (218 fps). Screen-space visibility removes depth bias using a single depth buffer (16 samples, 103 fps) or depth peeling (8 samples, 73 fps).

4.5.2 Global Illumination

Figure 4.13 shows how SSDO can be used for natural illumination. Here, an environment map is represented by a set of point lights with a shadow map for each light. Note the lost shadows of small details due to shadow mapping which can be seamlessly recovered in screen space. For each shadow map, 8 samples were used to detect occluders in screen space. Multiplying a simple ambient occlusion term on top of the image would not recover such illumination details correctly [Stewart and Langer 1997].

Figure 4.14 shows the integration of SSDO into an Instant Radiosity simulation. Again, missing contact shadows can be added in screen space. A related problem with Instant Radiosity is, that clamping must be used to avoid singularity artifacts for receivers close to a VPL. Such bounces of light can be added in screen space for nearby geometry, as shown in Figure 4.12. Although the estimated form-factor

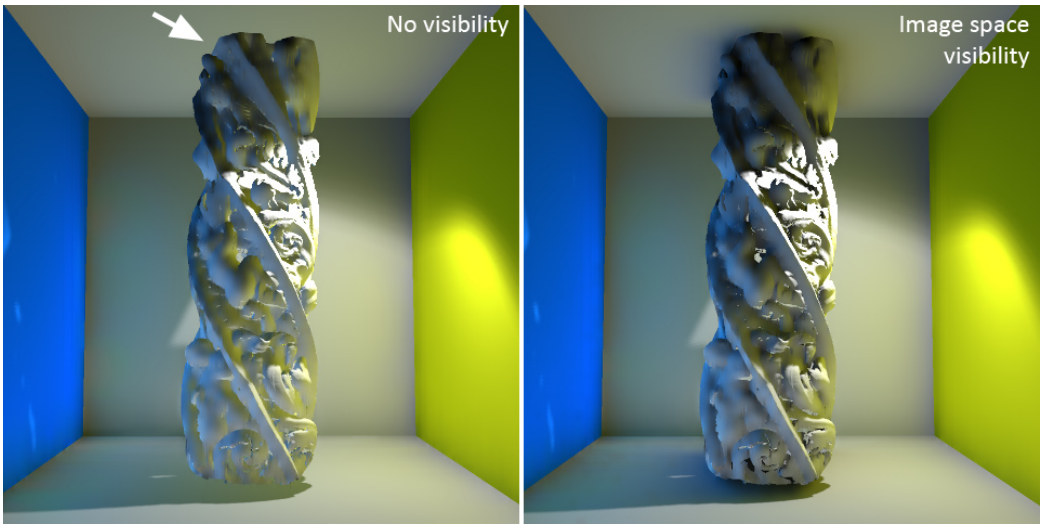

Figure 4.12: Instant Radiosity with shadow correction and an additional indirect bounce in screen space. Note the additional contact shadows and the color bleeding effects. The indirect light is slightly scaled here to make the additional effects more visible.

has a singularity too, the local density of samples is much higher than the density of the VPLs. This idea of correction in screen space can be further extended to any type of illumination which is represented from VPLs, e. g. illumination from area lights and all other approaches to visibility having a limited resolution $r_{\max}$ [Lehtinen and Kautz 2003; Ren et al. 2006][Ritschel et. al. 2008b].

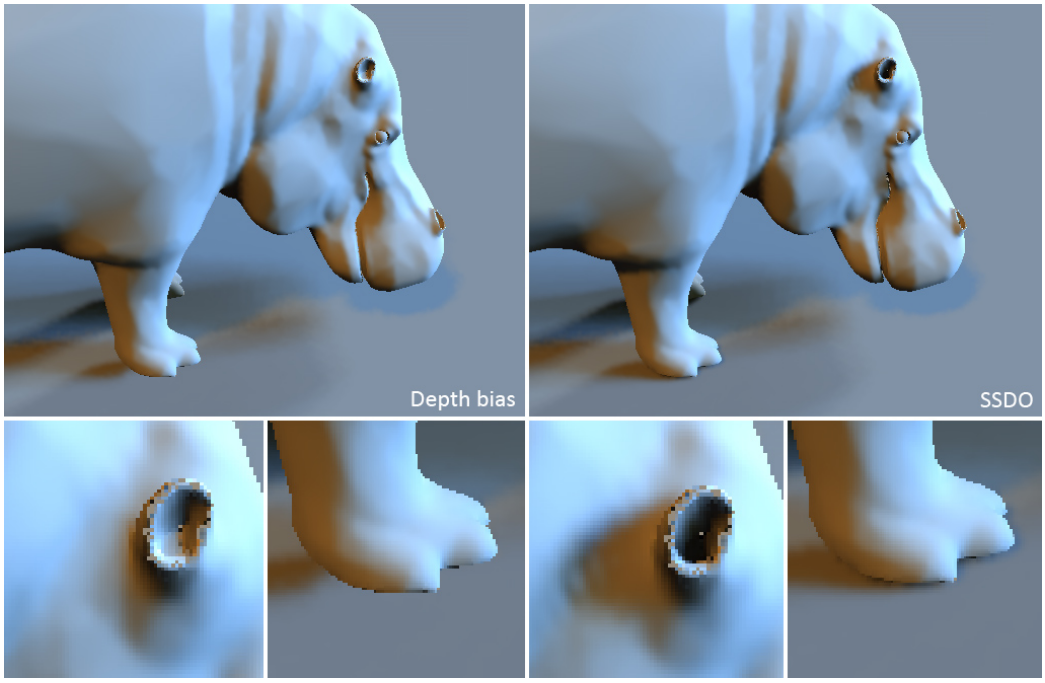

Figure 4.13: Depth bias in this natural illumination rendering (512×512, 54.0 fps) is removed using SSDO combined with shadow mapping (25.2 fps). 256 VPLs with a 512×512 depth map each are used

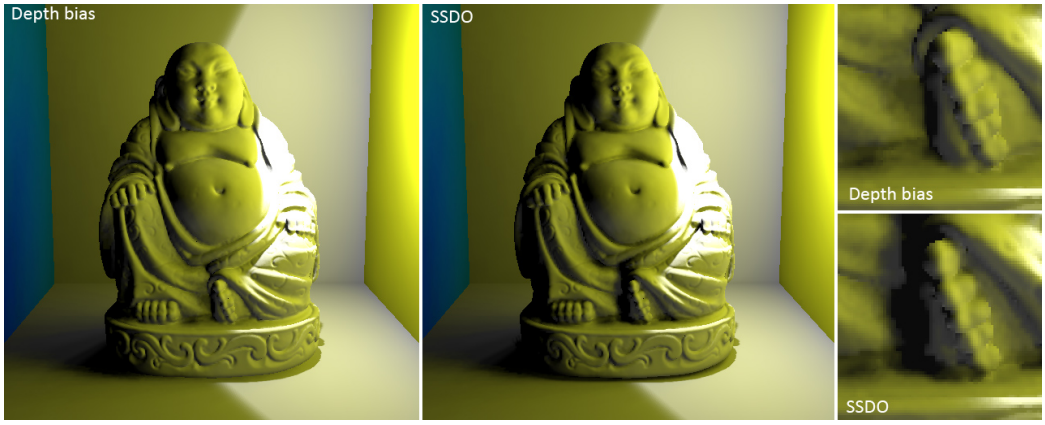

Figure 4.14: Instant Radiosity with screen-space shadow correction. Shadows of small details, like the toes, are lost due to the depth bias. These small shadows can be restored in image space.

4.6 Discussion

4.6.1 Perception

Adding bounces and DO improves the perception of meso-structures, bridging geometry and material in a consistently lit way. One problem with classic AO is, that lighting is completely ignored: everything in an occluder's vicinity is darkened equally, even if the lighting is not isotropic. In many scenarios, like skylights (Figure 4.1 and Figure 4.3), lighting is smooth but still has a strong directional component. Using classical AO under varying illumination introduces objectionable artifacts since the contact shadows remain static while the underlying shading changes. This results in an impression of dirt, because a change in reflectivity (dust, dirt, patina) is the perceptually most plausible explanation (in the sense of Helmholtz' inference from Section 2.4.2: the HVS as an inverse renderer) for such equal darkening under varying illumination. With proper DO, details in the meso-structure cast appropriate miniature shadows (softness, size, direction, color) that relate to the macro-illumination, as illustrated in Figure 4.3.

4.6.2 Quality

A screen-space approach approximates the geometry in 3D spatial proximity of pixel p with a set of nearby pixels $\mathcal{P}$ in 2D. This requires that the local geometry is sufficiently represented by the nearby pixels. By adjusting the radius $r_{\max}$ of the hemisphere, shadows from receivers of varying distance can be displayed. Using a small radius results in shadows of only tiny, nearby cavities whereas larger shadows of even distant occluders can be displayed with a large hemisphere. Figure 4.15 shows the effect of varying the size of $r_{\max}$. A smooth-step function fades out shadows from blockers close to the border of the hemisphere to avoid cracks in the shadow if a large blocker is only partially inside the hemisphere. The radius is adjusted manually for each scene, finding a good value automatically is difficult. For a closed polygonal model, an initial value for $r_{\max}$ can be found by computing the average distance between two local maxima in the geometry. With only a single depth layer, wrong shadows can appear at the edges of objects (an effect similar to depth darkening [Luft, Colditz and Deussen 2006]) if a large value is chosen for $r_{\max}$. This happens because the front most object can be misclassified as an occluder, as shown in Figure 4.6 left. While this effect enhances the appearance of still images, a sudden darkening around edges can be distracting for moving images (see the accompanying video). When reducing the radius, the width of the darkening around edges becomes smaller, but some real shadows disappear, too.

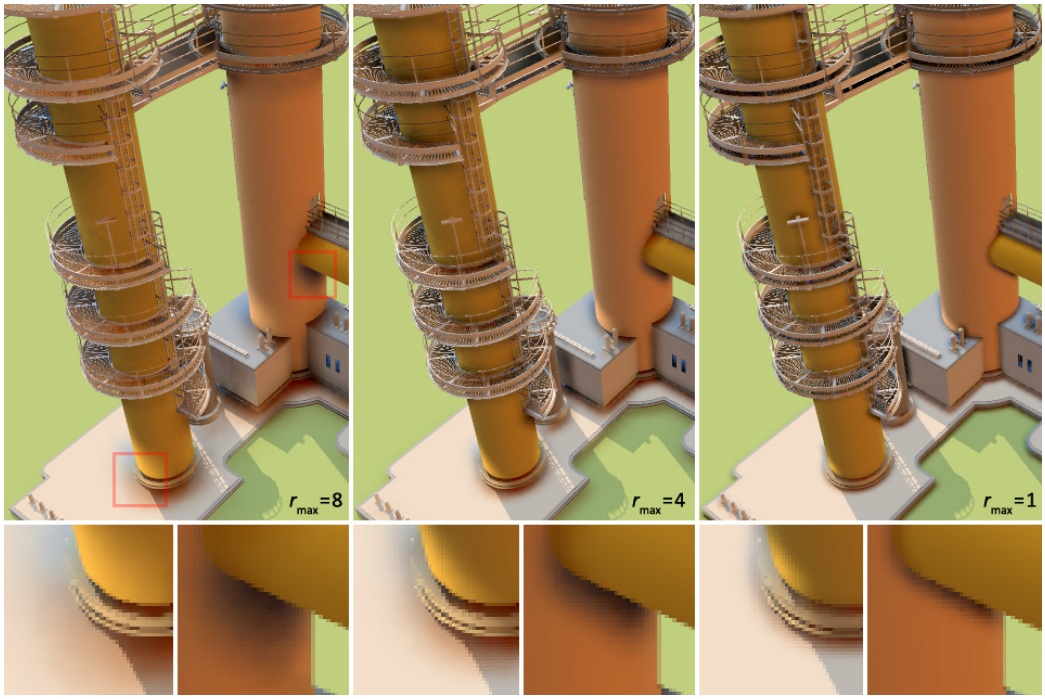

Figure 4.15: SSDO with one indirect bounce for different values of $r_{\max}$. In this case, the illumination is computed from a pointlight with a shadow map and an additional environment map. Note that using a smaller value for $r_{\max}$ decreases the size of the screen-space shadows as well as the influence region of the indirect bounce.

Depth peeling can solve this problem, but with additional rendering time. From a theoretical point of view, all shadows can be correctly displayed in screen space when depth peeling and additional cameras are used, because the scene structure is captured completely. However, a large radius and many sampling points N must be used to avoid missing small occluders, so standard techniques like shadow mapping should be preferred for distant blockers, SSDO then adds the missing small-scale details.

Besides the limitations (and solutions) discussed in Section 4.3, *sampling* and illumination *bias* affect the quality.

4.6.3 Sampling

When $\mathcal{P}$ gets small in the framebuffer, because it is far away or appears under grazing angles, the SSAO effect vanishes. Gradually increasing the distance or the angle will result in a smooth, temporally coherent fading of the additional effects. Increasing distance or increasing the angle gradually, will also give a gradual fading and will be temporally coherent. However, this issue can be the most distracting: pixels belonging to surfaces only a few pixels in size sometimes receive no bounces or occlusion at all. Still, this is an underlying problem of the original SSAO and the examples presented circumvent it by using high-resolution frame-buffers (e. g. 1600×1200).

4.6.4 Bias in Illumination

Since the algorithm is geared towards real-time display, it accepts a biased solution which is different from a physical solution. However, some of the errors can be reduced by spending more computation time. When using high-resolution depth buffers in combination with multiple cameras and depth peeling (adjusted to the depth complexity of the scene), the whole scene is captured and a correct visibility test can be computed. This means that SSDO can compute the correct illumination for a point when increasing the number of samples, the remaining error is a discretization in visibility. The bounce of indirect light is more complicated: Here *projected* positions of uniformly distributed samples are used. After the projection, the samples no longer follow a known distribution. This means that some regions have more samples while other regions suffer from undersampling. Adding more depth layers and multiple viewpoints can make every source of indirect light visible, but the sampling distribution is still non-uniform. Additionally, the approach only uses an approximated form factor. Visible artifacts were not observed from these

sources of error. An unbiased solution for indirect light in screen space has to be considered future work (Section 9.3.1).

In conclusion, SSDO does not introduce any additional (temporal) artifacts. In the worst the approach fades into traditional SSAO which again is equivalent to no AO at all in the worst case.

The distant and bounced incident radiance has to be sufficiently smooth, like the skylight in Figure 4.1. High-frequency environment maps require either pre-smoothing or higher sampling rates (e. g. at direct shadow boundaries) but this was not observed in practice.

5

Imperfect Shadow Maps

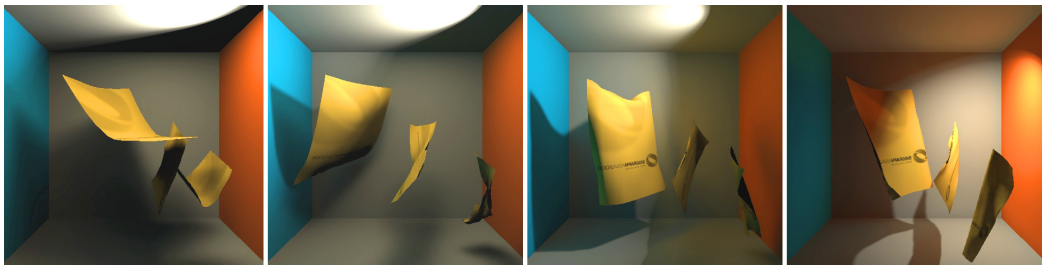

Figure 5.1: Global illumination for a completely dynamic scene (light, view, geometry, material) rendered at 19 fps on an NVIDIA GeForce 8800 GTX. The scene is illuminated with a small spot light (upper right); all other illumination and shadowing is indirect (one bounce).

5.1 Introduction

Global illumination effects, such as indirect illumination, are known to be perceptually important [Stokes et al. 2004] but are often omitted or coarsely approximated due to their high rendering cost – especially in interactive applications (cf. Section 3.1). One of the most expensive components when computing indirect illumination is visibility determination, i. e., determining if two points are mutually visible or not. Commonly, this is done accurately using ray-casting or other similar visibility queries, despite anecdotal evidence that visibility does not need to be accurate when computing indirect illumination [Sillion and Drettakis 1995; Arikan, Forsyth and O'Brien 2005], (cf. Section 3.1.2).

This chapter addresses the challenge of costly visibility with *imperfect shadow maps* (ISMs) in order to enable real-time global illumination of large and fully

dynamic scenes (cf. Figure 5.1). ISMs are low-resolution shadow maps that may contain inaccurate visibility information, i. e., some of the stored depths may be incorrect. Nonetheless, ISMs can be combined with *Instant Radiosity* (IR) [Keller 1997], (cf. Section 3.1.3) to compute convincing indirect illumination. IR represents indirect lighting with a set of virtual point lights (VPLs) for which visibility is usually determined accurately, either using shadow volumes [Keller 1997; Crow 1977] or high-resolution shadow maps [Williams 1978], constituting the main bottleneck.

While shadow maps can be precomputed for static scenes and even partially recomputed on-the-fly to handle light changes [Laine et al. 2007], the recomputation of all shadow maps (or volumes) for a dynamic scene is no longer possible at interactive speed, even for modest-sized scenes. In contrast, this chapter shows that hundreds of ISMs can be computed very efficiently on current GPUs and that they can replace accurate visibility, therefore enabling IR-based real-time indirect illumination of fully dynamic scenes.

In contrast to PRT (cf. Section 3.1.4), ISMs enable real-time global illumination of fully dynamic scenes through the use of approximate visibility without being restricted to low-frequency illumination or small scenes. In fact, the technique scales to scenes with several hundred thousand polygons.

Previous work exists to speed up visibility queries itself, e. g. by accelerating ray-tracing [Wald 2004]. In contrast, the goal of this chapter is to explore the perceptual influence of approximate visibility and use the gained insights for efficient rendering.

The goals for this technique are similar to many of the perceptually-motivated rendering algorithms mentioned in Section 3.1.6: Using visibility approximations without perceptually impacting the resulting images to speed up costly global illumination.

Despite the inaccuracies contained in ISMs and the limitations inherent to IR, the resulting renderings are very similar to reference renderings but rendered at ten times the original speed.

5.2 Imperfect Shadow Maps

Visibility tests between pairs of 3D points are often the most time-consuming part in global illumination algorithms. These queries can be made more coherent using algorithms that compute indirect illumination based on virtual point lights (VPLs) [Keller 1997; Wald et al. 2002], where visibility needs to be computed

only between a small number of VPLs (several hundred) and all other scene points instead of many arbitrary pairs of scene points. A good choice for solving these coherent visibility queries is the use of shadow mapping [Williams 1978]. However, computing several hundred shadow maps per frame is too slow for even moderately-sized scenes and does not scale well with scene complexity [Laine et al. 2007].

The goal is to achieve real-time frame rates for large and dynamic scenes, which necessitates a method that can quickly solve visibility queries for hundreds of dynamically generated VPLs. Realizing that fully accurate visibility queries are not necessary for computing indirect illumination, the use of imperfect shadow maps (ISMs) is proposed. ISMs are low-resolution shadow maps that are computed from a crudely simplified point-representation of the scene, followed by a pull-push pass to fill holes. While ISMs may contain incorrect depth values, the resulting errors in the indirect illumination are small but the computational gains are significant, which will be demonstrated in detail in Section 5.4.

5.2.1 Scene Preprocessing

In a preprocessing step, the 3D scene is approximated by a set of points with roughly uniform density. Each point is created by randomly selecting a triangle with probability proportional to the area of the triangle, and then picking a random location on the triangle. More elaborate schemes such as point repulsion can be used but did not prove to be necessary. For each point, its barycentric coordinates are stored relative to its triangle as well as the triangle index in order to support dynamic scenes without the need to recompute the point representation. The barycentric coordinates also allow to retrieve the normal and reflectance for each point, which will be required for secondary indirect light bounces. The number of points P to represent the scene is chosen by the user.

5.2.2 ISM Creation

A single ISM is created by splatting the point representation into the depth-buffer (with a box splatting kernel, i. e., GL points), where the size of a point splat is based on its squared distance to the corresponding VPL position (assuming each point sample represents the same average area). When an ISM is used for computing indirect illumination, it needs to cover a full hemisphere of depth information (VPLs emit light over a hemisphere) for which parabolic maps [Brabec, Annen and Seidel 2002] are used. In general, ISMs can be used with other common shadow map projections, such as orthographic or perspective projection. As indicated before, dynamic scenes can easily be handled via deforming the points according

to the deformation of the corresponding triangles. While this leads to non-uniform point distributions, the results do not suffer as the technique does not depend on accurate uniform distributions.

Many low-resolution ISMs are created and rendered in one pass and stored in a single, large texture. To this end, a vertex shader splits up the incoming stream of points representing the scene and distributes an equal amount of points to each ISM within the large texture. Each ISM receives a fixed, random subset of the point set. Typically, parabolic ISMs are used with a resolution of 128×128 pixels, stored in a 4096×4096 texture.

In order to reduce computation, a sparse set of points is used, which may leave holes in the depth map. The technique therefore uses a pull-push approach [Grossman and Dally 1998; Marroquim, Kraus and Cavalcanti 2007] to fill these holes and reconstruct a sensible depth map. The first step is the creation of an image pyramid in the pull phase where the image is down-sampled by a factor of two in each stage. Only valid pixels in the finer level are used for averaging the pixels in the coarser level. The second step is the push phase where the holes are filled top-down by interpolating the pixels from the coarser level to approximate the undefined pixels in the finer level. Inspired by Marroquim, Kraus and Cavalcanti [2007], outliers are rejected via thresholding in both the pull and push phase. In particular, the hole filling only combines depth values during the pull phase that are close to each other, and only replaces depth values during the push phase that are far from the coarse depth values that are pushed down. Separate depth thresholds for the pull and the push phase are used, which are scaled by 2^l according to the mip-map level l (0 being the finest level); both threshold values are typically set to 5 % of the scene extent. In all results, virtually all holes are filled, by going up only two levels in the pyramid.

5.2.3 Discussion

Given an effective resolution of $N_{\text{res}} = (\pi/4)\, n \times n$ for a single parabolic ISM and P points, an average of $\bar{P} = P/N_{\text{res}}$ points land on every ISM pixel. Assuming optimal point sampling, it is possible to find the correct depth if the scene has an average depth complexity of $\bar{P}$. In reality, the samples will be less optimal and only a much smaller depth complexity can be resolved [Wand et al. 2001]. Pull-push hole filling mitigates this; however, a fully correct depth map can never be expected due to the randomness of the point samples.

Figure 5.2 demonstrates this: shadows from a single point light are rendered with traditional shadow maps as well as ISMs. In this case, artifacts are visible when using an ISM. The pull-push pass does improve the result considerably, yet

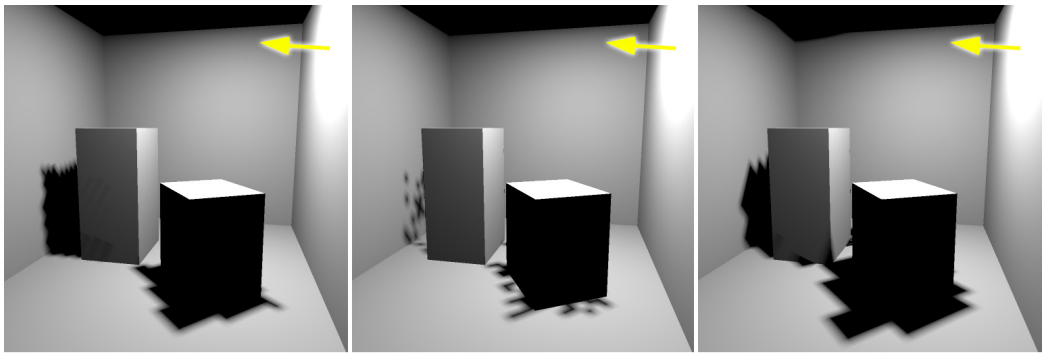

Figure 5.2: Comparison of a classic shadow map and an imperfect shadow map with and without pull-push. The ISM contains holes and incorrect depth values. Most of these pixels are corrected after the pull-push operation. Note that a single ISM is not expected to produce an accurate shadow. However, shadows from many ISMs average out and artifacts disappear, which is ideal for rendering indirect illumination.

differences to the reference shadows are still visible. However, these differences do not necessarily result in noticeable artifacts for indirect illumination, as the contributions of many ISMs are summed up, e. g., see Figure 5.1. Examples of ISMs with and without pull-push can be found in Figure 5.3 on the right.

Using a simple point-based geometry representation is desirable, as it can be easily used to create hundreds of ISMs in parallel and it offers straight forward support for dynamic scenes. Alternative representations were also considered to render imperfect shadow maps, such as more advanced point representations [Wand et al. 2001; Dachsbacher, Vogelgsang and Stamminger 2003] or polygonal level-of-detail [Hoppe 1996] but these either require location-dependent evaluation, which is costly for hundreds of ISMs, or cannot easily handle complex geometry, such as fences or trees.

Additionally, position-dependent representations of the scene can be used, like sequential point trees [Dachsbacher, Vogelgsang and Stamminger 2003] or combined approaches where large objects are represented by polygons and smaller, detailed objects are replaced by a point set.

5.3 Indirect Illumination with ISMs

As mentioned earlier, the indirect illumination in this work is based on the idea of instant radiosity [Keller 1997], where virtual point lights are distributed stochastically along light paths which then act as sender of indirect light. Summing up the

contribution of all VPLs, while taking into account shadowing, yields the resulting indirect illumination.

The current approach uses imperfect shadow maps to determine visibility for each VPL, as they are highly efficient – hundreds of ISMs can be rendered in a single pass – and their rendering cost is only loosely dependent on the overall polygon count. Since the first bounce of light is the most important [Stokes et al. 2004], this chapter concentrates on one-bounce indirect light and create the virtual point lights from the light source completely on the GPU, similar to Dachsbacher and Stamminger [Dachsbacher and Stamminger 2006] (single bounce for now). The 3D position of each VPL is determined by rendering a cube map from the viewpoint of the (direct) point light source, on which importance sampling is performed by building the necessary cumulative and marginal distributions from which N_{VPL} VPLs are selected. Of course, other methods that generate VPLs, such as photon shooting [Keller 1997], could also be used. Each VPL creates a parabolic ISM, oriented along the normal of the surface capturing the whole hemisphere in a single view.

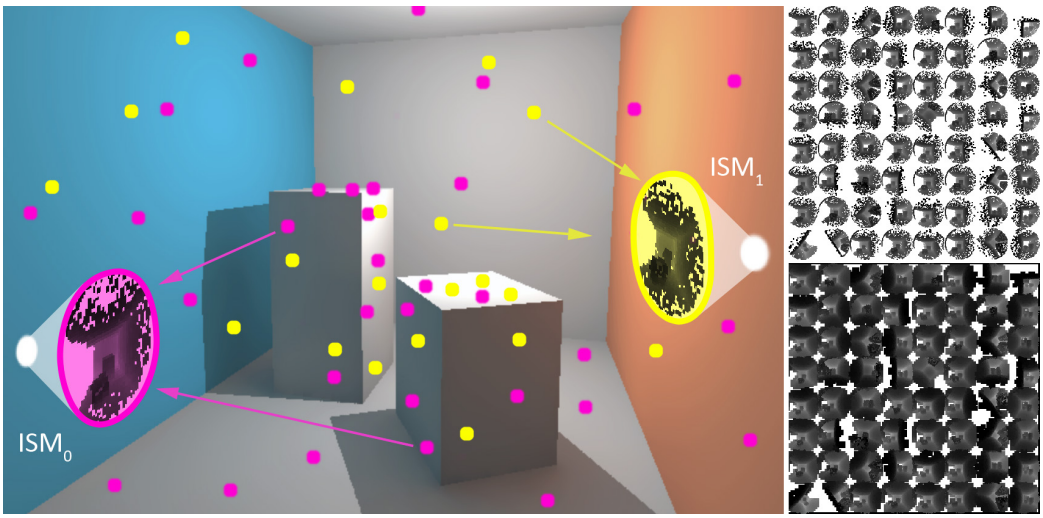

Figure 5.3: Global Illumination with imperfect shadow maps: The indirect light in this didactic scene is represented by two VPLs. Each VPL generates a shadow map using a point-based representation of the scene. Since coarse visibility information is sufficient, each VPL uses only a sparse subset of the points (encoded as colors) to generate an imperfect shadow map at real-time rates. The contribution of all VPLs is summed up yielding the final indirect illumination. Examples of imperfect shadow maps (bottom with and top without pull-push) are shown on the right; dark value signify small depth values and light values represent large depths.

As described before, only a subset of all the points is used for each ISM and all ISMs are stored in one large texture (see Figure 5.3). To further reduce the rendering cost, a G-Buffer is used to efficiently gather from N_{VPL} VPLs [Segovia et al. 2006]

during rendering of the final image. In other words, the algorithm only gathers from a subset of VPLs at each pixel and performs a geometry-aware blur to remove noise.

Like most of the instant radiosity methods, this VPL-based illumination restricts the approach to diffuse and slightly glossy materials, unless a very large number of VPLs are used. Possible extensions to non-diffuse materials are discussed in section Section 9.1.1.

5.3.1 Multiple Bounces

For multiple bounces, a contribution of this chapter is, to generalize ISMs to *Imperfect Reflective Shadow Maps* (IRSM) in the same way classic shadow maps generalize to reflective shadow maps [Dachsbacher and Stamminger 2005]. This allows to generate VPLs for each additional bounce directly on the GPU, which are then rendered with ISMs. Given the VPLs for the first bounce, generating VPLs for the second bounce works as follows: the point-based representation of the scene is used to create imperfect reflective shadow maps for each of the initial VPLs; i. e., instead of rendering the shaded geometry *shaded points* are rendered into the IRSMs. This results in a large texture which contains all paraboloid maps of the first bounce VPLs, but each paraboloid now stores its own indirect illumination (instead of depth values). Importance sampling is then applied to this texture in order to generate the second bounce VPLs (similar to the first bounce VPL generation from the direct light cube map). The position and normal information extracted from the points is used to position and orient the second bounce VPLs onto the geometry. The illumination of these second-bounce VPLs is then rendered with imperfect shadow maps much like the illumination from the first-bounce VPLs. The same principle can be applied to render further bounces.

5.4 Results

5.4.1 Numerical Analysis

This section presents results rendered at real-time rates with ISMs on an 1.8 GHz CPU with an NVIDIA GeForce 8800 GTX. As mentioned before, all scene components can be fully dynamic (geometry, materials, and lights). Unless mentioned otherwise, all results are rendered with a single-bounce of indirect illumination at 640×480 pixels and a G-Buffer block size of 8×8 pixels.

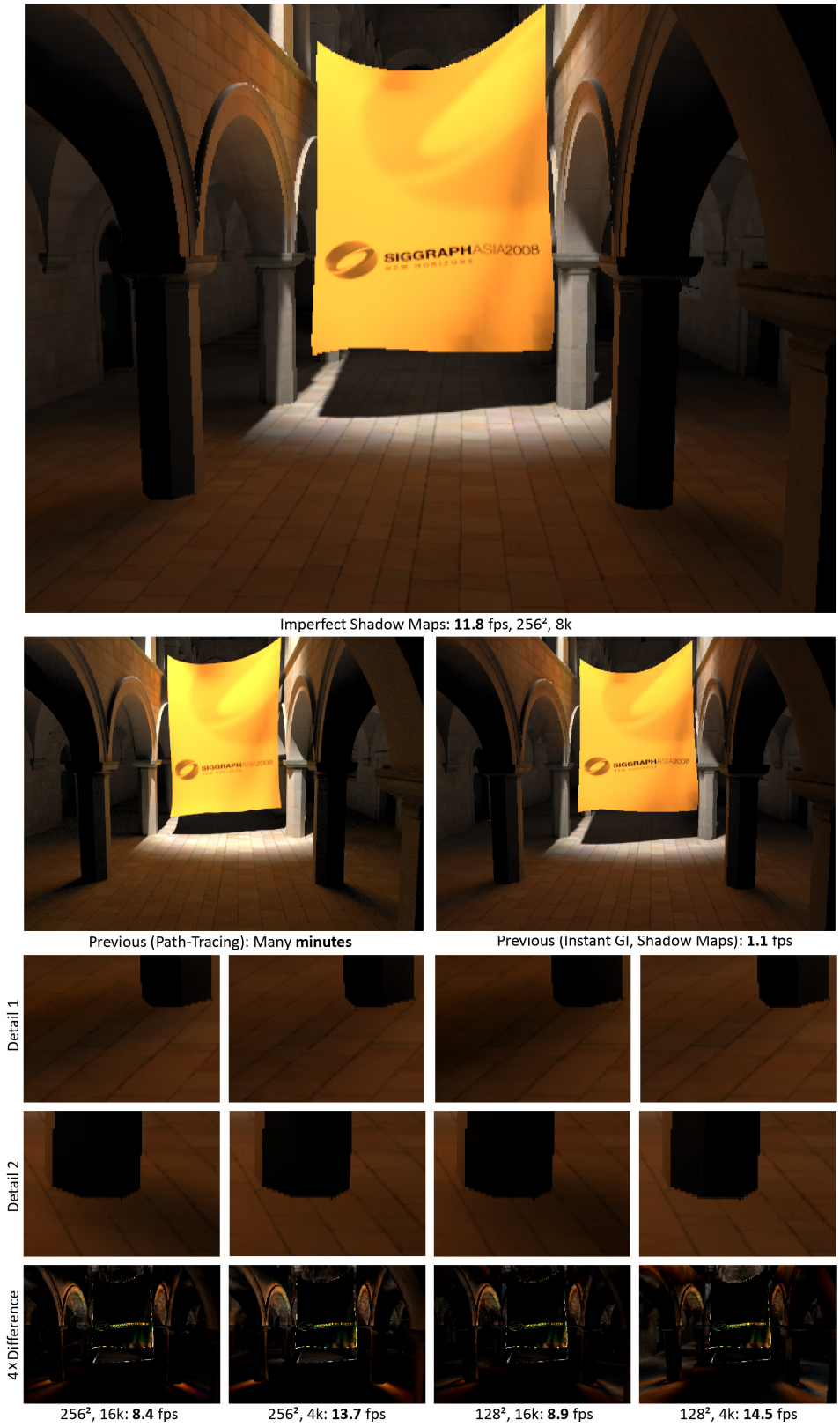

Figure 5.4: The first scene (Sponza, 70 k faces) achieves good results for all settings, only the indirect shadows are a bit smoother overall.

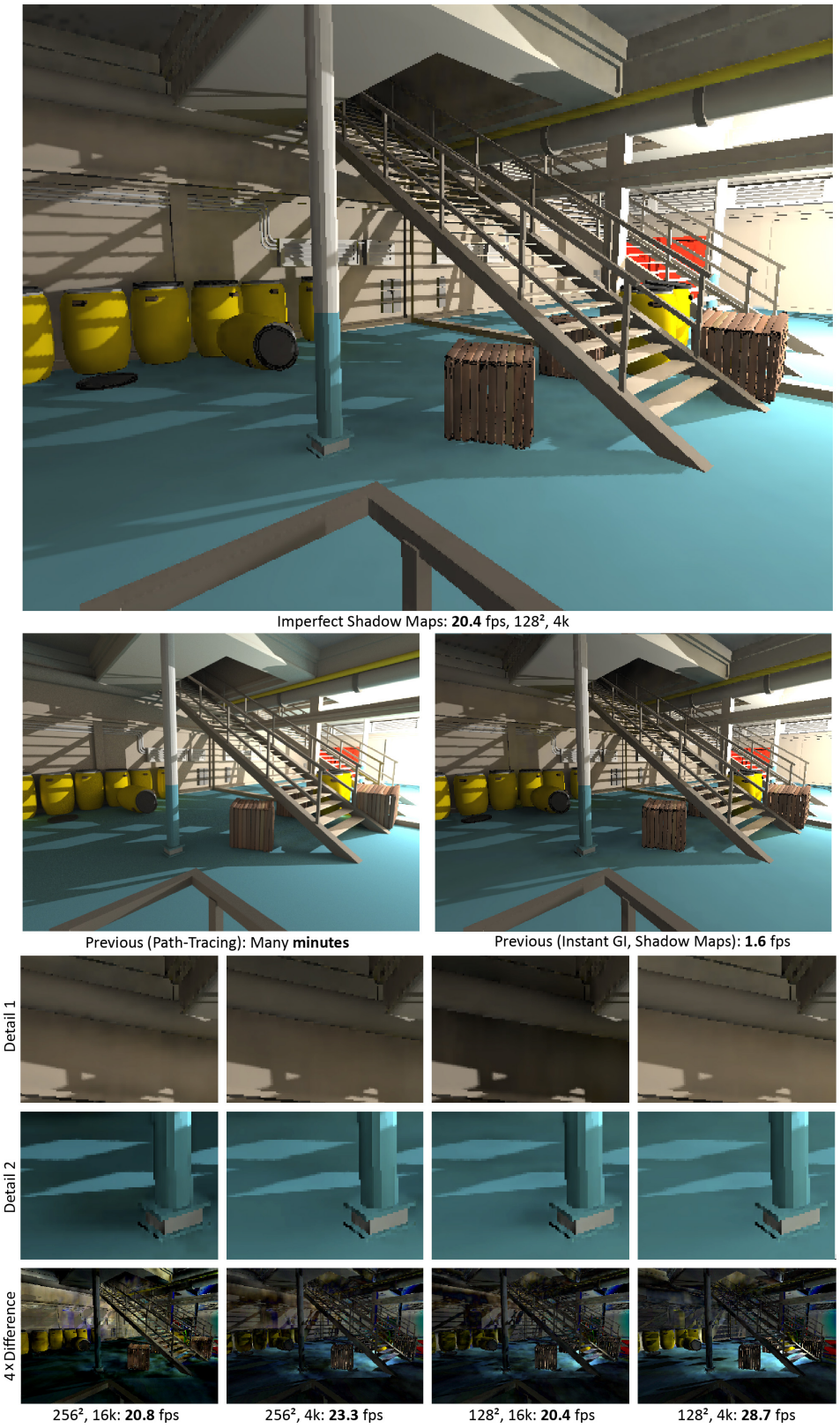

Figure 5.5: The industry scene (76 k faces) requires larger ISMs and more points to faithfully reproduce shadows from thin objects, such as the columns.

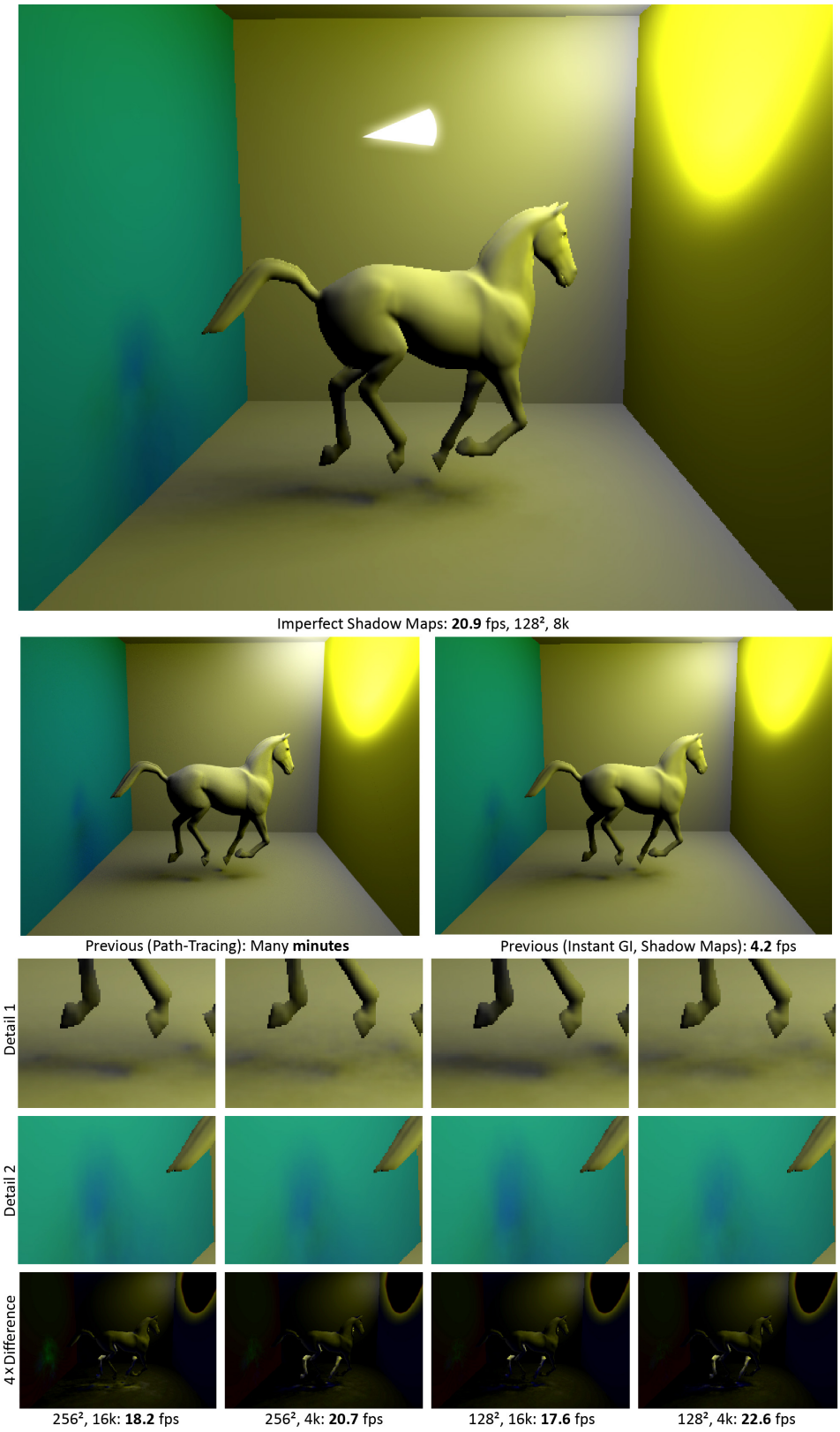

Figure 5.6: The horse scene (17 k faces) is close to the reference for all settings. A small part of the leg is consistently too bright, which is due to shadow biasing.

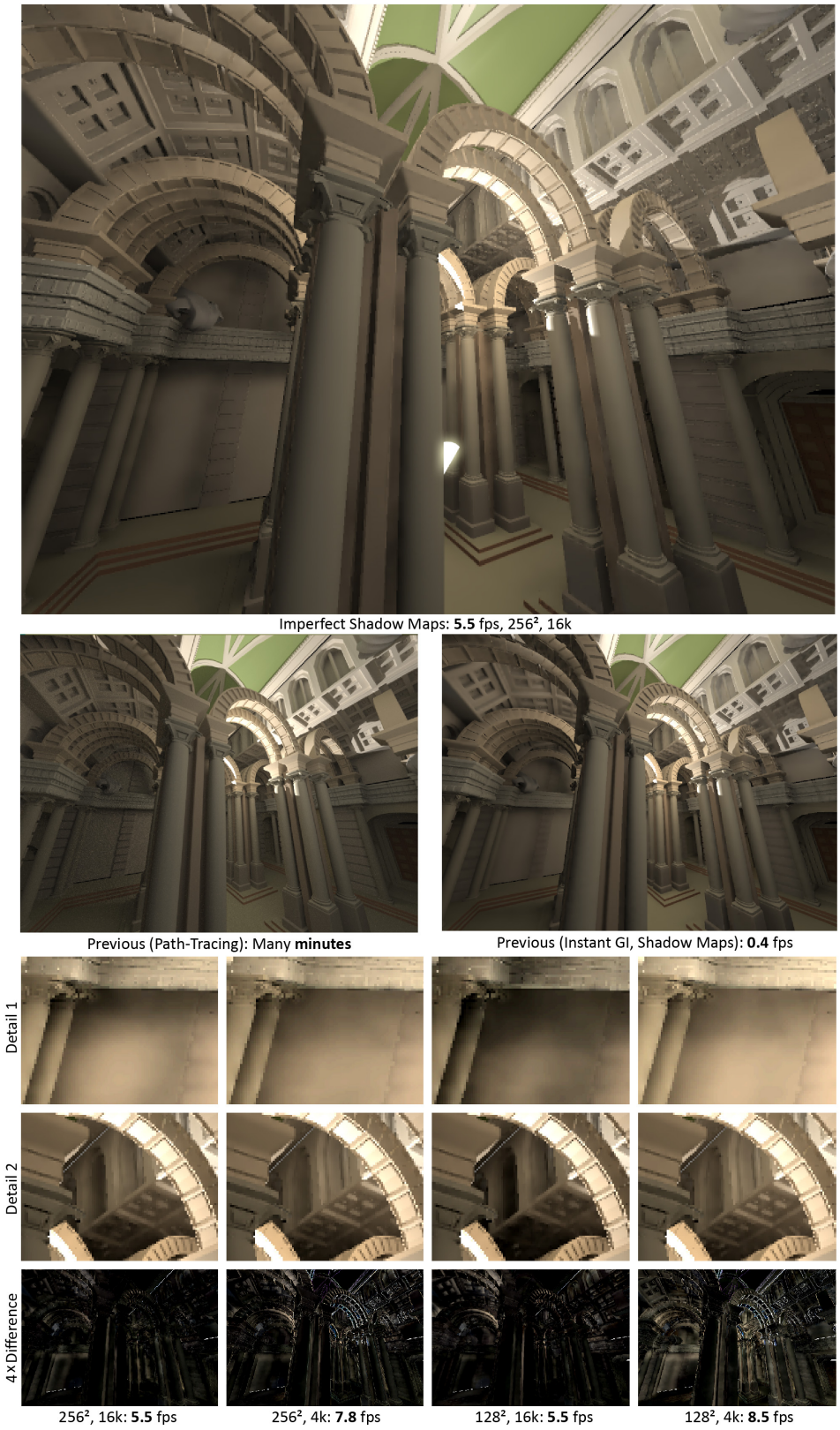

Figure 5.7: In the last scene (Palazzo, 380 k faces), very faithful renderings can be achieved with the highest settings. Frame rates are generally lower due to the additional second indirect bounce (another 256 VPLs) and higher image resolution (1280×960 to prevent aliasing).

Figure 5.4–5.7 show four scenes rendered with ISMs (one- or two-bounce, diffuse indirect illumination). The first three scenes are rendered with single-bounce indirect illumination, whereas the last scene with two bounces. ISM resolution as well as the number of point samples are varied (resolution of 256×256 and 128×128; 16000 and 4000 point samples per VPL); all scenes use 1024 VPLs. We compare to a reference GPU solution (using classic 256×256 shadow maps) as well as a reference image computed using path tracing (using the same number of bounces). Cutouts of each scene and difference images (4 times magnified) with respect to the GPU reference are depicted for each parameter set.

Generally speaking, the resolution of the ISMs influences the penumbra size that can be resolved (e. g., see column in second scene). Further, ISMs produces results that are very similar to the reference GPU solution but is an order of magnitude faster. Not surprisingly, using more point samples increases the quality of the resulting indirect illumination but also decreases the overall frame rates. Overall, our results are very similar to the GPU reference renderings, but are rendered at about 10 times the frame rate. However, there are some differences. ISMs produce indirect shadows that are bit softer than with classic shadow maps (e. g., see first scene). This is to be expected and is not a visually distracting difference. Indirect shadows of small or thin objects require a sufficient number of points to be resolved (see column of second scene with 4000 vs. 16000 points). Finally, if the ISMs' resolution is insufficient, indirect shadow areas can become overly dark or bright, e. g., see the close-ups of the last scene.

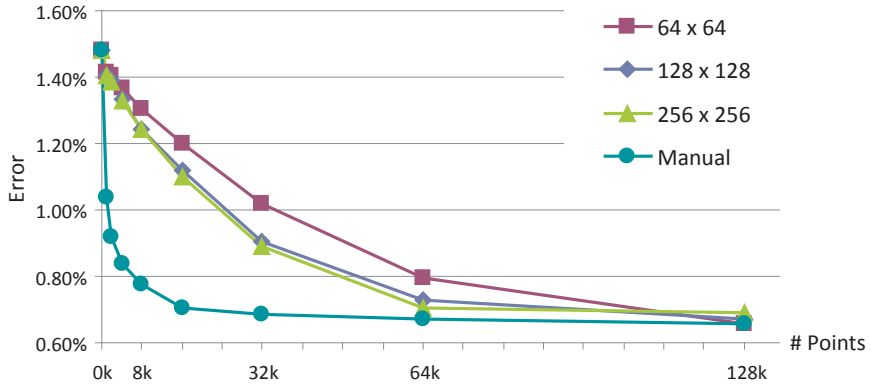

Figure 5.8: Image error (normalized RMS) of the Sponza scene with respect to the reference GPU solution. The number of point samples as well as ISM resolution is varied. The error goes down by increasing either of them. Manual adjustment of all parameters (ISM resolution, point samples, splat size, etc.) yields the lowest overall error.

Figure 5.8 shows the normalized root-mean-square error with respect to the GPU reference for the Sponza scene for three different ISM resolutions as well as a varying number of points, where all other parameters are fixed (512 VPLs). As

expected, based on the observations from Figure 5.4–5.7, more point samples reduce the error. Increasing ISM resolution decreases the error as well; however, the difference in error between 128×128 and 256×256 is small. The manual curve demonstrates that adjusting parameters by hand (ISM resolution, manually modified splat size, G-Buffer setting) can achieve high fidelity with a small number of point samples. Note that the error does not fully vanish due to the G-Buffer.

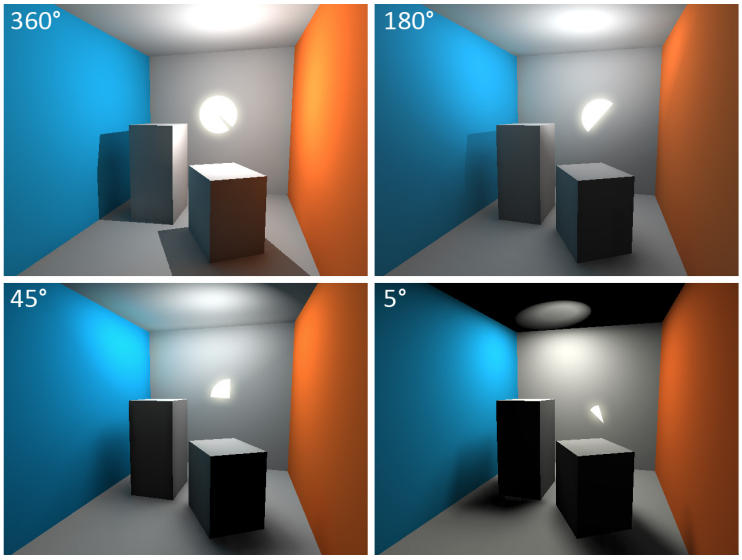

Figure 5.9: Spotlight with angles between 360 ° and 5 °. Small spot lights create high-frequency indirect shadows, which are difficult to reproduce with low-resolution ISMs. Normal illumination scenarios can be reproduced faithfully.

Figure 5.9 illustrates how the size of a spot light influences the quality of the result. A large spot light only creates very soft indirect shadows, as much of the scene is illuminated directly. In the contrived case, where a small spot light only illuminates a tiny fraction of the scene, high-frequency indirect shadows are created, which the low-resolution ISMs are not able to reproduce faithfully. However, ISMs are a valid technique for common illumination scenarios.

Figure 5.10 shows how the pull-push pass effectively removes small light leaks. The additional overhead is small and worth the cost. In comparison to classic shadow maps, ISMs can be created more than 30 times faster for this particular scene (using 256 VPLs and an ISM resolution of 256×256), even with pull-push. Furthermore, even complex scenes with fine geometric detail can be handled with ISMs and pull-push, see Figure 5.11.

The generalization to multiple bounces using imperfect reflective shadow maps is illustrated in Figure 5.12. The Cornell box is shown with one, two and three indirect bounces. Again, the scene is fully dynamic, geometry, view, lighting, and

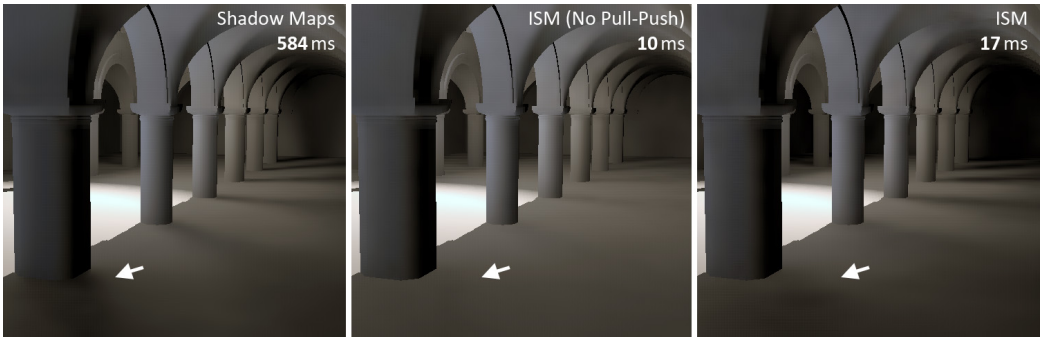

Figure 5.10: Using ISMs without pull-push often creates light leaks. Here, the columns do not cast a dark enough (indirect) shadow. Using pull-push largely removes these artifacts.

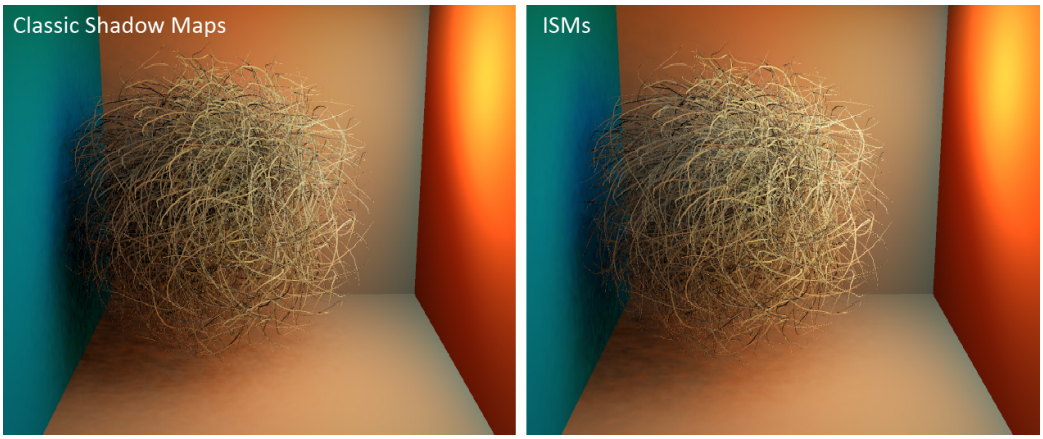

Figure 5.11: ISMs can handle geometrically complex scenes. Left: Classic Shadow Maps. Right: ISMs (1024 VPLs, 256×256, and 4 k points).

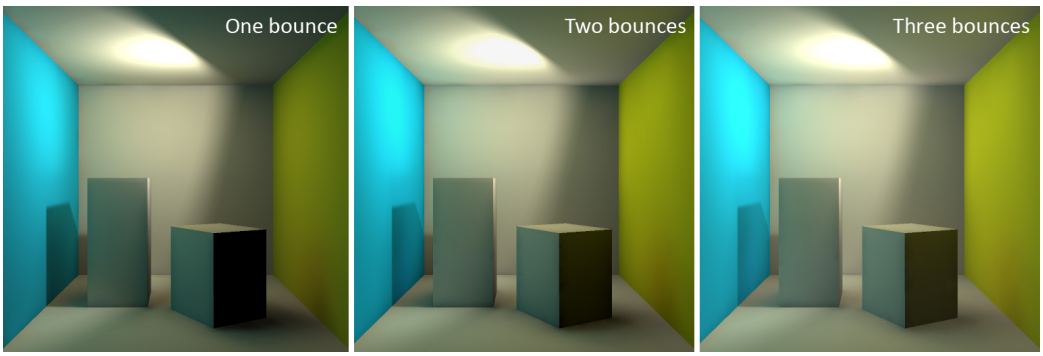

Figure 5.12: From left to right: One, two and three bounces of indirect illumination rendered at 1024×1024 with 14.0, 7.5, and 5.5 fps respectively (using 1024 VPLs for the first bounce and 256 VPLs each for the second and third).

materials can be changed on-the-fly. Nonetheless, it is possible to render multiple indirect bounces at interactive rates.

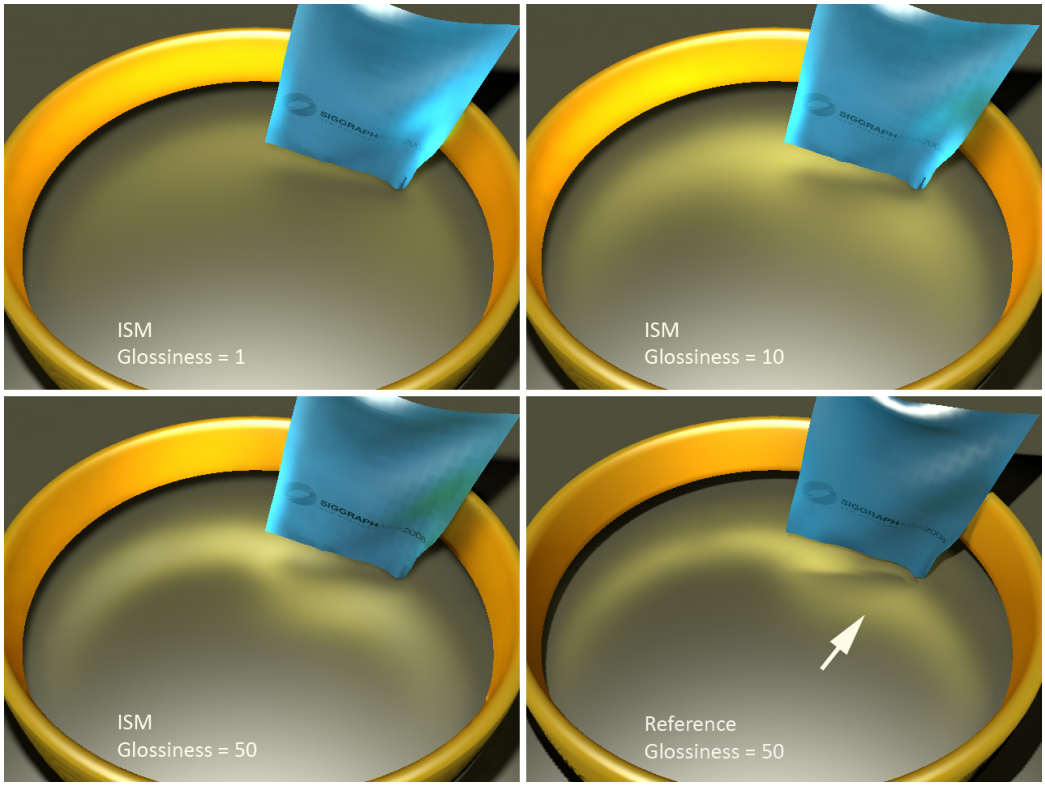

Figure 5.13: A deforming cloth inside the classic metal ring with one bounce at 13 fps. The ring's BRDF varies from slightly glossy (Phong with exponent of N=1) to more specular (exponent of N=50). Note how the glossy bounce casts shadows that significantly contribute to the appearance. 4096 VPLs, 64×64 ISMs, and 2 000 points are used in each. The reference rendering (converged IR solution) shows that some of the high-frequency effects, such as crisp indirect shadows, cannot be reproduced with low-resolution ISMs.

The method can handle diffuse as well as glossy indirect illumination. Figure 5.13 demonstrates rings of varying glossiness with a diffuse cloth occluding some of the light that is reflected from the ring. While there are faster dedicated methods for rendering caustics, such as Dachsbacher and Stamminger [2006] or Wyman [2008], they do not allow for occluded glossy indirect illumination of dynamic objects.

Temporal coherence is generally very high. However, temporal flickering can occur if an insufficient number of VPLs is used, much like the original instant radiosity method [Keller 1997]. This is further exacerbated when too few point samples are used. However, for all test scenes ISMs were able to achieve temporally coherent results at interactive rates (10 fps or more).

For a typical scene (Sponza with 256×256 ISM resolution and 8 k points in this

case), the rendering times can be broken down as follows: 7 ms for the reflective shadow map including importance sampling (1024 VPLs), 44 ms for creating the ISMs, 8 ms for the pull-push pass, 15 ms to render the scene using ISMs (16 VPLs per pixel), 4 ms for the G-Buffer blur, and 11 ms for the direct light. Less complex

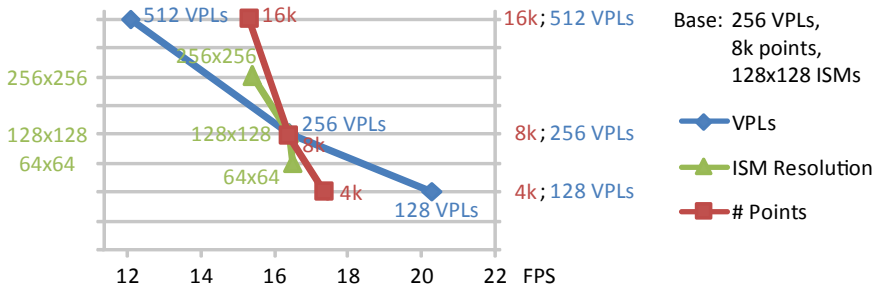

Figure 5.14: Performance scalability of the Sponza scene with respect to number of VPLs, number of point samples and shadow map resolution. One parameter at a time is varied and the achieved frame rate is graphed.

scenes require fewer VPLs, and the computation time goes down accordingly; see the blue curve in Figure 5.14, where the number of VPLs is varied for the Sponza scene. The number of points used has a direct influence on the amount of time it takes to create the ISMs, see the red curve in Figure 5.14, which is to be expected. However, note that ISMs scales well with polygon count: an increase in scene tessellation does not increase the cost to compute indirect illumination, which is decoupled through the use of a point representation; only the cost of direct illumination increases. Increasing the resolution of the shadow maps has little impact on rendering performance (green curve in Figure 5.14). Of course, larger ISMs require more point samples in order to actually improve rendering quality. The rendering performance can be kept almost constant for increasing screen resolutions by increasing the G-Buffer block size accordingly (see Table 5.1). This also keeps the quality approximately equivalent, whereas fixing the G-Buffer setting effectively increases the rendering quality for larger screen resolutions.

As these numbers illustrate, there is no single performance bottleneck in the algorithm. In fact, the method is specifically designed to eliminate the geometry and render setup bottleneck that was common to previous VPL-based methods.

5.4.2 User Study

Figure 5.4–5.7 compare images rendered with the technique to reference renderings. However, images do not allow to judge differences that might only become apparent in video sequences, such as temporal artifacts. To this end, we created short

Resolution	Same G-Buffer FPS	Same G-Buffer G-Buffer	Same Quality FPS	Same Quality G-Buffer
640×480	15.1 fps	4×4	15.1 fps	4×4
800×600	13.2 fps	4×4	17.0 fps	8×8
1024×768	10.7 fps	4×4	17.3 fps	8×8
1280×1024	8.6 fps	4×4	15.2 fps	16×16
1600×1280	6.3 fps	4×4	14.7 fps	16×16
1920×1600	4.6 fps	4×4	12.1 fps	16×16

Table 5.1: Varying screen size for Sponza scene (128×128 and 4 k points), while either keeping G-Buffer settings or quality fixed.

video sequences of the first three scenes from Figure 5.4–5.7 with the same ISM resolution and number of points (4 versions each, all using pull-push).

In a simple study, 12 participants were asked to judge the similarity to the reference GPU video sequences in a side-by-side comparison (on a scale from *1 – not similar* to *5 – extremely similar*).

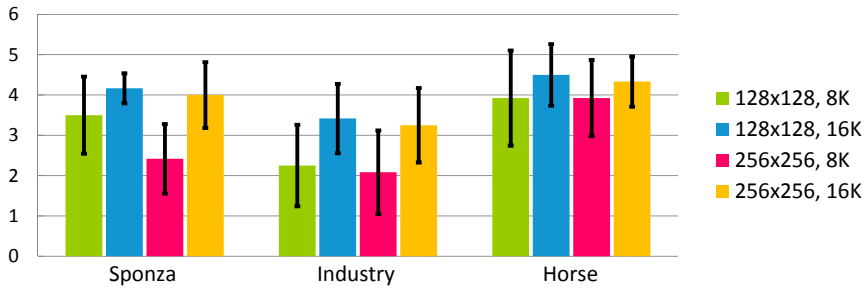

Figure 5.15: Result of the proposed perceptual study, where participants rated similarity of short video sequences to reference sequences. The short video sequences were rendered with different parameters (Figure 5.4–5.7).

Generally speaking, using more point samples leads to better similarity ratings, see Figure 5.15. This is especially true in the Cornell industry scene, where some flickering artifacts are noticeable with a small number of point samples. In less complex scenes, like the horse example, similarity is high even for a small number of point samples. Increasing the resolution did not have much influence on the perceived quality. In summary, these findings are in line with the results in Figure 5.4–5.7.

5.4.3 Other Applications of ISMs

Here, we describe how imperfect shadow maps can be used in other applications, such as area ligths, environment maps.

Area Lights

ISMs can also be used as a method to render soft shadows from (textured) area lights of complex shape. Figure 5.16 shows an animated scene with a glossy BRDF

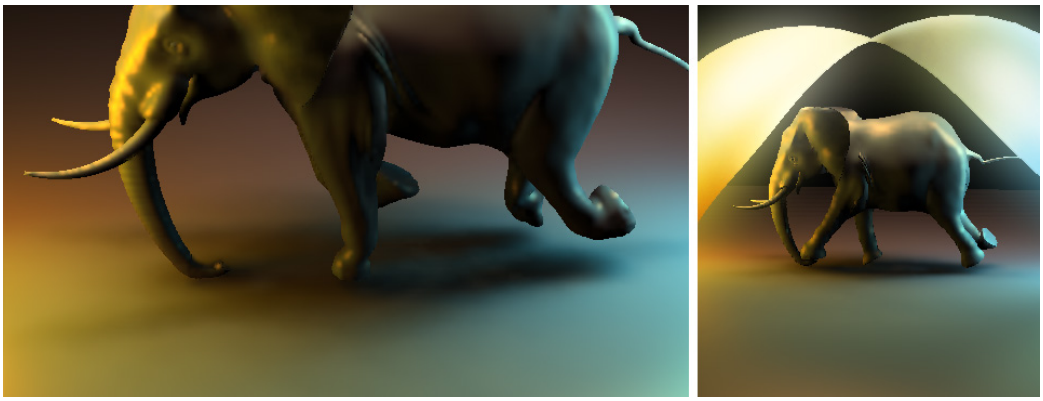

Figure 5.16: A curved area light (right) illuminating an elephant (85 k faces) is approximated with 512 point lights and then rendered with ISMs at 17 fps (256×256 and 8 k points each).

illuminated by a multi-colored, curved area light, which was approximated with 512 point lights. The shadows from each of these point lights are computed with ISMs. The resulting rendering does not exhibit noticeable artifacts, despite each individual ISM being inaccurate. While there are techniques solely targeted at rendering rectangular area lights which provide higher performance, e. g. Guennebaud, Barthe and Paulin [2006], ISMs enable textured light sources of complex shape as well as the integration of the BRDF with the incident lighting yielding glossy reflections.

Environment Maps

The same idea can be used to render direct illumination from environment maps. The environment map was approximated using 1024 VPLs (again using importance sampling [Pharr and Humphreys 2004a]) and use ISMs to compute the shadows. Figure 5.17 shows an example where a glossy object is illuminated by an environment map, rendered both with classic shadow maps and ISMs. There are only minor differences between the two, demonstrating that ISMs can be a useful

Figure 5.17: Depth map computation time for direct environment map illumination from 1024 VPLs using a model of 525 k faces with classic and imperfect shadow maps (256×256 and 8 k points each).

technique for environment map lighting of fully dynamic scenes including glossy reflections. There are other techniques that enable the efficient evaluation of direct illumination from environment maps for dynamic scenes [Annen et. al. 2008]; however, ISMs allow for the correct integration of the BRDF with the environment map lighting.

5.4.4 Discussion

While ISMs are a powerful method to speed up the computation of indirect illumination, there are some limitations. In the current implementation, reflective shadow maps (RSMs) are used to generate VPLs [Dachsbacher and Stamminger 2005], restricting the approach to point and spotlight illumination. However, other techniques to create VPLs can be used to lift this restriction.

The technique is not limited by the total polygon count of the scene; however, good results require a sufficient number of point samples, which are currently chosen by hand. Scenes with a high depth complexity require more samples, otherwise the ISMs cannot contain meaningful information. For very large scenes with many different rooms, portal-based techniques could be used to break up the point representation into smaller chunks but this remains future work.

In order to maintain high frame rates, low-resolution ISMs are used. As a consequence, indirect shadows from small geometry cannot be resolved (e. g., see thin

columns in Figure 5.4). This is usually not crucial if the direct illumination covers large parts of a scene yielding only very low-frequency indirect shadows in the first place.

As ISMs are based on instant radiosity, it inherits some limitations. A sufficient number of VPLs need to be created, or the indirect illumination is prone to artifacts such as temporal flickering. Light leaking due to shadow biasing may appear, as can be seen in the horse example in Figure 5.6. In general though, failure cases can be prevented by increasing the ISM resolution and the number of VPLs and point samples as well as adjusting shadow biasing parameters.

The method is not fully automatic – parameters need to be chosen by the user, such as the number of point samples or depth bias. It is possible to provide some rough guidelines on how to set these parameters. In all (diffuse) test scenes, 1024 VPLs have been used and provide good quality and very little flickering. Fewer VPLs would even suffice for less complex scenes. Empirically, we have found that it is sufficient to use $P \approx {}^1\!/_4 N_{\text{res}} \times N_{\text{VPL}}$ point samples for scenes with N_{VPL} VPLs (and an equivalent number of ISMs) and a modest depth complexity. Further, an ISM resolution of 128×128 or 256×256 is generally sufficient, see Figure 5.8.

6

Micro-Rendering

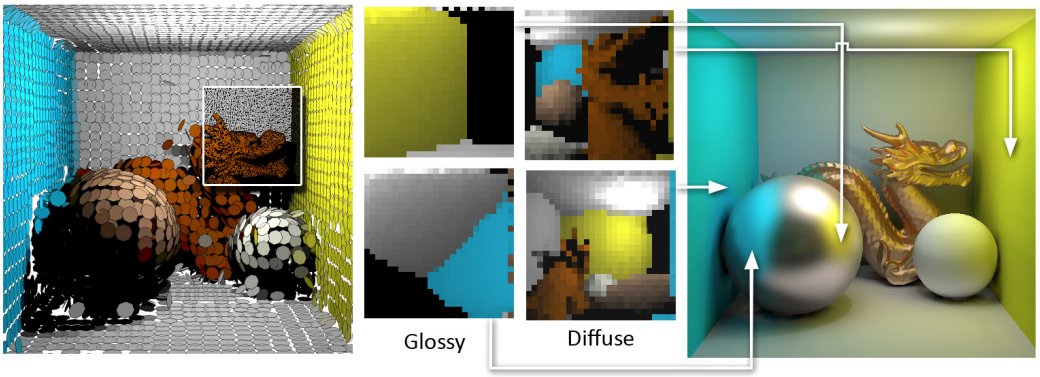

Figure 6.1: Micro-Rendering computes global illumination by rasterizing many thousands of tiny micro-buffers (middle) in parallel, using a sub-linear point rendering technique with an importance-warped projection. Two interior levels of the hierarchy with 1M points are shown on the left. The dynamic Cornell box with glossy objects (1 M points, right) renders at 1.1 Hz (512×512 resolution).

6.1 Introduction

High-quality global illumination at interactive speed is a difficult challenge, especially in complex and fully dynamic scenes. On the one hand, conventional methods for off-line rendering produce photo-realistic results, but do not allow interactivity. On the other hand, available interactive or real-time techniques often harness the computational power available in modern GPUs, but suffer from various limitations, such as static geometry [Sloan et. al. 2002], precomputed movement [Dachsbacher et al. 2007], or low-frequency lighting [Sloan et. al. 2007].

In principle, a global illumination solution – at least for one indirect bounce – can be computed easily: render the directly illuminated scene from every visible point (either using rasterization or ray tracing) and convolve it with the BRDF. While this method, commonly called final gathering [Ward, Rubinstein and Clear 1988] (cf. Section 3.1.8) is conceptually simple, it is quite expensive, and most GPU-based global illumination techniques try to avoid it, often leading to rather intricate methods. We return to the final gathering idea and propose a GPU-based rasterization approach that turns it into a highly efficient interactive technique. Micro-Rendering itself is independent of the cache placement and interpolation strategy [Ward, Rubinstein and Clear 1988; Ward and Heckbert 1992; Gautron et al. 2005; Křivánek et al. 2005; Gassenbauer, Křivánek and Bouatouch 2009].

Imperfect shadow maps [Ritschel et. al. 2008b], (cf. Chapter 5) achieve interactive frame rates for moderately complex and fully dynamic scenes using approximate visibility, but ultimately fail to handle large scenes due to a non-hierarchical point representation. Further, indirect shadows are generally smoothed out considerably. In contrast, Micro-Rendering enables high-quality, indirect illumination for both diffuse and glossy scenes of high geometric complexity.

In contrast to Lightcuts [Walter et. al. 2005], (cf. Section 3.1.3) Micro-Rendering uses a hierarchy of point samples, which are rasterized in parallel into micro-buffers at less cost (roughly $O(1)$ due to depth buffering), while enabling more accurate visibility. The main difference is the cost per cut element. With Lightcuts a ray is traced to each light with a cost of around $O(\log m)$ in scenes with m geometric elements. In contrast, Micro-Rendering does not separate lights and geometry and the cost for each splat from the cut is roughly $O(1)$ due to Z-buffering to resolve visibility. Lightcuts computes visibility only towards the center of each node, which can lead to artifacts, e. g., banding, if the nodes' solid angles is large. As a remedy, Multidimensional Lightcuts [Walter et. al. 2006] proposes to reduce banding artifacts by using multiple representatives. In contrast, Micro-Rendering evaluates visibility for every pixel of a large node, hence preventing artifacts.

Different from Christensen [2008], Micro-Rendering exploits GPU compute power and employs importance sampling for arbitrary BRDFs – both for rasterization and on-demand raycasting.

In concurrent work, Wang et. al. [2009b] demonstrate interactive global illumination using GPU-based final gathering with ray-tracing. It enables complex lighting effects but relies on sparse gathering locations for efficiency.

Many rendering techniques share the goal of evaluating the rendering equation more densely, when one of the factors inside the integral is high, and less densely everywhere else. This goal is also the inspiration of Micro-Rendering but in contrast

to many other techniques it evaluates everything on the fly. Traditional importance sampling in ray tracing [Dutré, Bekaert and Bala 2003] is probably the best-known method that tries to focus computation on where it is most needed. It has usually been coupled with ray tracing, since visibility needs to be checked in arbitrary directions. Micro-Rendering makes this idea amenable to GPU-based rendering by rasterizing the hierarchy of points into warped micro-buffers, effectively performing importance sampling.

Jensen [1995] uses an importance-warped table (demonstrated for diffuse BRDFs) to roughly record directions that contribute most to the reflected radiance. This is computed from nearby photons in a photon map only, thus eventually containing holes, but sufficiently accurate to improve importance sampling for path tracing. Micro-Rendering uses a BRDF-warped micro-buffer to record all incident lighting, while ensuring that no holes occur and that the integral over the micro-buffer is equivalent to the lighting convolved with the BRDF.

There are two main challenges Micro-Rendering has to face. First, the costly standard triangle-based rasterization needs to be avoided as it is inefficient when executed many times (e. g., for every surface point), and second, importance sampling is vital to reduce unnecessary computation but is difficult to incorporate on the GPU.

This chapter introduces *Micro-Rendering* to address these issues: At the core, the approach performs final gathering of the incident illumination at a large number of visible surface points. This final gathering technique operates in parallel on the GPU, and rasterizes the scene into a large number of *micro-buffers* by traversing a hierarchical point-based representation of it. The micro-buffers employ a non-linear projective mapping to allow for importance sampling of the BRDFs. Convolution of the incident illumination with the BRDF is a simple sum of each micro-buffer's content due to importance sampling. For high-quality results, Micro-Rendering performs final gathering at every image location. Faster renderings can be achieved by performing final gathering at a subset of all image locations followed by bilateral upsampling [Sloan et. al. 2007].

This chapter demonstrates the proposed method for complex, fully dynamic scenes with indirect, diffuse and glossy illumination, for final gathering from photon maps, and for radiosity-style global illumination.

6.1.1 Overview

The method described, allows the rendering of global illumination in fully dynamic scenes. It is scalable such that the user can trade rendering quality for speed,

with a smooth transition ranging from fast previews to solutions that are close to ground truth. At the heart of the contribution is an efficient Micro-Rendering technique which performs a BRDF importance sampled final gathering of the incident radiance. The term “micro” refers to the low overhead of launching the rendering, as well as to the low resolution of the frame buffer. It is shown, that the method runs an order of magnitude faster than previous approaches and reaches preview quality at interactive speeds of up to 10 frames per second.

The key points of the proposed algorithm are:

- A hierarchical point-based representation of the scene’s surfaces is generated for adaptive level-of-detail rendering.
- The novel Micro-Rendering technique facilitates a highly parallel rendering of arbitrary (in the present case hemispherical) mappings on the GPU, in order to gather the incident radiance at many surface points at the same time.
- Importance sampling is integrated into the Micro-Rendering allowing us to efficiently compute final gathering for diffuse and glossy surfaces with arbitrary BRDFs.
- For preview quality final gathering is computed at a subset of the image pixels and use bilateral upsampling [Sloan et. al. 2007]. High-quality renderings at approximately 0.5 to 1 frames per second perform final gathering at every pixel (512×512 res.).

Micro-Rendering is beneficial for many different global illumination methods. It directly renders one-bounce indirect illumination when point samples are directly lit, and multiple bounces when used with instant radiosity techniques. It can also be used to compute and display radiosity solutions, and for interactive walkthroughs of photon mapping results with final gathering.

6.2 Scalable, Parallel Final Gathering

In this section all steps of the Micro-Rendering method are described in detail, starting with the basic technique, which is then extended with importance sampling and bilateral upsampling.

6.2.1 Hierarchical Point-Based Representation

Micro-Rendering is based on a hierarchical point-based representation of the scene, since point-based representations allow for efficient level-of-detail rendering on GPUs [Dachsbacher, Vogelgsang and Stamminger 2003],[Ritschel et. al. 2008b] and Chapter 5, often provide a simple point selection criterion [Rusinkiewicz and Levoy 2000].

In comparison to triangle-based rendering, point-based rendering has a lower setup and rasterization cost for low image resolutions and can be efficiently rasterized when using non-linear projections.

Similar to QSplat [Rusinkiewicz and Levoy 2000], Micro-Rendering uses a hierarchy of bounding spheres, where leaf nodes represent a single surface element (an oriented disc with radius r) and interior nodes represent a collection of surface elements. For rendering, this hierarchy is traversed starting from the root node: each node's bounding sphere is projected into micro screen space to compare its size to a given threshold. This determines if the respective disc is rendered and the traversal is terminated, or if its child-nodes are to be tested recursively. The approach replaces the "size-in-screen-space"-test of the original QSplat method with a test based on the solid-angle subtended by a node. This criterion will allow us to define *warped projective mappings*, and thus to integrate importance sampling easily. Note that warped (non-linear) mappings would be difficult to combine with triangle rasterization.

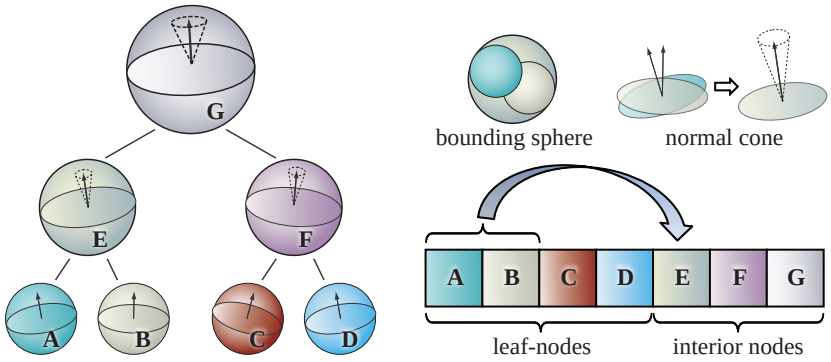

Figure 6.2: The scene's surfaces is represented using a point hierarchy, similar to QSplat, which is stored as a complete binary tree. This allows for easy traversal and fast updates at run-time.

Point Hierarchy Generation

The point-based representation of the scene is generated in an offline preprocessing step. First, random points are created on the triangles of the scene, proportional

to the triangle areas, using a best candidate sampling. For every point sample, we store the triangle index and the barycentric coordinates of the point relative to its triangle. The barycentric coordinates allow us to recompute positions and normals for deforming geometry [Ritschel et. al. 2008b] (see Chapter 5). The point density, which is initially constant, determines the radius of the point samples. Under deformations the points' radii are scaled to compensate for the varying point densities.

These point samples form the leaf nodes of the hierarchical point representation (Figure 6.2). The hierarchy is build by computing a binary-space partitioning of the point samples, which is stored as a complete binary tree (hence the number of points n is a power of two). This enables us to compute skip-pointers on-the-fly during traversal, instead of storing additional offsets. The construction first sorts the leaf nodes and works as follows: the list of leaf nodes is taken (the initially created point samples) as input and determine along which coordinate axis the point list has the largest extent. Then, all points are sorted along this axis, spliting the list into two parts with an equal number of points, and recursively processing both sub-lists in the same manner. In total, the cost for sorting all points is $O(n \log_2 n)$ for n points. As a complete tree is used, the order of the points in the list implicitly defines the hierarchy. Consider the example shown in Figure 6.2: nodes A to D are the leaf nodes after sorting. In the same way, nodes A and B are children of the node E, and so on. For all interior nodes, the minimum bounding sphere enclosing all child nodes is computed, as well as the cone of normals (stored as direction plus cone angle).

Deforming and Moving Geometry

For deforming geometry, the hierarchy (tree topology) itself is left unchanged and only the per-node data is updated. At run-time at the beginning of every frame, the leaf nodes' positions and normals are recomputed, and the interior nodes are updated in parallel, i. e., the minimum bounding sphere and cone of normals are recomputed, containing the two child-nodes' bounding spheres and normals, respectively.

This process works bottom-up by successively merging two nodes at a time, yielding a total of $O(n)$ operations for n leaves. This keeps the run-time cost for maintaining the point hierarchy low, and can reasonably handle deforming geometry. For moving objects a separate point hierarchy is created. The normals and normal cones are used for lighting computation and back-face culling during the point hierarchy traversal.

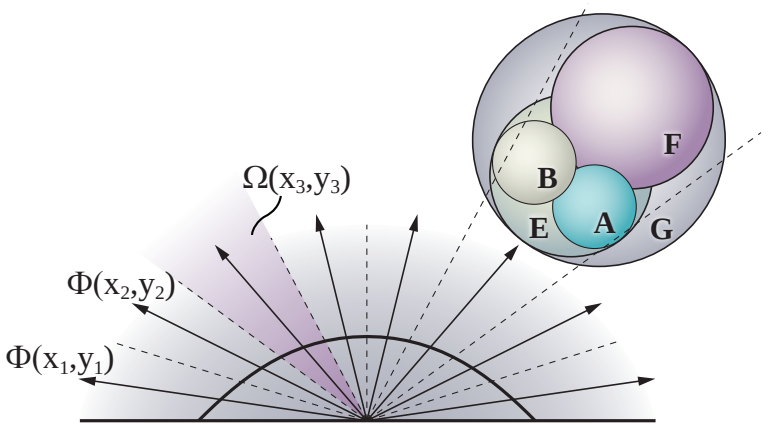

Figure 6.3: Every pixel (x_i, y_i) of a micro-buffer corresponds to a direction $\Phi(x_i, y_i)$ and subtends a solid angle $\Omega(x_i, y_i)$. The point hierarchy is traversed and rasterized such that nodes project to no more than one pixel in the micro-buffer. In this example, the nodes A, B, and F, of the point hierarchy in Figure 6.2, are selected.

6.2.2 Final Gathering Using Micro-Rendering

Final gathering is used for high-quality renderings to compute the indirect illumination at every visible surface point **p**. It involves gathering incident radiance $L_{\text{in}}(\omega_{\text{in}})$ from direction ω_{in} of the upper hemisphere at **p**. Usually BRDF importance sampling is used to gather more from directions that contribute more to the reflected radiance towards the observer.

Due to the typically large number of gather directions involved, this is an expensive operation and commonly used in the context of offline rendering only. The method enables parallel final gathering and achieves interactive frame rates through Micro-Rendering, which has been developed with the high parallelism of contemporary and future GPUs in mind.

In the following we first detail the basic Micro-Rendering procedure for a single gather point. In Section 6.3 we then describe the implementation details, and how it is ensured that the computational power of such hardware is utilized to a very high degree.

Micro-Rendering generates images using the mapping $\Phi(x, y) = \omega$ relating a pixel (x, y) of the micro-buffer to a gather direction ω. We denote the solid angle subtended by the pixel under this mapping as $\Omega(x, y)$ (see Figure 6.3). Such a mapping can be any standard hemispherical projection; however, a custom mapping as described in the next subsection is used. The micro-buffers store an index of the nearest visible node as well as its distance at every pixel (i. e., an index and depth buffer is maintained); A micro-buffer is typically small, ranging from 8×8

to 24×24 pixels in the examples.

The basic image formation process starts with computing the cut in the point hierarchy, which also determines which point sample is visible for every micro-pixel. Micro-Rendering then gathers the incident radiance for every micro-pixel, and convolves it with the BRDF, yielding the radiance reflected towards the observer. Instead of the convolution with the BRDF, importance sampling can be integrated easily at little additional cost by changing the mapping function $\Phi(x,y)$ appropriately (Section 6.2.3).

Point Hierarchy Cut

The cut is computed using a depth-first search in the point hierarchy starting from the root node. For each node, the algorithm evaluates the selection criterion: first the direction ω_{in} to the node's center, the solid angle ω_{in} that it subtends, and the pixel $(x_i,y_i) = \Phi^{-1}(\omega_{\text{in}})$ that the direction maps to are computed. If $\omega_{\text{in}} > \Omega(\Phi^{-1}(\omega_{\text{in}}))$, then the node is larger than 1 pixel under the projective mapping, and the process will proceed with the child-nodes. Otherwise, a depth test is performed and if the node's distance is smaller than the depth value at (x_i,y_i), its index is stored and updates the depth buffer.

On-demand Ray Casting

When encountering leaf nodes, a further refinement is not possible. Such nodes potentially project onto several pixels, with possibly distorted shapes. The approach opts to resolve the exact visibility of such nodes using ray casting after computing the cut. To this end, their indices are stored in a *post-traversal list*, which is maintained for every Micro-Rendering, for later processing. That is, after traversal and rasterizing all 1-pixel-sized nodes, rays are cast, one for every pixel in the micro-buffer, to find the closest intersection with the nodes in the post-traversal list and the micro-buffer is updated accordingly. This obtains an accurate Micro-Rendering with water-tight surfaces.

Convolution

After ray casting, the micro-buffer stores the index to the nearest visible node in the point hierarchy for every pixel. To obtain the reflected radiance, Micro-Rendering computes the radiance $L_{\text{in}}(\omega_{\text{in}})$ from every node, weights it by the respective solid angle, multiplies by the BRDF, and sums up all contributions. Note that there are different strategies to obtain $L_{\text{in}}(\omega\,\text{in})$. The approach can determine it dynamically

by extracting the position and normal of each node and computing the lighting including shadowing using shadow maps (for direct lighting or instant radiosity lighting). When using only diffuse surfaces, the system can store the reflected radiance inside the point hierarchy [Christensen 2008], or obtain a radiance estimate for every node from a photon mapping solution (Figure 6.9). Results for all these strategies are presented in this chapter. In a final application, a radiosity-like light propagation scheme using the point hierarchy in Section 6.4 allows to efficiently compute multiple bounces.

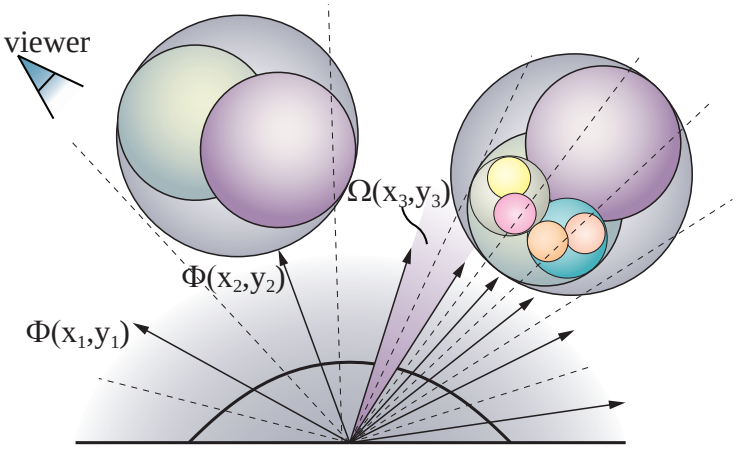

Figure 6.4: The warping function $\Phi(x_i, y_i) = \omega_{\text{in}}$ relates pixels in the micro-buffer to directions distributed according to the BRDF. Consequently the solid angle, $\Omega(x_i, y_i)$, corresponding to each pixel varies as well.

6.2.3 BRDF Importance Sampling

We have yet to specify the mapping $\Phi(x, y)$ that relates a pixel to a gather direction ω. One could use a standard hemispherical parameterization, such as a Nusselt projection: $\Phi(x, y) = (x, y, \sqrt{1 - x^2 - y^2})$. However, if the surface point $\mathbf{p}$ is highly glossy, much of the information stored in the micro-buffer is practically irrelevant, as incident radiance from only a small solid angle (and thus few pixels) will be reflected towards the viewer. Therefore, importance sampling is applied to the BRDF. In the context of Micro-Rendering, this means that we require more pixels in the micro-buffer to correspond to important sample directions.

This can be achieved by defining the mapping $\Phi(x, y)$ appropriately. For a given point $\mathbf{p}$ in the scene, we know the current viewing direction ω_{out}. We take the 2D light-dependent slice $f_{\text{r}}^{\omega_{\text{out}}}(\omega)$ of the BRDF $f_{\text{r}}(\omega \rightarrow \omega_{\text{out}})$ at that point, i. e., we fix the view direction ω_{out}, and parameterize it by the x- and y-coordinate of ω: $f_{\text{r}}^{\omega_{\text{out}}}(\omega_{\text{x}}, \omega_{\text{y}})$. After normalizing the slice with $1/\rho$, where $\rho = \int f_{\text{r}}^{\omega_{\text{o}}}(\omega) \text{d}\omega$, we

regard it as a 2D probability distribution function (PDF). We compute its inverse cumulative marginal and conditional distributions, M^{-1} and C^{-1}, which are used to map from uniformly distributed x and y (the pixels) to

$$\begin{aligned} \omega_{\mathrm{x}}(x) &= M^{-1}(x) \qquad \text{and} \\ \omega_{\mathrm{y}}(y) &= C^{-1}(y|\omega_{\mathrm{x}}(x)). \end{aligned}$$

These are then used to define the importance-warped mapping

$$\Phi(x,y) = \left(\omega_{\mathrm{x}}(x), \omega_{\mathrm{y}}(y), \sqrt{1-\omega_{\mathrm{x}}(x)^2-\omega_{\mathrm{y}}(y)^2}\right).$$

This particular mapping performs importance sampling according to $\omega_{\mathrm{z}} f_{\mathrm{r}}(\omega \to \omega_{\mathrm{o}})$. The cosine term ω_{z} is implicitly included due to the chosen parameterization of the BRDF slice. Figure 6.4 illustrates a BRDF-based mapping function $\Phi(x,y)$ and its associated $\Omega(x,y)$. Figure 6.5 shows a micro-buffer for a glossy gather sample with and without importance-warping.

While for some BRDFs this mapping can be derived analytically (e. g., specular Phong component), the approach opts for generality; i. e., the PDF is always tabulated and the inverse distributions computed numerically.

Note that this requires the inverse mapping $\Phi^{-1}(\omega) = (x,y)$ in order to project a node onto the micro-buffer, while the forward mapping is required for ray casting. In practice, Micro-Rendering computes both the forward and inverse mapping. Further, it needs to know what the subtended solid angle of a pixel is, i. e., we need to define $\Omega(x,y)$ as well. The technique uses a first-order approximation by taking the magnitude of the gradient of $\Phi(x,y)$. Note that the local coordinate system is slightly jittered at every gather sample to avoid banding artifacts.

As the micro-buffer is now importance-warped, a simple *sum* of all pixels scaled by ρ yields the convolved indirect illumination; no more multiplication with the BRDF $f_{\mathrm{r}}(\omega \to \omega_{\mathrm{out}})\omega_{\mathrm{z}}$ or the pixel's solid angle is required.

Discussion

Rendering with highly glossy materials is demanding in terms of warping, ray-casting and point sampling. Although warping itself does not restrict glossiness if analytic formulas for mapping BRDFs are available, tabulation might miss features, effectively limiting glossiness. Highly glossy BRDFs also require more point samples in the hierarchy, because surface edges and textures can become visible in reflections. Also, the projection of more nodes is likely to be larger than 1 pixel in diameter, thus requiring ray-casting.

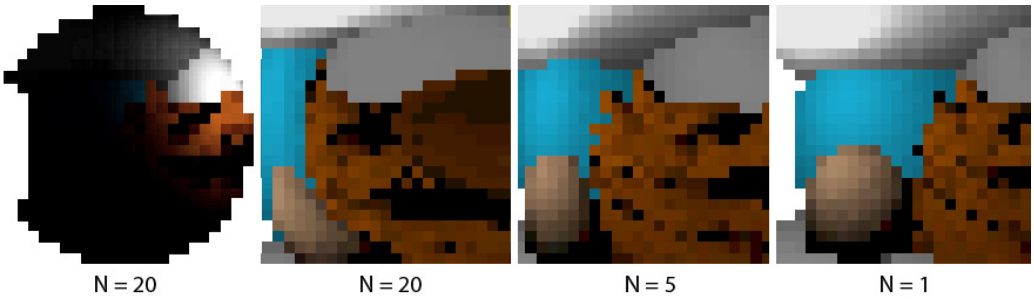

Figure 6.5: Micro-buffer for a glossy BRDF with a standard hemispherical mapping (BRDF-weighted) and with importance-warped mapping (Phong, N = 20). The importance-warped micro-buffer uses the available space more efficiently. On the right, the importance-warped micro-buffer for Phong N = 5 and N = 1 are shown.

Similar to the proposed method, Jensen [1995] uses an importance-warped table to roughly record directions that contribute most to the reflected radiance in order to improve importance sampling for path tracing. This table is computed from nearby photons in a photon map only, thus possibly containing holes. In contrast, a BRDF-warped micro-buffer is used to accurately record all incident lighting, directly yielding reflected radiance after summation.

6.2.4 Bilateral Upsampling

For interactive previews the indirect illumination is computed at a lower resolution image only, and upsample the shading results using bilateral upsampling [Sloan et. al. 2007] to the full resolution; direct lighting is always computed at full resolution. During upsampling, interpolating across discontinuities in geometry or reflectance is avoided, i. e., across silhouettes or over large differences in normal. For this, interpolation weights as proposed in Sloan et. al. [2007] are used. Indirect illumination is typically low-frequency and thus can be interpolated and, in addition, direct illumination often masks possible artifacts.

Nevertheless, pixels for which the interpolation is deficient are detected. This is the case when the interpolation weights are nearly zero. For such pixels the indirect illumination is computed in an additional render pass rather than interpolating (similar to [Dachsbacher and Stamminger 2005]). Unfortunately, a high overhead due to additional memory transfers (in a CUDA implementation), not the Micro-Rendering cost, slows down this approach significantly.

Preview quality using indirect illumination computed at $1/16$-th image resolution (i. e., ¼ in each dimension) typically renders at 4 to 10 Hz, whereas the full simulation runs at 0.5 to 1.0 Hz. There are more sophisticated techniques [Ward

and Heckbert 1992] to determine where to compute indirect illumination, and how to interpolate irradiance. Note that these techniques could be combined with Micro-Rendering method, but the integration into a GPU-based framework is challenging.

6.3 Implementation

Micro-Rendering has been designed to exploit the parallelism of GPUs. The method was implemented using NVIDIA's CUDA and thus the respective terminology will be used in this section.

6.3.1 Data Structures

The Micro-Rendering implementation is based on two important data structures: the scene's geometry stored as a point hierarchy, and the micro-buffers and post-traversal lists for efficiently computing parallel final gathering.

The point hierarchy is created in a preprocessing step (Section 6.2.1), and stored as an array in global memory that is used by all rendering threads. For each node, 128 bits are allocated, storing a node's position and radius (4×16 bits), normal and surface albedo (both quantized to 3×5 bits, packed into one 32 bit value), and the cone angle (quantized to 8 bits). In contrast to the original QSplat data structure, Micro-Rendering stores absolute positions and radii to allow direct access to interior nodes, and to avoid stack maintenance which is expensive on GPUs. This packing leaves 24 bits free, which was experimentally used to store the reflected radiance for every point sample. As expected, the clamping to 8 bits per channel corrupts the results, and the approach thus opts for storing the reflected radiance as half-floats in a separate array instead. The update of the point hierarchy data is done using CUDA at the beginning of every frame.

The micro-buffers are allocated in local memory, storing 32 bits for every pixel, out of which 8 bits are used for the depth component, and 24 bits for the node index. Thus the current implementation is limited to a point hierarchy with a maximum of 2^{24} nodes (i. e., 2^{23} point samples), which is significantly more than what is used in the examples. In addition to the micro-buffer, the post-traversal list is allocated of the same size providing space for the indices of the nodes whose visibility is resolved using ray-casting.

Parallel Micro-Rendering using CUDA

In order to perform as many Micro-Renderings in parallel as possible, one CUDA thread is launched for each Micro-Rendering, i. e., every gather sample. Each thread first computes the BRDF warping functions, and then continues to compute the point hierarchy cut for the respective gather sample. Note that the BRDF warping is computed for every single Micro-Rendering, which allows for arbitrary, spatially-varying BRDFs. The output of this first step is a partially finished micro-buffer and the post-traversal list. Ray casting is typically only necessary for 10 % to 30 % of the Micro-Renderings (i. e., the post-traversal list is empty for the other 70 %-90 %). The threads with a non-empty list cast one ray for every micro-pixel, according to the tabulated warping function, and intersect it with the nodes stored in their respective post-traversal list to find the closest intersection. Experimented the Micro-Rendering was split into two kernels: computing the warping and the point hierarchy cut in one thread, and next casting rays through all pixels of a micro-buffer in parallel by using many threads. However, the increased parallelism did not amortize due to the overhead of memory transfers required for this approach. Instead of ray casting we have also experimented with rasterization of large (multi-pixel) point samples, which turns out to be costly due to the warped projective mapping. The required accuracy, i. e., no holes in the micro-buffer, eas found to be cheapest to achieve with ray casting.

A parallel execution works best if the instruction sequences of CUDA threads are as similar as possible. This is the case if two threads compute a similar cut and post-traversal list. In order to support this, all Micro-Renderings are enumerated according to a three-dimensional Morton-order space-filling curve to provide high spatial coherence [Sagan 1994]. Note, that the nodes are already memory-coherent, due to the nature of the space partitioning tree construction.

After filling the micro-buffers, each thread computes the reflected radiance by summing up its content. The results are then hand over to OpenGL for bilateral upsampling and final display.

6.4 Applications

Micro-Rendering can be used in various ways to achieve high-quality interactive renderings, and full-resolution renderings comparable to off-line ray tracing methods such as PBRT [Pharr and Humphreys 2004b]. This section outlines four applications.

6.4.1 One-Bounce Indirect Illumination

When directly lighting the point hierarchy, e. g. using shadow mapping techniques, and then using Micro-Rendering for final gathering, the system achieves renderings with one bounce of indirect light. As shown in Figure 6.1 and Figure 6.6 it captures all $L\{S|D\}^2E$ light paths.

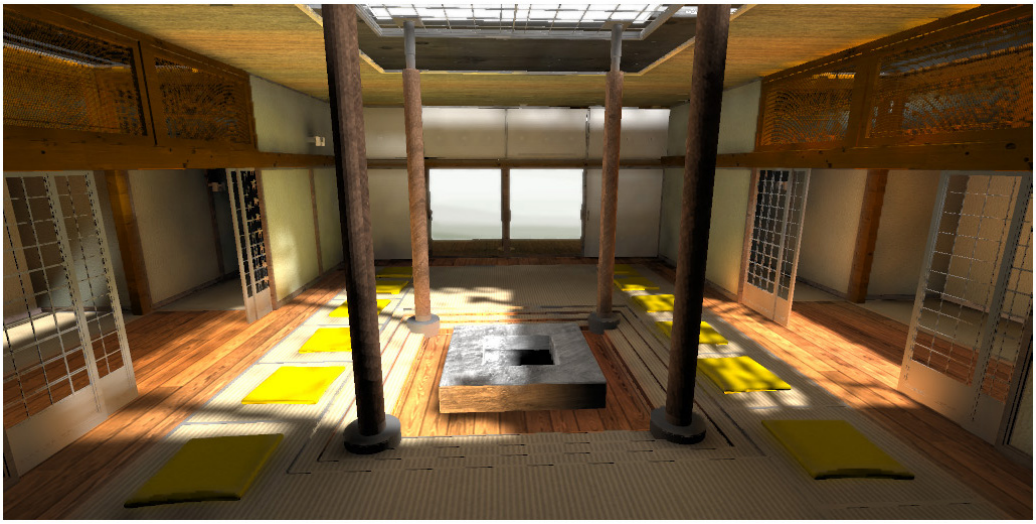

Figure 6.6: A scene made of 700 k triangles using 1 M points, renders at 0.7 Hz.

6.4.2 Multiple Bounces with Instant Radiosity

The basic technique can be extended to handle additional diffuse bounces by using instant radiosity to generate a set of virtual point lights [Keller 1997], which is then used to illuminate the points in the hierarchy. This enables us to render $LD^*\{S|D\}^2E$ light paths (Figure 6.7). Note that the typical instant radiosity artifacts, such as bright splotches and shadow aliasing, diminish thanks to final gathering.

6.4.3 Multiple Bounces with Radiosity

The hierarchical point representation of the scene's surfaces can also be used to compute light transport similar to hierarchical radiosity. Micro-Rendering is used to perform gathering in a Jacobi-iteration scheme: in every iteration the indirect lighting is gathered at the point samples from other surfaces in the hierarchy to

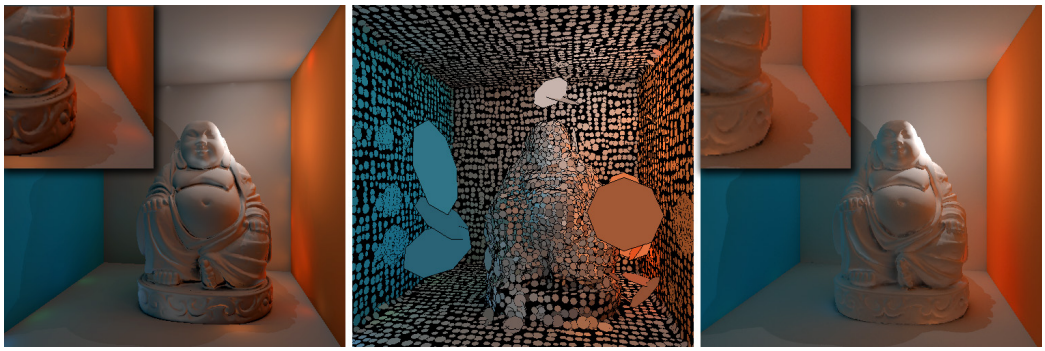

Figure 6.7: Multi-bounce indirect illumination for instant radiosity. Instead of directly using instant radiosity to illuminate a scene (left), the points in the hierarchical scene representation are shaded with Instant Radiosity (visualized in the middle) and a final gathering step is performed (right, 0.7 Hz, no upsampling). The additional final gathering step removes many of the artifacts of instant radiosity.

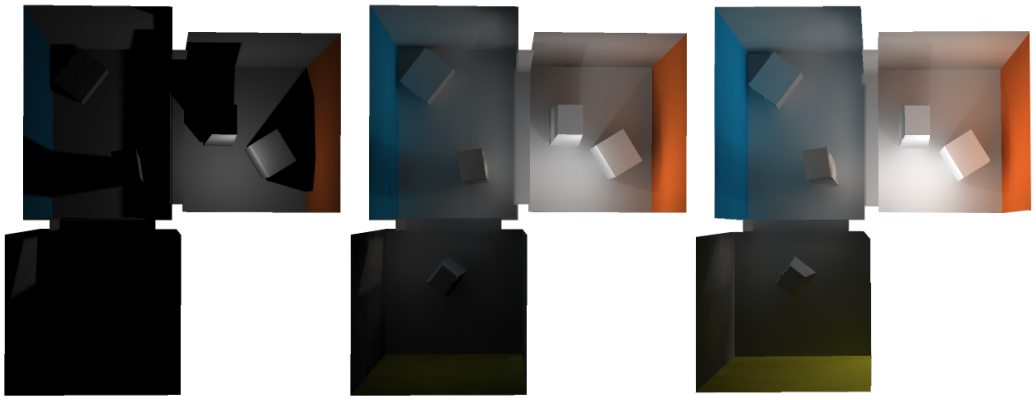

Figure 6.8: Direct illumination (left), one-bounce indirect illumination (middle) and two-bounce indirect illumination (right). Multiple bounces are computed at 5.0 Hz by gathering incident illumination at the hierarchy level 12 (4 k nodes) followed by a final gathering step at 128×128 surface locations. The final image is obtained from bilateral upsampling with anti-aliasing.

compute the reflected radiance. It is usually sufficient to do this for an interior node level and then to update the illumination of the entire hierarchy using push-pull [Cohen, Wallace and Hanrahan 1993]. This computation is initialized with the direct illumination at all points. The final rendering pass consists of performing Micro-Rendering at every pixel, effectively displaying the radiosity solution using final gathering; see Figure 6.8 (right).

6.4.4 Photon Mapping

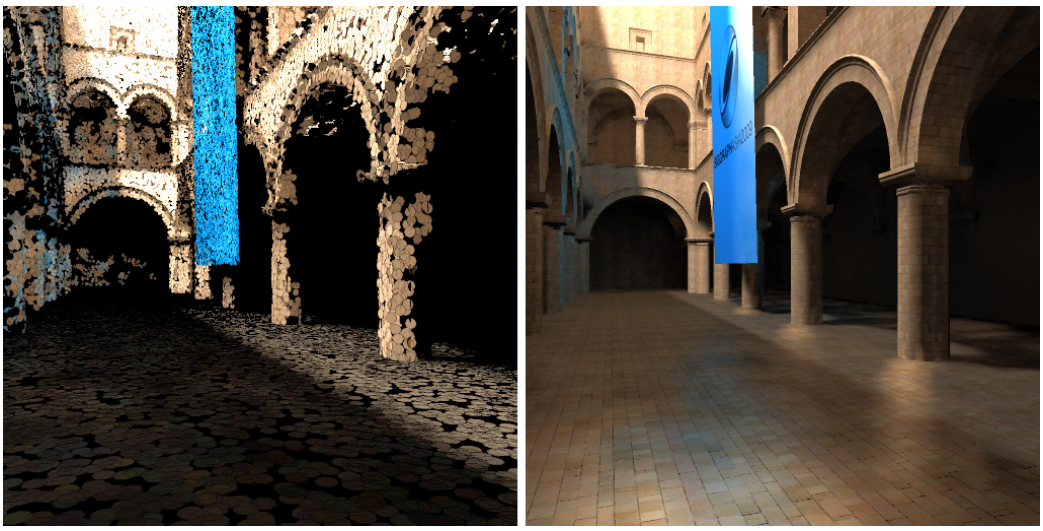

Figure 6.9: Final gathering for photon mapping. Density estimation is first performed on the photon map (computed offline) for every point in the hierarchy (left). On the GPU, final gathering is performed (right, note the glossy floor), which allows for interactive walkthroughs of photon mapped scenes (at 2 Hz) using bilateral upsampling with anti-aliasing from 128×128 to 1024×1024.

The method can be used to interactively display solutions stored in diffuse photon maps. In a preprocess, a radiance estimate is obtained for each leaf node of the point hierarchy through density estimation from the photon map, and compute the radiance values for the interior nodes using pull steps. Micro-Rendering is then used for final gathering; see Figure 6.9.

6.5 Results

This section presents results, rendered at interactive frame rates on a quad-core 2.4 Ghz CPU with an NVIDIA GeForce 280 GTX. As mentioned before, all scene components (geometry, lighting, BRDFs) are allowed to change on-the-fly. Unless

otherwise mentioned, result images are rendered at 512×512 pixels using 24×24 micro-buffers with one-bounce indirect illumination and direct lighting of the points in the hierarchy using shadow maps.

Figure 6.10–6.13 show four different scenes rendered with the proposed method with gather samples at every pixel and at every 4th pixel (in each dimension) using bilateral upsampling, and we compare to a reference solution computed with Monte-Carlo path-tracing [Pharr and Humphreys 2004b]. Broadly speaking, the differences between the reference solution (taking minutes to hours to compute) and Micro-Rendering are minor. When using bilateral upsampling, the differences are slightly more perceptible; however, interactive frame rates of up to 10 Hz are achieved.

Figure 6.14 compares Micro-Rendering using bilateral filtering to ISMs [Ritschel et. al. 2008b]. The number of gather locations was adjusted to achieve roughly the same speed as ISMs. Note how Micro-Rendering achieves superior quality in indirect shadows compared to ISMs. Furthermore, ISMs can only be used for low-glossy scenes, otherwise VPLs will become visible; this precludes scenes as shown in Figure 6.1.

The number of required gather locations is scene dependent, and scenes with higher geometric complexity or glossy materials naturally require more samples. Figure 6.15 compares the rendering quality using different numbers of sample locations. For diffuse surfaces $1/16$ resolution is visually sufficient, whereas for the glossy dragon more locations ($1/4$ resolution) are required.

Figure 6.16 demonstrates what happens if the post-traversal ray-casting step is omitted. Holes occur in the micro-buffers and the rendering quality decreases, especially around edges. Post-traversal ray casting is therefore an integral step in the proposed method. The quality improvement is worth the additional ray casting cost.

Figure 6.17 compares the influence of the size of the micro-buffer on rendering quality. 24×24 pixels (used for all other figures) is the maximum that fits into local memory on the GPU used, but even lower resolutions still achieve acceptable results at a higher frame rate. However, note that at any resolution there is a possibility that small holes are missed (e. g., see the plant in Figure 6.11), as is the case for all rasterization-based methods.

The resulting performance is sub-linear in the number of input points, which is to be expected from a hierarchical representation; see the red curve in Figure 6.18, which indicates running time vs. scene complexity (dynamic horse scene represented with more and more points). The blue curve indicates that tree update is sub-linear until a hardware limitation is reached (at around 2^{20} point samples). As shown by

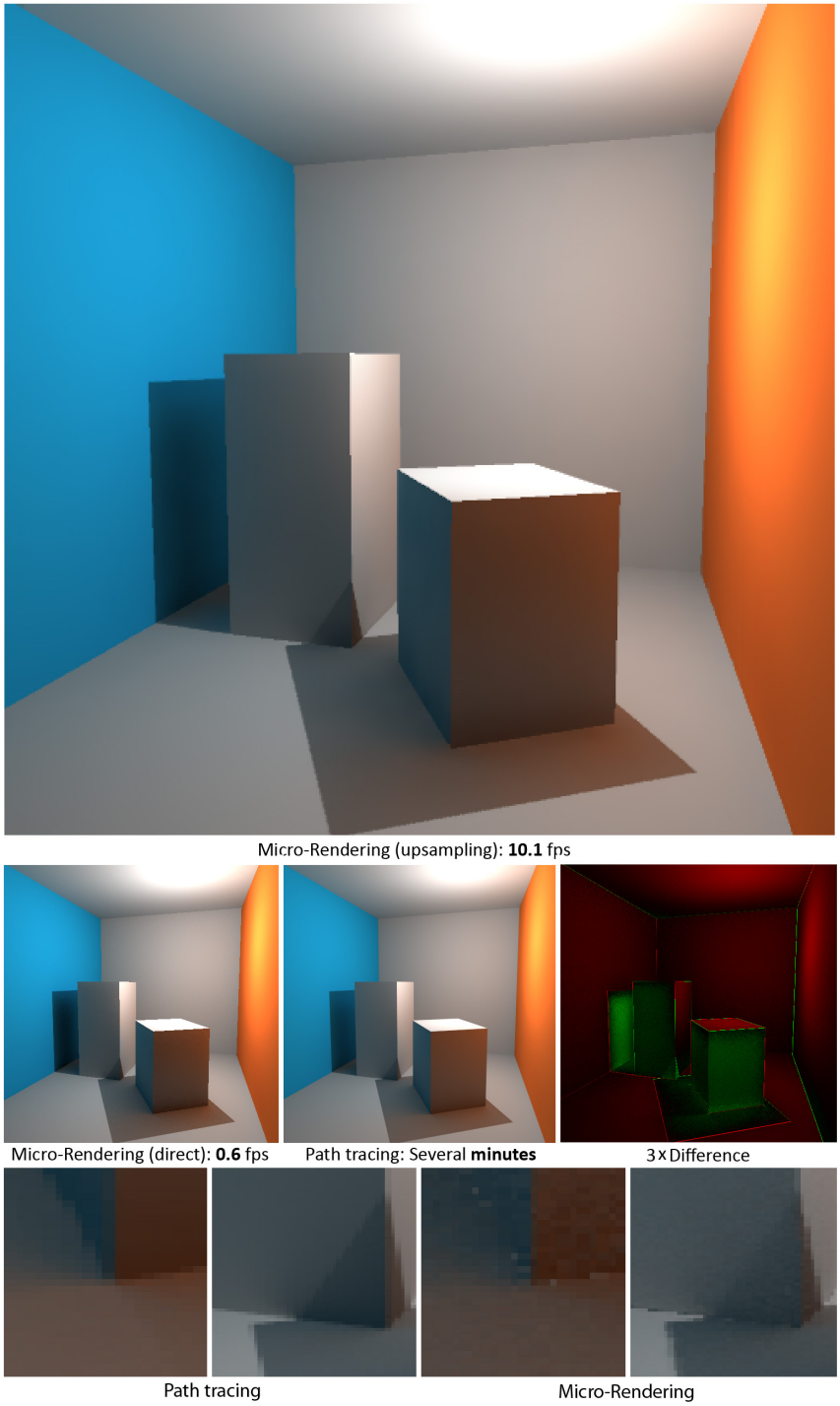

Figure 6.10: The simple Cornell box achieves high-quality global illumination results even with bilateral upsampling.

Figure 6.11: The geometrically complex plant scene shows some slight differences (see insets), which can be attributed to the discrete micro-buffers.

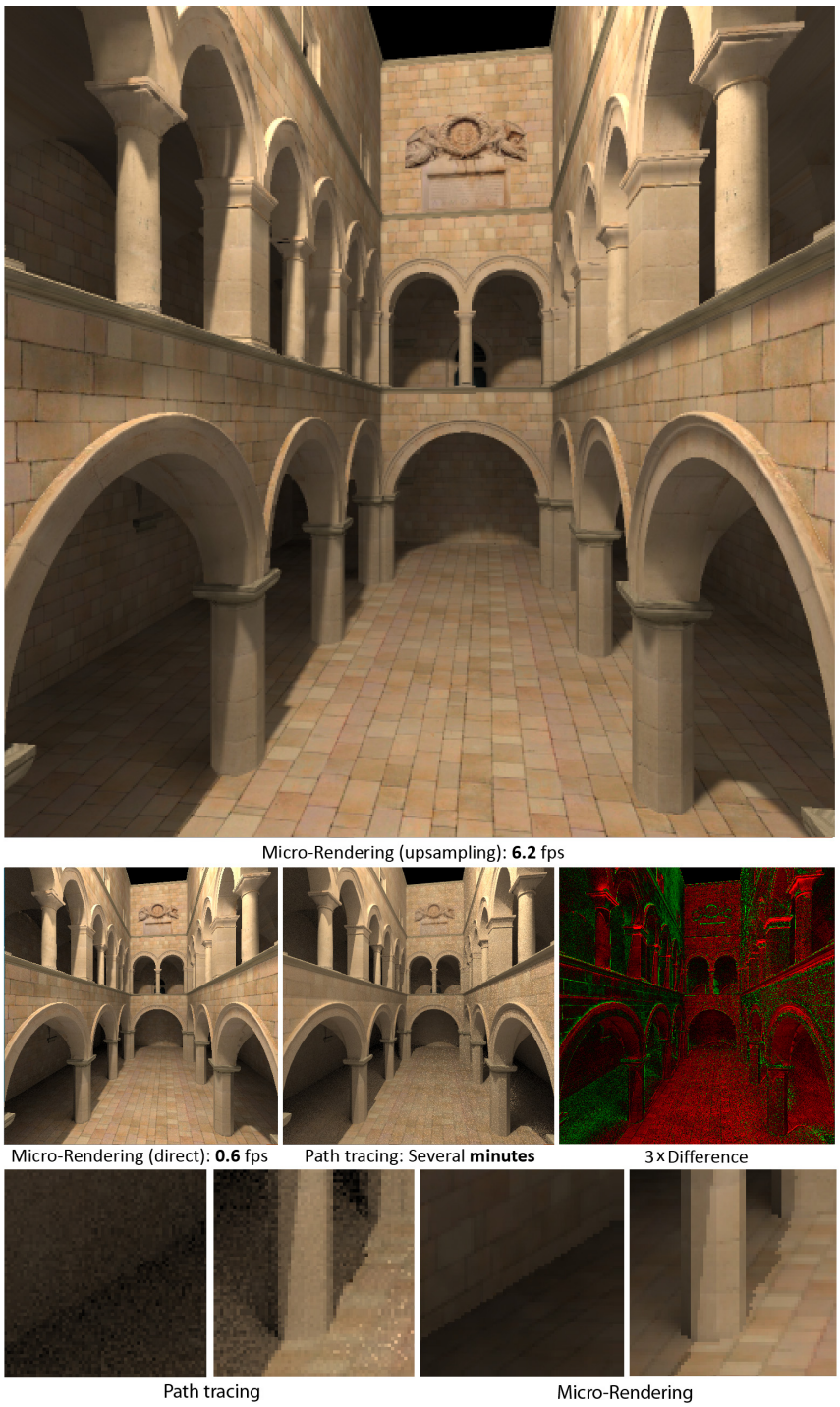

Figure 6.12: For the “Sponza” scene, the method produces results that are indistinguishable from the reference rendering. In fact, bilateral upsampling removes noise and produces the visually most pleasing result.

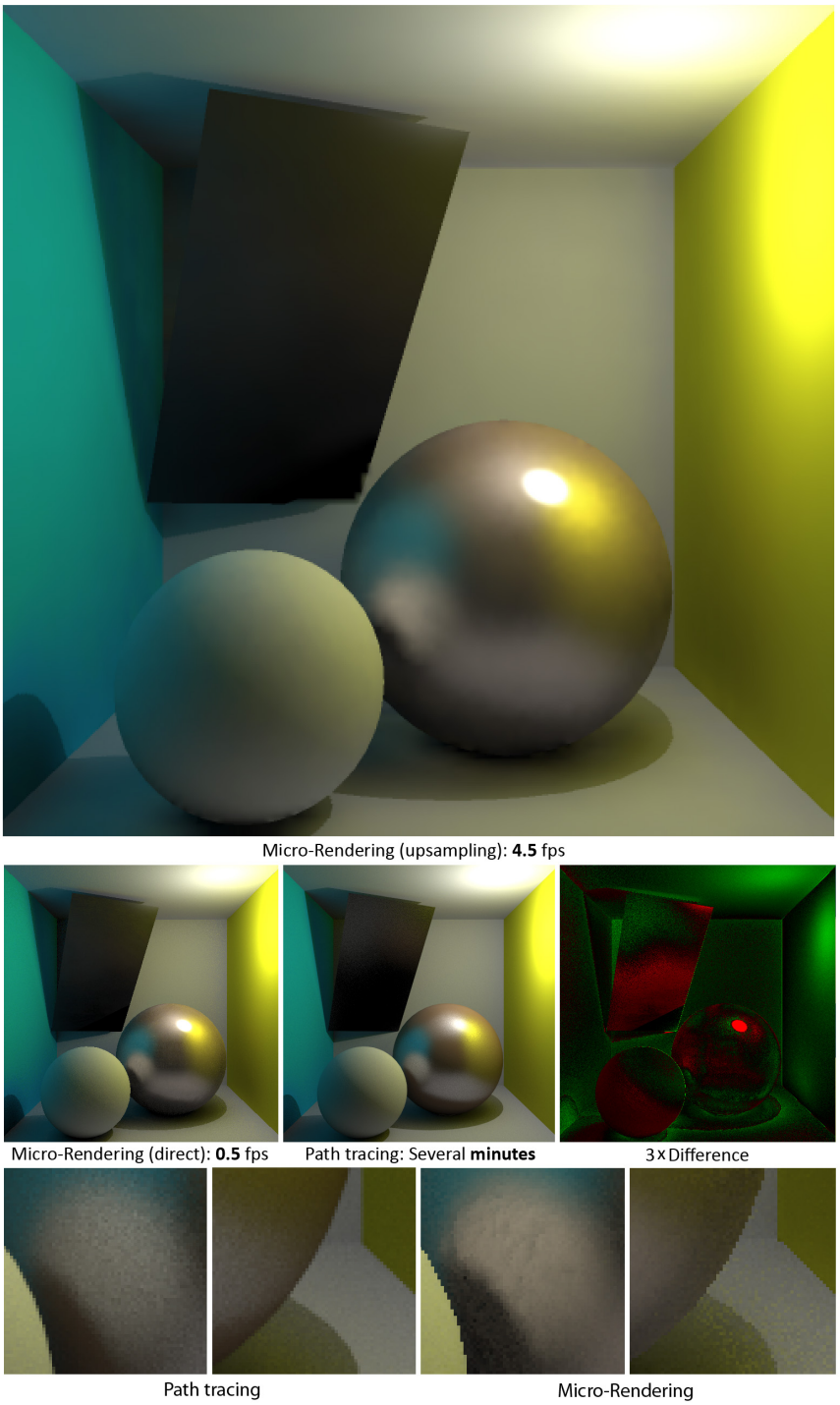

Figure 6.13: Very similar results are achieved for the glossy scene. However, as expected, bilateral upsampling changes the glossy reflection on the sphere slightly.

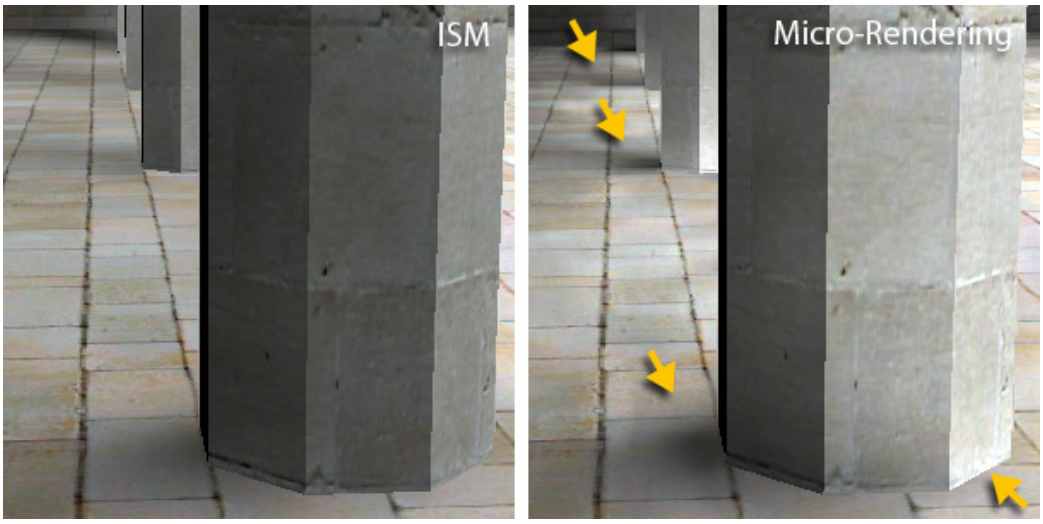

Figure 6.14: Comparison between ISMs [Ritschel et. al. 2008b] and Micro-Rendering. At the same rendering speed (5 Hz), Micro-Rendering achieves higher quality than ISMs.

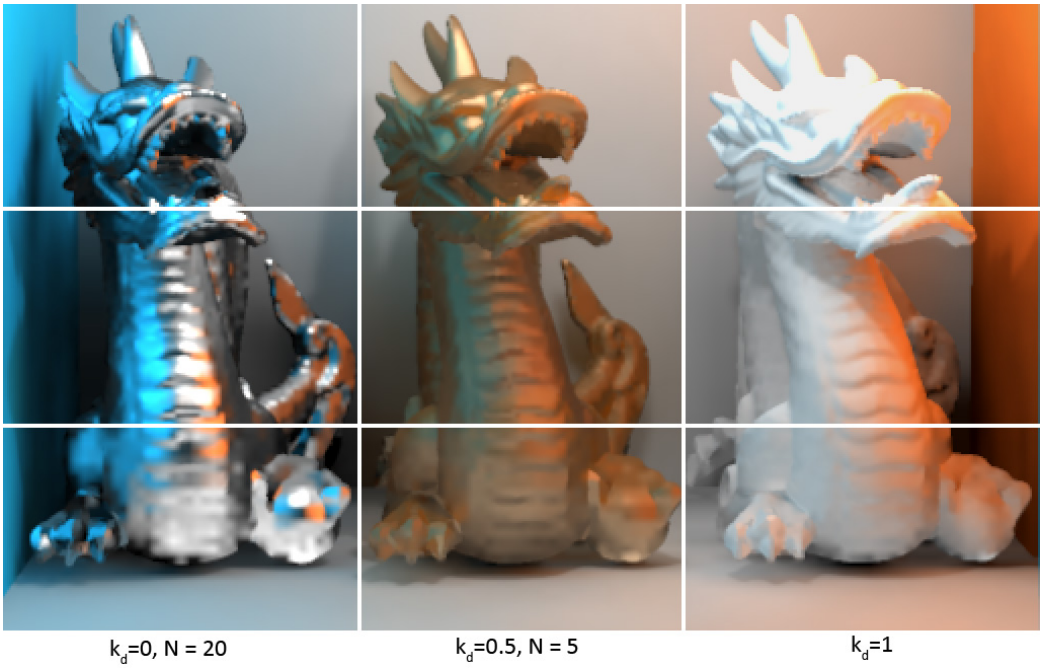

Figure 6.15: Final gathering at every pixel (0.83 Hz, top), ¼ of all pixels (2.2 Hz, middle), and 1/16 (4.2 Hz, bottom). Preview quality can be achieved with a very low number of gather samples for diffuse or low-glossy materials.

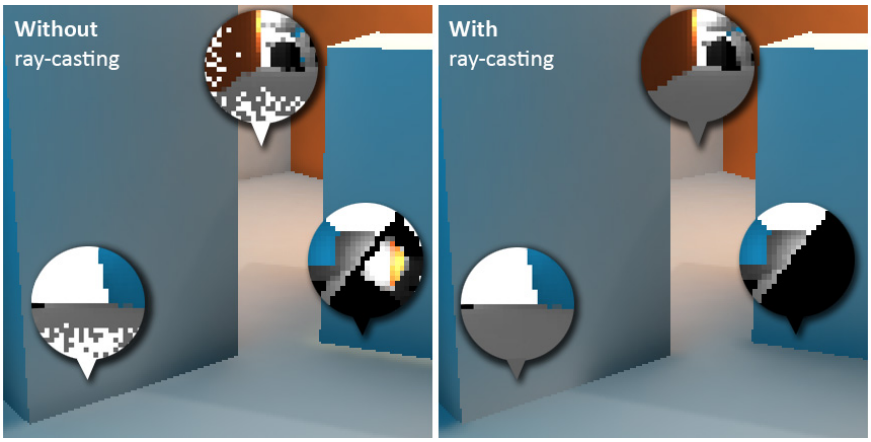

Figure 6.16: Micro-Rendering with (2.5 Hz) and without (2.9 Hz) the post-traversal ray casting (256×256). Ray casting is an integral step that is needed for high-quality results, especially around edges.

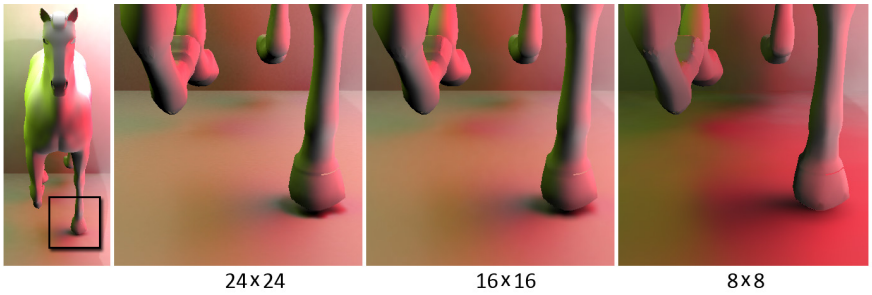

Figure 6.17: Influence of micro-buffer size on rendering quality (256×256). 8×8 (3.2 Hz), 16×16 (1.5 Hz), and 24×24 (0.7 Hz) are used. Smaller sizes are faster but quality decreases.

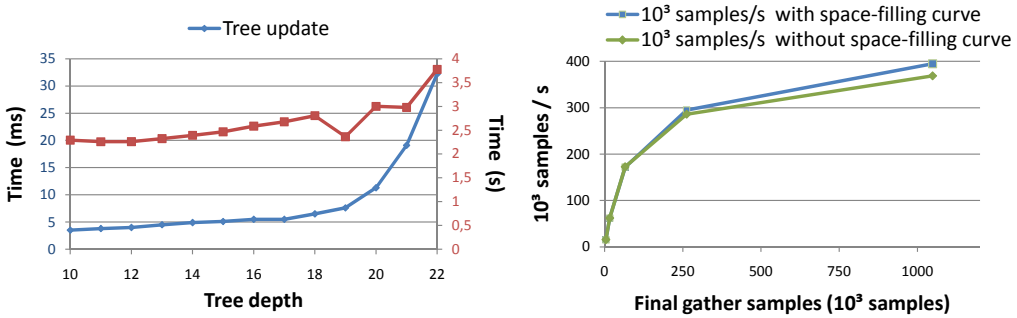

Figure 6.18: Left: the red curve indicates computation time vs. scene complexity (measured as tree depth, equals 2^N point samples). It indicates sub-linear complexity. The blue curve outlines tree-updating time. Right: the number of processed gather samples per second vs. the total number of gather samples in the image (green with, blue without space-filling curve to support spatial coherence).

the green curve in Figure 6.18, more gather samples can be processed per second when the total number of gather samples increases. This is to be expected due to the increased coherency between gather samples.

The computation time is (roughly) split as follows for a typical scene, such as the teaser. 1.5 % is spent on updating the hierarchy, 2 % on building the view-dependent per-pixel mappings $\Phi(x,y)$, 18 % on evaluating them, 60 % on rasterizing the point hierarchy, 8 % on ray casting, and 11 % on bilateral upsampling, tone mapping and direct lighting.

6.5.1 Discussion and Limitations

In typical scenes, Micro-Rendering is able to compute about 150 M final gathering samples with importance sampling (a 512×512 image with 24×24 micro-buffers at every pixel renders at about 1 Hz, including tree update, shading, etc.). CPU-based ray tracing can send out about 10 M rays per second per core, if the rays are coherent [Shevtsov, Soupikov and Kapustin 2007], i. e., about an order of magnitude less than the method proposed here. Currently reported numbers on GPU-based ray tracing indicate that about 20 M rays can be traced in dynamic scenes (including a complete rebuild of the acceleration structure) [Zhou et al. 2008].

While Micro-Rendering deals well with complex scene and arbitrary BRDFs, it has certain limitations. When glossy surfaces are present in a scene, more gather samples are required. In this case a simple regular gather sample distribution is not ideal and more elaborate distributions should be used [Křivánek et al. 2006]. As mentioned earlier, the approach opts to prevent banding artifacts by jittering the local coordinate system at each pixel. As a result some noise is visible in the result images.

While the method renders one-bounce indirect caustics – simply by gathering from highly glossy surfaces – more than two specular bounces are not supported. Other effects, such as transparent objects and refractions are currently not simulated.

The method was implemented in CUDA to explore the potential of parallelizing the various Micro-Rendering tasks (see Section 6.3). In the end, performing all tasks in a single kernel was fastest. This suggests that a pure OpenGL implementation might be just as fast or faster.

7

Interactive Reflection Editing

7.1 Introduction

The composition of traditional, as well as computer-generated images can be understood as an optimization process, wherein a number of artistic goals should be fulfilled within a number of physical constraints. When all physical constraints are met, photo-realistic images can be expected, but some artistic goals might not be achieved. On the other hand, fulfilling all artistic goals might lead to images with inadequate realism. For many centuries the limitations of human perception have allowed skilled artists to simultaneously achieve both goals.

One famous example from traditional art is the painting "The Rokeby Venus" by Diego Velázquez, shown in Figure 7.1: The reflection of the face in the mirror is physically incorrect, as was proven by photographic reproduction [Braham 1976]. Despite this, a pleasant and naturalistic image was achieved, as human observers have difficulties in assessing the physical correctness of reflections on complex surfaces [Fleming, Dror and Adelson 2003; Khan et al. 2006; Ramanarayanan et al. 2007].

Although human difficulties with the assessment of reflections are known in the field of computer graphics, very little research has been done on how to define and manipulate reflections outside physical bounds. Non-physical reflections are difficult to achieve with today's tools. Manual techniques, like 2D image warping [Wolberg 1998], compositing approaches, reflection map editing, or normal map editing, are tedious to apply and fail to properly capture perspective, complex geometry, animation sequences, local edits, partially diffuse materials, or occlusions.

In this chapter, a system for interactive reflection editing is presented, which allows

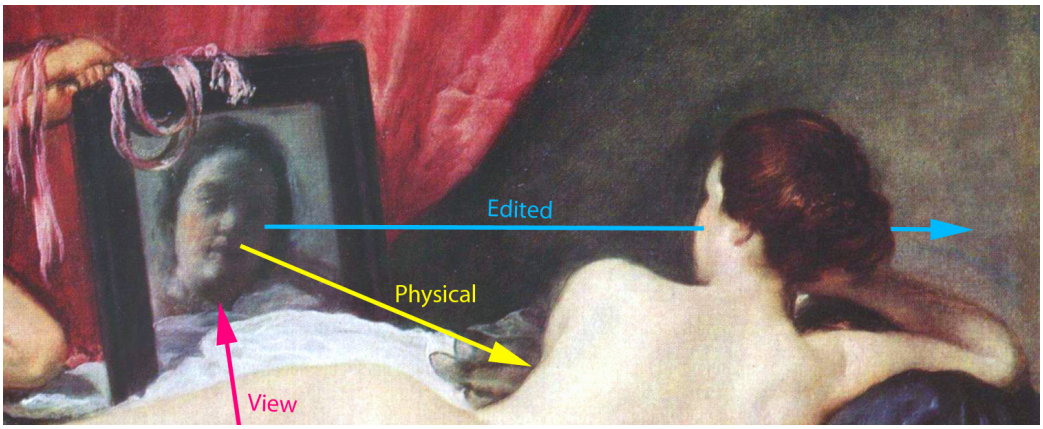

Figure 7.1: "The Rokeby Venus" (before 1651; detail) by Diego Velázquez (1599–1660) uses reflections beyond physical laws.

the user to escape the boundaries of physically correct rendering, without any such limitations. The system takes as input a virtual three-dimensional scene containing reflecting objects, together with a number of user-defined constraints, which define the edited reflection directions (e. g., Velázquez used one such constraint in his painting: "The mirror should reflect the face, not the body"). The output of the system is an image in which all constraints are fulfilled and the change in reflection direction is smooth. The system estimates the best (least squares) interpolated edited reflection direction for each pixel. The system runs in real-time on a GPU in HD resolution, is independent of the underlying reflection rendering method (ray-tracing or reflection mapping), is independent of meshing or parameterization, and works without pre-processing. It can handle animated meshes and allows more general specular light transport, like glossy reflections and refractions. Furthermore, the system does not require any user parameters besides the desired constraints. The user study shows that the system allows not only professional designers but also novices to design complex reflections in minutes. With today's tools, this is difficult for even an experienced designer to achieve.

7.2 User Interactions

With a given three-dimensional scene, the user can start to edit reflections by specifying a region of interest and using a mouse to put several constraints in the region (Figure 7.2-a). The user can then move, rotate and deform the reflection in the region by dragging the constraints (Figure 7.2-b and c) and inspecting the resulting reflections in real-time.

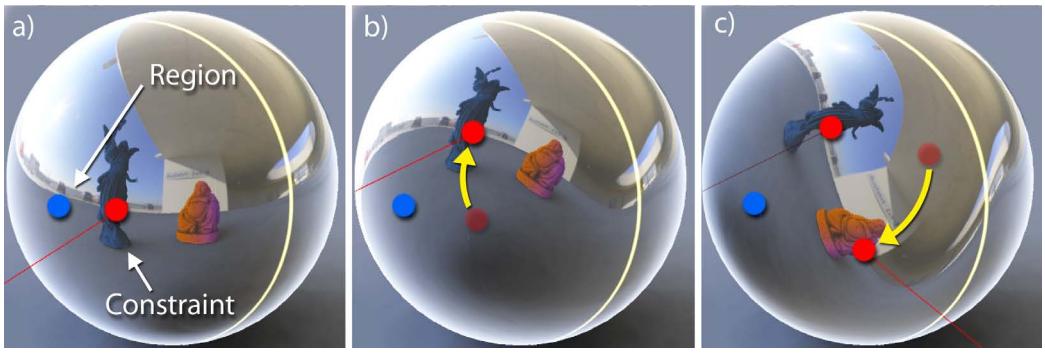

Figure 7.2: (a) Specifying the region of interest on the sphere (blue center and white boundary) and a constraint (red point). (b) Moving the constraint to translate the reflection. (c) Putting in one more constraint and deforming the reflection.

7.2.1 Constraints

Each constraint has two handles: a red one on the surface of the reflective object and a green one located on the reflected object (red and green points in Figure 7.3-a). Both handles can be moved to edit the reflection.

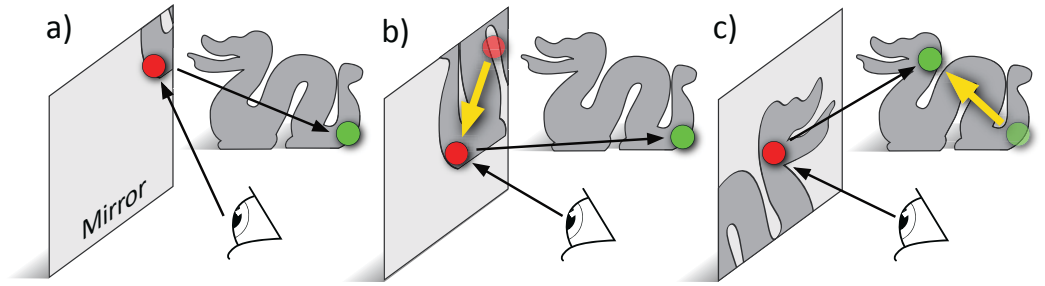

Figure 7.3: (a) A constraint is placed on a mirror, which reflects only a small part of the dragon. The constraint visualizes how the green point is reflected at the position of the red point. (b) Moving the red handle (yellow arrow) on the mirror, drags down the reflection. (c) Dragging the green handle (yellow arrow), changes the position that is reflected at the red point.

The red handle drags a reflection to a new location, e. g., in Figure 7.3-b moving the red handle down on the reflecting mirror also moves the reflection of the dragon down. Moving the green handle from the reflected location to a different location makes this the new reflected location, e. g., in Figure 7.3-c moving the dragon's head to the center of the mirror. Practice has shown that manipulation of the red handle is useful for subtle edits, whereas manipulating the green handle results in more substantial changes.

7.2.2 Regions

To limit the influence of the edit operation within the scene, several tools are used for specifying the region of interest. Without a region, changing a reflection would alter the entire scene. Creating a region allows multiple independent and well localized edits within one scene. Once created, regions can also be moved by dragging their blue handle. The user can choose between three types of regions: Euclidean, geodesic, and free-form.

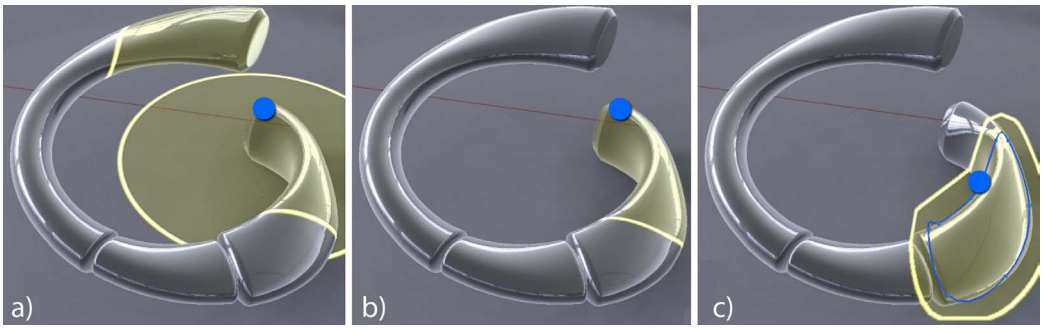

Figure 7.4: Regions on a spiral object: Euclidean regions (a) influence the upper part, which can be avoided by using geodesic (b), or free-form regions (c).

Euclidean regions are defined by a (blue) center and a radius in Euclidean space (Figure 7.4-a). These are easy to use and work on multiple or disconnected objects. Geodesic regions use a surface location and radius, which is measured over the surface (Figure 7.4-b). These are more intuitive if the object has a complex shape. Free-form regions are sketched on the surface (Figure 7.4-c) and allow any shape to be achieved but these have to be manually defined by a user. These are most suited to expert users who want to work on small details. All regions have smooth-stepped borders with a user-defined width. Edited reflections in each region are smoothly merged with non-edited reflections. All constraints *inside* a region influence the solution for the entire region. We define a constraint as *inside* a region, when the region's influence at the constraint location is nonzero. Constraints outside all regions have no influence.

7.2.3 View control

When the user changes the viewpoint, the constraints no longer correspond to the original reflections because reflections are view-dependent. Nevertheless, a reflection constraint, created in its *generating* view, can be manipulated from any other arbitrary current view. There are two options to help the user manipulate

the reflections in a consistent manner. When the user selects a constraint and presses the "return to view" button, the system reverts to the generating viewpoint where the constraint was created. The alternative option is to apply the "freeze" button and make the reflections temporarily view-independent: while changing the viewpoint, the reflection does not move but freezes on the surface. This allows the user to edit the reflection as it is visible from the generating viewpoint from another arbitrary view. This feature can be very useful in some situations, e. g., if a handle is occluded in the generating view.

7.2.4 Animation

To simplify the handling of animated objects and deforming surfaces, the reflecting and reflected position, as well as the region positions, are optionally made relative to the (time-varying) surface. If the surface is deformed or the object is moving, the corresponding item naturally follows the object. This is useful for preserving a desired appearance on moving surfaces, and can also be used for special effects, where a reflection "tracks" a moving object. Additionally, a constraint can be key-framed over time, or can be dependent on any other scene parameter, such as the current viewpoint.

7.3 Reflection Editing

According to the law of reflection, the angle of the incident ray **i** is equal to the angle of the reflected ray **r** for a perfectly mirroring surface (cf. Figure 7.5). Given the surface normal **n**, the reflected ray is given by

$$\mathbf{r} = \mathbf{i} - 2\,(\mathbf{i}^\top \mathbf{n})\mathbf{n} \quad , \tag{7.1}$$

where **i**, **n**, and **r** are 3-vectors $(x,y,z)^\top$ normalized to 1.

This chapter presents a real-time system for interactive reflection editing, which allows the user to violate the law of reflection by specifying a constraint that redirects the reflected ray **r** in another direction, given by the edited ray **e**. The user can specify this edited ray with an intuitive user interface that was described in Section 7.2.

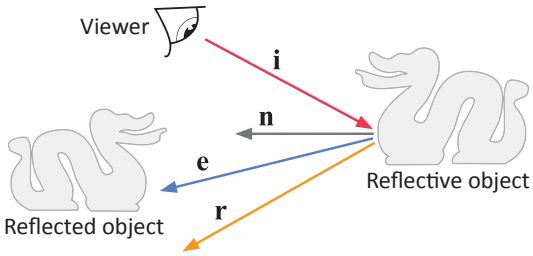

Figure 7.5: Following the law of reflection, an incident ray $\mathbf{i}$ is reflected in direction $\mathbf{r}$ at normal $\mathbf{n}$. With reflection editing, $\mathbf{r}$ is redirected to the edited ray $\mathbf{e}$, to fulfill a user-defined constraint, e. g., to always reflect the left dragon.

7.4 Interpolation Algorithm

If the user specifies multiple constraints, a smooth field of edited rays that interpolates between the user-defined constraints is sought, which can be found by a Moving Least Squares approach. Let N be the number of given constraints, where each is specified for the location $\mathbf{p}_n$ on the reflective surface, with $n = 1, \ldots, N$. For all locations $\mathbf{p}_n$, the user-defined constraints define the original reflected rays $\mathbf{r}_n$ and the edited rays $\mathbf{e}_n$. To find the interpolation at the intermediate positions $\mathbf{q}$, the best 3×3 rotation matrix $\mathsf{R}(\mathbf{q})$ is determined, that minimizes

$$\underset{\mathsf{R}(\mathbf{q})}{\arg\min} \quad \sum_{n=1}^{N} w_n (\mathsf{R}(\mathbf{q})\,\mathbf{r}_n - \mathbf{e}_n)^2 \tag{7.2}$$

with

$$w_n = \frac{1}{d(\mathbf{p}_n, \mathbf{q})^2} \quad . \tag{7.3}$$

Depending on the geometric complexity of the reflecting object, either the Euclidean or geodesic distance can be used as a distance metric $d(\ldots)$, as described in Section 7.5.2. Other weighting schemes could be used as well, however, it is important that the weighting w_n goes to infinity when the distance d approaches zero. This is required to make the reflection interpolating (instead of approximating), because only the local constraint is in effect when the distance is zero. This property is essential because it guarantees that edits accurately show the reflection of the green handle (the reflected location) at the red handle (the reflecting position).

Differentiating Eq. 7.2 with respect to the elements of R and setting the derivatives to zero yields the optimal solution

$$\mathsf{R}'(\mathbf{q}) = \left(\sum_{n=1}^{N} w_n \mathbf{r}_n \mathbf{r}_n^\top \right)^{-1} \sum_{n=1}^{N} w_n \mathbf{e}_n \mathbf{r}_n^\top \quad . \tag{7.4}$$

This solution does not enforce that R' is a rotation matrix, however, the rotation matrix R can be obtained by an ortho-normalization step. Ortho-normalization ortho$(\dots)$ is achieved by polar decomposition of $\mathsf{R}' = \mathsf{R}\,\mathsf{S}$ into a rotational part R and a symmetric part S. A fast algorithm can be found in [Horn 1987]. Because the first part of Eq. 7.4 is always symmetric, R can be determined by

$$\mathsf{R}(\mathbf{q}) = \text{ortho}\left(\sum_{n=1}^{N} w_n \mathbf{e}_n \mathbf{r}_n^{\top}\right) \quad . \tag{7.5}$$

Consequently, the edited reflection direction for each position $\mathbf{q}$ on the surface is given by

$$\mathbf{e}(\mathbf{q}) = \mathsf{R}(\mathbf{q})\,\mathbf{r}(\mathbf{q}) \quad . \tag{7.6}$$

It is worth emphasizing that this solution to $\mathsf{R}(\mathbf{q})$ allows for very intuitive editing, because the user merely specifies the edited ray $\mathbf{e}_n$ and does not need to define complete rotation matrices at each constraint location. The 3×3 matrix $(\mathbf{e}_n \mathbf{r}_n^{\top})$, used in Eq. 7.5, can be interpreted as a degenerated rotation matrix of rank 1. To upgrade this matrix to a full rotation matrix the user would have to specify an additional degree of freedom, namely, the amount of rotation around the edited ray. The ortho-normalization step automatically fills this degree of freedom by taking the neighboring constraints into account.

If the viewpoint is changed, the edited reflection direction $\mathbf{e}(\mathbf{q})$ must be recalculated with Eq. 7.6, because the reflected ray $\mathbf{r}(\mathbf{q})$ is dependent on the direction of the incoming ray $\mathbf{i}(\mathbf{q})$. However, such a change in viewpoint does not affect the rotation matrix $\mathsf{R}(\mathbf{q})$ because the constraints are associated with the viewpoint for which they were specified by the user. This means that the edited reflections are view-dependent, and therefore behave similarly to physically correct reflections under changing viewpoints. The rotation matrix $\mathsf{R}(\mathbf{q})$ only needs to be re-calculated when the user changes existing constraints or specifies additional constraints.

Following Eq. 7.6, the constraints have an infinite area of influence. For example, if the user specified a single constraint, all reflection directions on the surface would be altered with the same rotation matrix, originating from that single constraint. To limit the area of influence of constraints and to allow better control over complex editing operations, the user has the option of specifying a region of interest. For each surface point $\mathbf{q}$ a region of interest defines a value $a(\mathbf{q})$ in the interval 0.0 to 1.0 (usually this value is 1.0 in the central area of the region and has a smooth fall off to 0.0 at the region's borders). For a smooth transition to the unaltered reflection direction $\mathbf{r}(\mathbf{q})$ at the border of the region, the rotation matrix R from Eq. 7.5 is replaced by

$$\mathsf{R}_a(\mathbf{q}) = \text{ortho}\left(a(\mathbf{q})\;\mathsf{R}(\mathbf{q}) + \Big(1.0 - a(\mathbf{q})\Big)\;\mathbf{r}(\mathbf{q})\mathbf{r}(\mathbf{q})^{\top}\right). \tag{7.7}$$

For the calculation of R (and the resulting R_a) only those constraints that fall within the region of interest are considered, i. e., $a(\mathbf{p}_n) > 0.0$. If no constraint lies in the region of interest, the unaltered reflection direction $\mathbf{r}(\mathbf{q})$ is used.

The system allows the user to specify multiple regions of interest that are mutually independent and can be edited completely on their own. If multiple regions overlap, their contributions are weighted according to their specific value of $a(\mathbf{q})$.

7.5 GPU Implementation

When editing visually complex reflections, high quality rendering and real-time feedback are necessary. This section describes how this can be achieved with current consumer hardware by stream processing on the GPU.

7.5.1 Interpolation

To generate a smooth interpolation field of edited reflection directions, the edited reflection direction $\mathbf{e}(\mathbf{q})$ (cf. Eq. 7.6) must be calculated for every pixel. This can be done in parallel on the GPU. First, the visible part of the scene is rendered into a texture, where each pixel stores the three-dimensional position and normal that is visible at this pixel. Second, the weights w_n, from Eq. 7.3, for all N constraints and all pixels are computed and stored into an N-layer array texture. While an Euclidean weighting can be computed on-the-fly, geodesic weights need to be computed for each pixel (as described in Section 7.5.2). Third, all constraints and regions are uploaded as shader constants to the GPU. The original reflected ray $\mathbf{r}(\mathbf{q})$ for each pixel can be calculated using the given position, normal, and the current viewer position. Afterwards, the system solves for the edited reflection direction $\mathbf{e}(\mathbf{q})$ with Eq. 7.6. In this process, all computations are done in parallel and independently for each pixel. A final step passes the calculated edited reflection direction to the rendering system to compute a reflection along this direction.

7.5.2 Geodesic Distance

The geodesic distance can be used instead of Euclidean distances in Eq. 7.3. Assuming densely tessellated meshes, the proposed system coarsely approximates the continuous geodesic distance using the discrete distance along triangle edges. Letting $\bar{\mathbf{p}}_n$ be the mesh vertex closest to the constraint position $\mathbf{p}_n$, the approach computes the discrete edge distance from $\bar{\mathbf{p}}_n$ to every other vertex on the mesh using

Dijkstra's algorithm on the CPU. Then, on the GPU, the geodesic distance $d(\mathbf{p}_n, \mathbf{q})$ from $\mathbf{p}_n$ to a surface location $\mathbf{q}$ is substituted, by interpolating the distance from the discrete distance at the vertices of the triangle that contains $\mathbf{q}$. As the location $\mathbf{q}$ always corresponds to a screen pixel, this can be done efficiently, just by drawing the per-vertex discrete distance field using smooth shading in one pass. Every layer of the N-layer array texture, which stores the weights, can be calculated simultaneously. While this approximation works well for dense meshes, efficient high-quality approximations for general meshes have been published [Surazhsky et al. 2005].

7.5.3 Rendering

Editing of reflections is independent of the reflection rendering implementation itself. For high frame rates and interactive feedback, reflection mapping is used by the system although fast ray-tracing could alternatively be employed.

The system applies a reflection mapping technique similar to the technique proposed by Heidrich and Seidel [1999], on top of a (precomputed) diffuse global illumination. For reflection mapping, the scene is broken up into individual objects and one cube map is rendered from the center of every object. If the reflection direction is edited, the altered reflection can simply be read from a different position in the cube map. The diffuse lighting is unchanged when the reflection direction is edited.

When the user performs an extreme edit, it is possible that the edited reflection direction points below the surface. Even in such a case, since the system will simply reflect the object behind the surface, the edited reflection is always smooth and there are usually no visible artifacts, such as discontinuities.

Ray-tracing is another option for rendering multiple, high quality, local reflections and refractions. Recent GPU approaches [Zhou et al. 2008] allow for dynamic scenes at interactive speed, but their implementation is intricate. While in theory ray-tracing scales well with geometric complexity, scenes with several hundred thousands of faces have not been demonstrated to run at the same speed as that granted by rasterization based reflection mapping. Ray-tracing glossy BRDFs is more exact, but is also much more time-consuming compared to pre-convolved [Heidrich and Seidel 1999] reflection maps.

7.6 Results

7.6.1 Applications

This system can be applied to all computer-generated three-dimensional scenes, as used nowadays for product visualization, movie and TV productions, or in computer-generated art. Figure 7.6–7.9 show some examples (please see the video provided with this thesis for details of the process by which the edits were created).

As complex geometry can result in visual masking [Ramanarayanan et al. 2007] the examples use mostly smooth surfaces to expose the edit quality. In Figure 7.6 the motivating example from art was reproduced (29.1 fps). The planar mirror is simple to handle with a single constraint and a single region, allowing an experienced user to comfortably perform the edit operation in less than half a minute. The "Kitchen" scene, Figure 7.7, is a complex scene with several 100 k triangles and multiple reflecting objects. With 3 regions, 4 constraints and 4 reflecting objects, it still can be edited at 16.8 fps. Figure 7.8 shows a scene, called "Ring", which might be used in a movie or TV production. The example also serves to show that reflection editing is not limited to mirror-like BRDFs, because it features reflections on a glossy metal ring (11.1 fps). The edit operation on the "Car" hood in Figure 7.9 shows how the method could be applied to product visualization. Even unclean meshes of complex shape and topology, such as the car body can be handled (9.4 fps). The video goes into more detail on how constraints can be applied to moving or deforming meshes.

Highlight editing

The system provides the means for highlight editing as a special case [Anjyo and Hiramitsu 2003]. In Figure 7.10, a round highlight is stretched over the side of the car.

Decoupling shadows and highlights

In physically correct rendering, the location of highlights and shadows are coupled. For artistic reasons, however, it can be useful to decouple them. In Figure 7.11, the scene is lit by an area light that is reflected inside a collection of objects. While keeping the shadows in place, the technique allows the user to move the highlight into a more prominent place, consistent with the reflection of other objects, but still keeping all soft shadows in place.

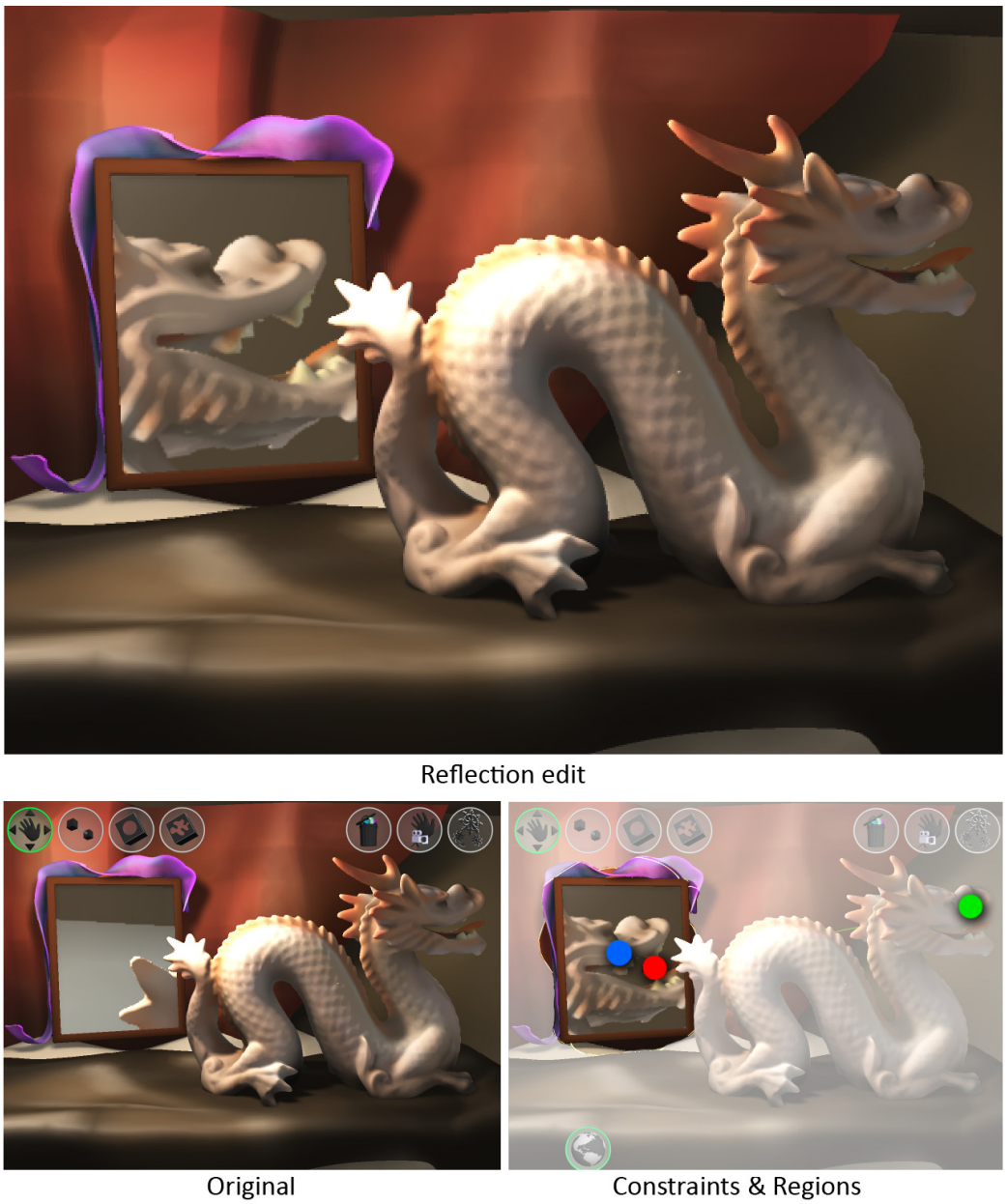

Figure 7.6: In a physically correct rendering the tail of the “Rokeby dragon” is visible in the mirror from the current viewpoint. After reflection editing the mirror reflects the dragon’s head (1 constraint, 1 region, 29.1 fps).

Figure 7.7: "Kitchen" is a complex scene with a large number of objects and triangles. After reflection editing the sink is displayed at a different position in the large reflective pot (4 constraints, 3 regions, 16.8 fps).

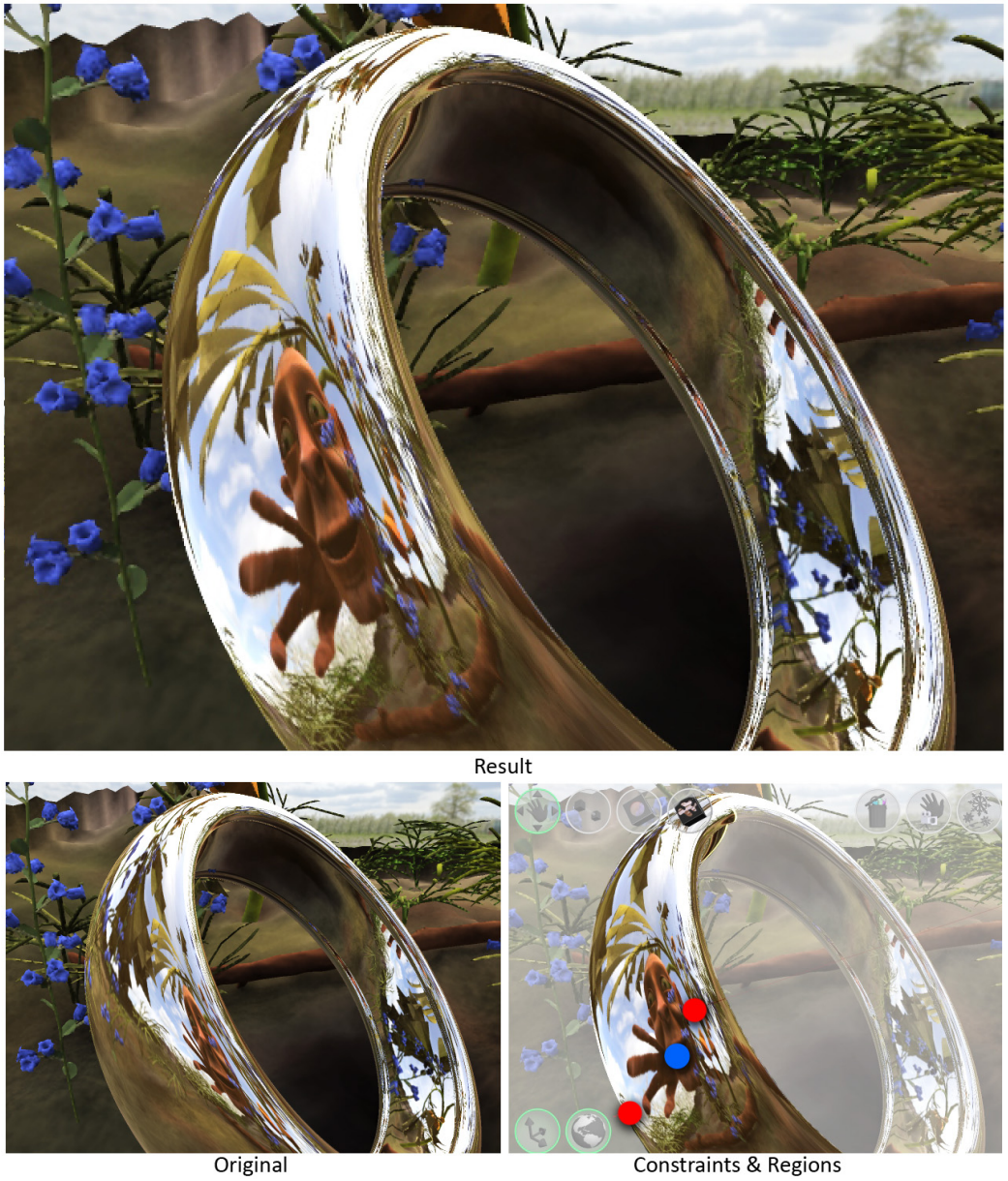

Figure 7.8: Editing the reflections on this “Ring” makes the reflected face more visible (2 constraints, 1 region, 11.1 fps).

Figure 7.9: A user changed the tree reflected in the hood of this “Car” to become more visible, also editing the highlights (4 constraints, 2 regions, 9.4 fps).

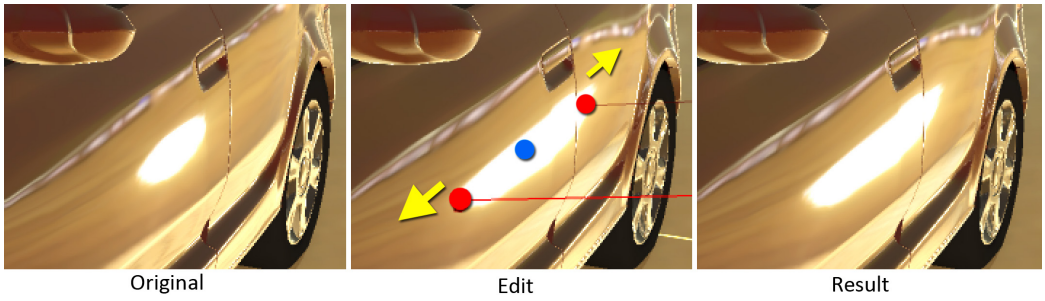

Figure 7.10: Highlight shape editing (2 constraints, 1 region, 10.8 fps).

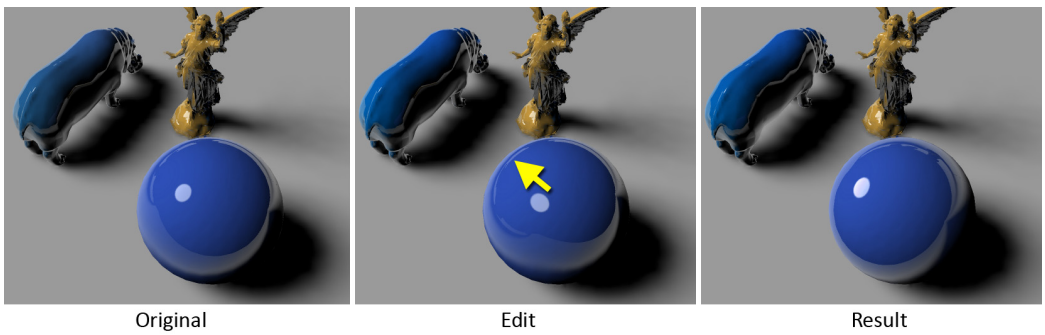

Figure 7.11: In this example the highlight was moved to another place. Starting from the original image, the highlight is moved, to give it the desired more pronounced look, yet left all the soft shadows unchanged (1 constraint, 1 region).

Refraction

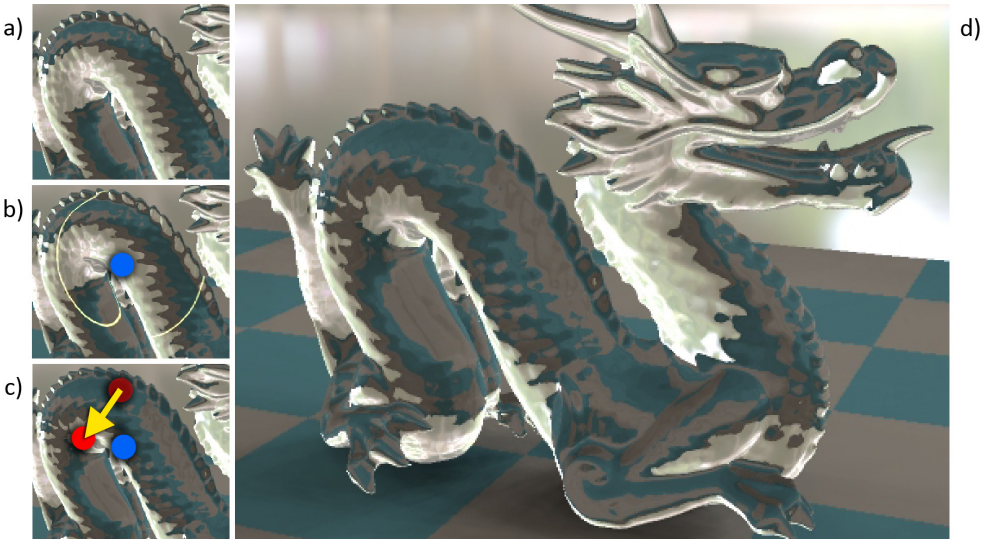

Figure 7.12: Refraction edit (1 constraint, 1 region, 29.1 fps): Starting from the original (a), a region is defined (b), and a constraint is manipulated (c) until the desired result is achieved (d).

As an extension to reflection editing, the system can also be used for interactive refraction editing. The user interface for refraction editing is similar to that of reflection editing. However, refraction constraints define a refracting position and direction, but no refracted position. This is because specifying a point that should be refracted is not unique: a ray could be edited when either entering or exiting the object. In this case, the system reverts to directly manipulating the exiting refraction direction. From a user's perspective, it is not possible to define which world position should be refracted where, but dragging the refraction works as expected. The refracted direction must be specified manually and adjusted until the desired effect for this particular geometry is achieved. In Figure 7.12 an example of refraction editing is shown (at 27 fps).

Complex BRDF

Editing of reflections is possible also for non-mirror, i. e. glossy materials. This is demonstrated in Figure 7.13, where editing a reflection on an object with a Lafortune shading model material is shown. Still, highlights can be moved from the original position to achieve a perceptually plausible, but altered result.

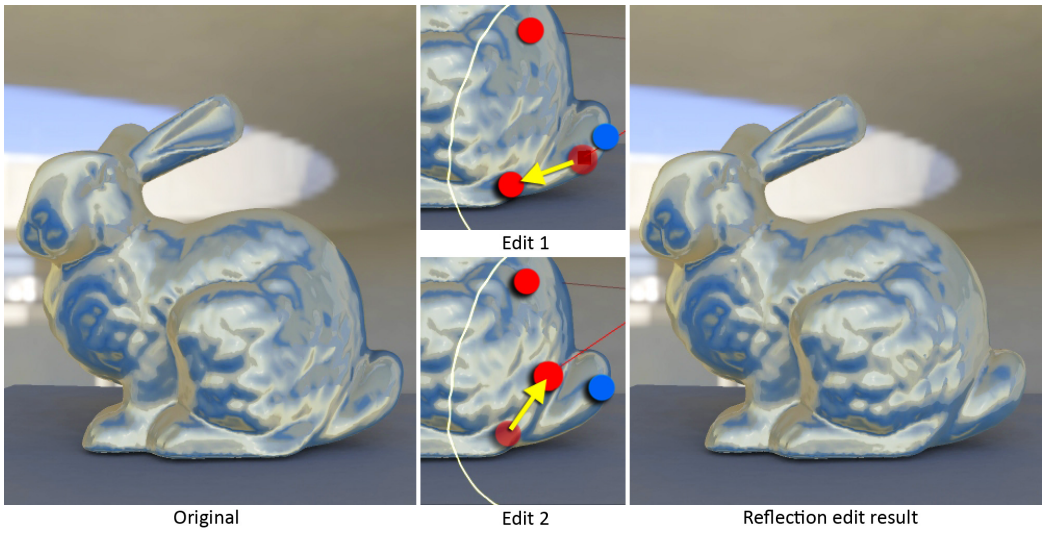

Figure 7.13: Editing a reflection on an object with a glossy Lafortune shading model material. Reflections in objects made of such materials appear blurred. Still, highlights on the tail can be moved from the original (Left) using some simple interactions (Middle) to arrive at a perceptually plausible, but altered result (Right).

7.6.2 Performance

The performance of the reflection editing technique was measured on a 2.4 GHz CPU with an NVIDIA GeForce 260 GTX. Please see Figure 7.6–7.8 for timings. All results are rendered at 1280×960, where reflection editing took less than 100 ms. The most expensive computation is the reflection interpolation, which requires at least one orthogonalization per pixel: a one-constraint system is solved with 24.5 megapixels / s while 32 constraints still result in 18.5 megapixels / s. Moving a geodesic region requires a re-computation of the geodesic distance field on the CPU, which is done at interactive speed (e. g., in 139 ms for the 525 k triangle "Lucy" model used in Figure 7.11).

7.6.3 User Studies

This chapter presents two user studies that were performed to to evaluate the system. In the first study the usability of the system was investigated. In the second study the visual quality of the reflection edits was assessed.

16 subjects, all novice users of the system, participated in the first study. Three where professional 3D graphics designers, while the others did not have much experience in surface modeling or post-production. On average all users rated

their skills in using commercial modeling software 2.75, where a score of 0 is worst and 10 is best. After a detailed tutorial, which took approximately 5 ± 2 min, each subject was given three three-dimensional scenes and corresponding goal images of the target reflections (Figure 7.6, Figure 7.8, and Figure 7.10). The goal images were designed by the author in advance. The subjects were asked to use the system to adjust the reflections in each 3D scene so that they look similar to the goal image. The participants were allowed to work on the task until satisfied. Everybody completed the tasks successfully in a very short time (on average $2{:}22 \pm 1{:}01$ min:sec, $4{:}04 \pm 1{:}44$ min:sec, and $2{:}00 \pm 1{:}16$ min:sec for the task of Figure 7.6, Figure 7.8, and Figure 7.10, respectively). When the users were asked whether the system was useful for achieving the task of the session, the result was an average rating of 9.42 for all tasks and users, where 0 considered worst and 10 best.

The feedback of the three professional 3D graphics designers is especially interesting. Initially, they were asked to go through the same tutorial and tasks as performed by the novice users. Their reactions were positive: the system seems to be the first one that enables them to edit reflections quickly and easily. They were then asked if it would be possible for them to achieve the same tasks using the commercial software that they usually work with. They stated that this would be difficult, then went on to suggest the following possibilities:

Multiple passes It is possible to edit reflections by moving, deforming or changing the reflected objects using the commercial software. However, it is still difficult or impossible if the reflected objects and their reflections are visible in the scene, e. g., the sink reflected on the pot in Figure 7.6. As a result, with this method, the designer has to render the scene at least two times: once as an unedited scene, and the other with edited reflections. The designer then uses a 2D tool to make a composite of the two images (or sequences) by introducing an alpha matte.

Texture Baking It is possible to bake the rendered reflections as the texture over the surface and edit it. However, with this method, it is difficult to change the viewpoint later or manipulate an animation sequence.

Normal editing Several current commercial products support a tool to edit surface normals. However, it is difficult and unintuitive for a designer to predict how the edited surface normals affect the final rendering result. Moreover, normal editing changes the diffuse lighting, which should ideally be unaffected.

Through this first user study, it was shown that, after a short training session, the

user interface for editing reflections is easy to learn as well as intuitive even for novices. The system also solves the problems pointed out by the professional users of conventional tools.

In the second study the visual quality of the edited reflections was evaluated. A total of 9 videos was presented showing a Buddha statue, where 8 videos contained differently edited reflections and 1 video showed the unedited reflections, to 20 subjects. In the videos the virtual camera is orbiting around the Buddha statue so that the reflections can be evaluated from different viewpoints. The subjects were able to activate slow playback or to stop the video to observe the edited reflections very carefully. The Buddha model was chosen because it contains complicated as well as smooth geometry. The supplemental video shows a few seconds of the 9 videos used in this study.

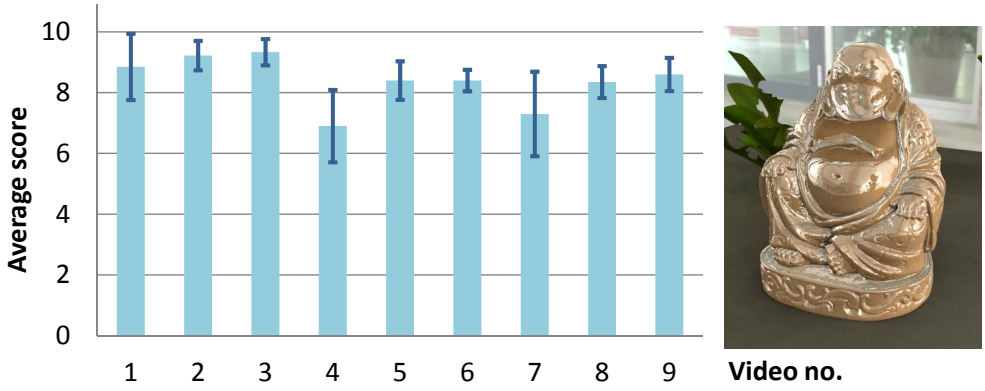

Figure 7.14: Average rating of the perceived physical correctness of different reflection edits on a Buddha statue (0 is worst and 10 is best). Video 9 is the original version of the scene with unedited reflections. The small error bars (dark blue) indicate the standard deviation of the ratings.

The subjects were asked to find reflections that are physically incorrect and rate the physical correctness of reflections on a scale of 0 to 10, where 0 is worst and 10 is best. Figure 7.14 compares the average ratings for each of the 9 videos. As the subjects were aware that the reflections were edited, they were very eager to find the slightest errors. Some participants even thought they found errors in the unedited video, which received an average rating of 8.6 out of 10. As some of the edited versions received even higher average scores than the unedited version, it can be concluded that the edited reflections are usually very hard to detect. Slightly lower average scores were given for video 4 and 7, which both contain strong edits in smooth regions on the belly and the head of the Buddha. This confirms the observations made in literature [Ramanarayanan et al. 2007] that the physical correctness of reflections is harder to assess on more complicated geometry, and edits can be more easily detected on smooth surfaces. When the subjects were

asked what was wrong with the reflection edits they had ranked as low, most answered that the reflections were at the wrong position or appeared deformed. Nobody thought that the edited reflections caused a shape deformation because of the way that the reflections move when the viewpoint is changed. While this result could be expected for the complicated shape of the Buddha statue, however, it must be stressed that local reflection edits on very regular surfaces, like on a flat plane or sphere, can easily result in a perceived shape deformation under changing viewpoints. When subjects were asked if they could detect the unedited version of the video out of the 9 possibilities, only 3 out of 20 participants were willing to make a guess, but all three guessed incorrectly. Finally, we showed all 9 videos simultaneously and asked the subjects to compare the unedited with the edited versions. All subjects were able to identify the performed reflection edits based on a given textual description ("Please tell us, in what video we changed the reflection to cover the right leg."). This proves that the edits were significant enough to alter the appearance of the original.

Through this second user study, it was shown that the edited reflections are usually very hard to detect. Even if, in the case of strong edits, a careful observer notices that the reflections appear deformed or are located in the wrong position, the edits do not cause a perceived surface deformation of the Buddha statue.

The full set-up and analysis of both user studies are detailed in the supplemental material.

7.7 Limitations

The system does have some limitations that may lead to future work. Whether an edit is acceptable is highly dependent on the scene, the camera motion, and the performed edit. Exaggerated edits on smooth surfaces, like on a flat plane or sphere, can readily be assessed as physically incorrect, and can become noticeable under changing viewpoints or within animated scenes. However, reflection edits are very difficult to detect on more complicated surfaces, as verified by the user study.

The system does not currently prevent users from creating non-realistic or unpleasant reflections. Consequently, when using the system, the artist must always check the edits before they can be applied in production. With the proposed real-time system, a user can easily explore the space of possible solutions between physically correct rendering and the artistic goals. Developing criteria for acceptable edits would be challenging future work.

Another limitation occurs if the user specifies too many constraints in a small

region. Because the algorithm tries to fulfill all constraints, the generated field of interpolated reflection direction is no longer smooth and is difficult to control.

The system does not support the bending of reflection rays, which means that it is impossible to reflect objects that are occluded.

Using a real-time ray-tracer as the underlying renderer would allow to experiment with multiple bounces of reflections, multiple refractions, or mixtures of them.

Visually distracting flickering can occur if the red handle (reflecting location) is dragged over a high-frequency surface (e. g., with bump or displacement maps), because the change in reflection direction then also occurs with high frequency. However, this distraction is only encountered while dragging and does not compromise the final result. Optionally, flickering can be suppressed by using a smooth version of the geometry.

It has been shown in practice, that careful placement of region borders, e. g., locating them in areas of high surface curvature, gives more pleasant results. Automatic or guided placement of regions to make edits less objectionable is a possible avenue of future work.

8

Temporal Glare

8.1 Introduction

Glare effects are typical in any optical system used for capturing an image with directly visible bright light sources, caustics, or highlights. Glare is common in our everyday observation of the real world because light scatters in the human eye. Effectively, instead of having a crisp projected image of bright objects on the retina, surrounding regions are affected by scattered light. This leads, among other effects, to local contrast reduction (also called the veiling effect or disability glare). While this is often an unwanted effect in photography [Raskar et al. 2008], it is here exploited for displaying HDR-content on LDR-devices. When a veiling pattern is painted on the image (as a gradient surrounding the light source), the corresponding retinal image is similar to the observation of a real bright object and is thus interpreted by the human brain as brighter [Yoshida et al. 2008]. This effect has been used by artists for centuries to improve apparent dynamic range of their paintings, and it is just as attractive today in a digital imaging context.

A typical glare pattern for a small light source as perceived by most subjects with normal eyes is depicted in Figure 8.1, but the actual appearance of the glare varies with viewing conditions and observers. In general the effects of glare can be divided into *bloom*, a general loss of contrast in the surroundings of the retinal image of the light-source (veil), and *flare* which comprises the *ciliary corona* (the sharp needles) and the *lenticular halo* surrounding the light [Spencer et al. 1995]. Although it cannot be reproduced in static images on paper, people usually report that the glare pattern fluctuates in a fluid-like motion when observed under real world conditions. Additionally, flickering of the fine needles forming the *cilliary corona* is readily observable and many people perceive a pulsation of the glare

Figure 8.1: Glare effect rendered for a point source using the proposed model. The colorful ring is called the *lenticular halo*; the fine radiating needles constitute the *ciliary corona*.

intensity. While these effects are striking, glare has previously only been modeled as a static phenomenon. The dynamics discussed in this chapter do not occur for cameras but only for eyes. In traditional animation, dynamic glare effects are used for artistic effects.

In this work temporal aspects of glare appearance are investigated and simulation of light scattering is performed within the eye based on Fourier optics for high fidelity glare rendering. The goal is not only to render glare realistically with real-time performance, but also to show that by better mimicking the real world experience the image brightness impression and overall perceived quality can be improved. The approach proposed in this chapter is the first to consider a dynamic eye model that allows to simulate the temporal fluctuations of glare. The proposed model is also more complete with respect to existing static models by considering all significant contributions to light scattering in the eye. As an example, the approach model light scattering on particles in the lens nucleus and vitreous humor as well as the grating-like fiber structure in the lens cortex (cf. Figure 8.2). These greatly contribute to the ciliary corona and the lenticular halo (cf. Figure 8.1), but they are ignored by Kakimoto et al.

8.2 A Dynamic Human Eye Model for Glare

This section proposes a novel time-dependent human eye model that is suitable for real-time computation of plausible dynamic glare. It outlines anatomical and physiological characteristics of all parts of the eye that contribute to the light-scattering characteristics of glare. Furthermore, it discusses dynamics shown by these anatomical structures and incorporate their characteristics in the model. Figure 8.2 is a schematic diagram of the human eye. From front to back the optically important parts are the cornea, the aqueous humor, the iris and pupil, the lens, the vitreous humor, and the retina. Table 8.1 summarizes their contribution to scattering, whether they show temporal fluctuations, and whether they are included in the proposed model or not. Multiple scattering is relatively unimportant with respect to glare [van den Berg 1995], and, in single scattering, only the light scattered in the forward direction will reach the sensory system. This work will therefore only consider forward scattering. There are large differences between individual eyes and hence in how glare is perceived by different subjects. In this work, data obtained from studies of the normal healthy eye will be used to fit the model.

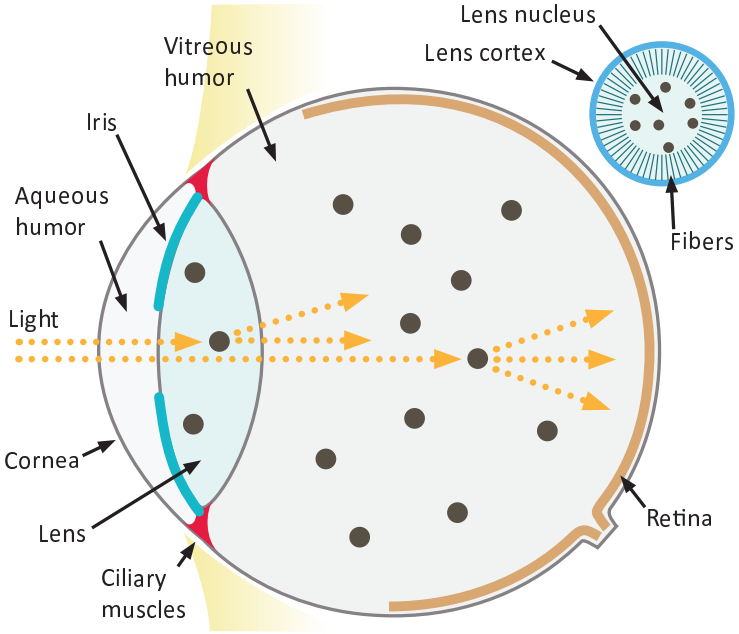

Figure 8.2: Anatomy of the human eye. The upper-right inset shows the lens structure.

8.2.1 The Cornea

The main part of the cornea is formed by collagen fibrils which are densely packed, regularly arranged, cylindrically shaped particles [Freund, McCally and Farrell 1986]. This special arrangement of the fibrils ensures that they are almost transparent [Benedek 1971]. Between the fibrils there is a transparent ground substance, and a few flat cells are interspersed. These flat cells occupy 3–5% of the corneal volume and they have a diameter of 15μm [Freund, McCally and Farrell 1986]. The flat cells contribute to the glare pattern (25-30%, cf. Table 8.1), but the effect is static. This is simulated by having large, sparsely distributed, static particles in the pupil plane.

8.2.2 The Iris and Pupil

A very small percentage (1% or less depending on eye color) is scattered through the iris which is tinted due to absorption [van den Berg, IJspeert and de Waard 1991]. This faint straylight is ignored by the model proposed.

The pupil contributes important diffraction to the glare pattern as it is the aperture of the model. The iris muscles have the ability to control the size of the pupil. Exposition to a glare source will typically give rise to the *pupillary hippus*, an

Eye part	Scatter	Dynamic	Included
Eyelashes [Kakimoto et al. 2004]	varies	yes	yes
Cornea [Vos and Boogaard 1963; Boynton and Clarke 1964]	25-30%	no	yes
Aqueous humor [Wyszecki and Stiles 1982]	none	no	no
Lens [Yuan et al. 1993]	40%	yes	yes
Iris [van den Berg, IJspeert and de Waard 1991]	$\leq$1%	yes	no
Pupil [Fry 1991]	aperture	yes	yes
Vitreous humor [Könz et al. 1995]	10%	yes	yes
Retina [Vos and Bouman 1964]	20%	no	yes

Table 8.1: An overview of the contribution from each part of the eye to the glare phenomenon and to the model. From left to right, the columns describe: the name of the structure, its scattering (percentages refer to the fraction of the total intraocular scattering); whether it is dynamic or not; and whether it is included in the model or not.

involuntary, periodic fluctuation of the pupil size. It is presumably caused by opposing actions of the iris muscles due to the vastly different lighting conditions of glare source and background when attempting to adjust the pupil [Murray, Plainis and Carden 2002]. Curves describing the pupil diameter as a function of time for different glare source intensities have been measured by Fry [1991]. The following expression has been found to mimic these dynamics:

$$h(t,p) = p + \text{noise}\left(\frac{t}{p}\right) \frac{p_{\max}}{p} \sqrt{1 - \frac{p}{p_{\max}}} \quad , \tag{8.1}$$

where t is time (in seconds), p is the mean pupil diameter (in mm) for a given glare source intensity, $p_{\max}$ is the maximum pupil size ($p_{\max} = 9$ mm is used), and noise$(\cdot)$ is a noise function. It remains to map glare source intensity to the mean pupil diameter p. The function proposed by Moon and Spencer [1944] is also used in this work:

$$p = 4.9 - 3 \tanh\left(0.4 \left(\log L_{\text{v}} + 1\right)\right) \quad , \tag{8.2}$$

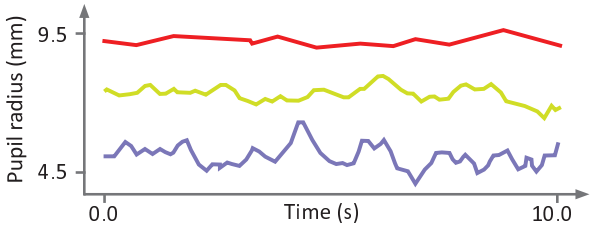

Figure 8.3: Change of pupil size over time for 3 different adaption levels, which effectively depend on of the glare source strength. Bright conditions result in stronger oscillations.

where L_v is the field luminance measured in $^{cd}/_{m^2}$.

The visual effect due to the pupillary hippus is a sort of "pulsation" of the glare pattern. The effect is easily included in the glare model by changing the size of the aperture according to Eqs. 8.1 and 8.2. In Eq. 8.1, value noise with three octaves [Ebert et al. 2003] gave us plausible results. Time-damped average screen intensity is used to approximate the field luminance [Durand and Dorsey 2000].

8.2.3 The Lens

The lens is an important source of intraocular scattering as it accounts for about 40% of the forward scattering (cf. Table 8.1). Close investigation of the refractive index of lens fiber membranes [Michael et al. 2002; Michael and Barraquer 2006] has shown that they produce significant scattering, and that they are regularly spaced in the lens cortex. This high spatial order of the lens fiber lattice increases the transparency of the lens, but is also a diffraction grating which produces the lenticular halo. The refractive index of the fiber membranes decreases towards the center of the lens. This is why the grating is only significant in the lens cortex (as illustrated in the upper right corner of Figure 8.2). The wave optics model easily accounts for the light diffraction on such a grating pattern.

Recently, the ciliary corona has been ascribed to randomly distributed particles in the lens [van den Berg, Hagenouw and Coppens 2005]. Van den Berg et al. explain the sharp needles to be the result of a seamless alignment of scaled copies of the same diffraction pattern originating from light of different wavelengths. However, Hemenger [1992] provides some theoretical evidence that the randomly distributed particles in the lens are too small to produce the ciliary corona on their own. Originally, Simpson [1953] suggested that the ciliary corona is due to particles situated in the vitreous, whose size is comparable to the larger particles in the lens nucleus [Ansari et al. 2001]. This strongly suggests that the ciliary corona is produced by a combination of the few larger particles of the lens nucleus together with particles in the vitreous humor.

This work therefore considers both types of particles and simulates their motion, which is argued to be the main cause of temporal fluctuations observed in the ciliary corona. The motion of the particles in the lens is caused by lens deformations due to accommodative microfluctuations and is quantitatively different from the motion of the particles in the vitreous humor which is mostly inertia-driven. At first the motion of lens particles is modeled and in the following section the vitreous particles are described in more detail.

The human lens has been extensively investigated by physiologists and recently, elaborate numerical finite-element models for the lens' geometry were proposed [Burd, Judge and Cross 2002]. A simplified two-dimensional deformation model is used, assuming a radially symmetric lens with heuristic uniform deformation properties (spring stiffness) and a uniform discretization. This deformation is denoted as $f(\mathbf{x})$, a mapping from two-dimensional coordinate $\mathbf{x}$ to deformed two-dimensional coordinates. The deformation is first computed over a coarse discrete grid using a mass-spring system (cf. Figure 8.4) yielding a discrete approximation $\bar{f}$ to f. To apply f to a high number of lens particles in the continuous three-dimensional domain bi-linear interpolation from $\bar{f}$ is used. To this end, every particle stores a random $\mathbf{x}$ located inside the lens volume (using rejection sampling) and a random ϕ, which is an angle around the axis of symmetry. At runtime, $f(\mathbf{x})$ is looked up for every particle at $\mathbf{x}$ in $\bar{f}$ and is mapped from cylindrical to Cartesian space.

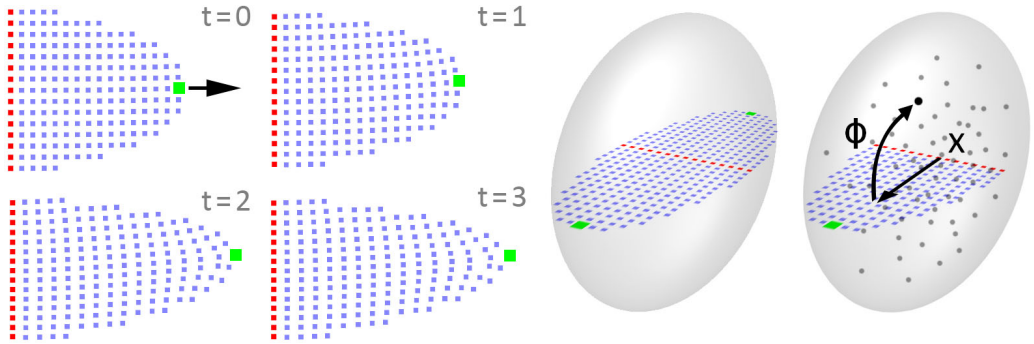

Figure 8.4: *Left:* The lens deformation model simulating ciliary muscle contraction (green element, arrow). *Right:* Mapping of the coarse 2D deformation to smooth 3D deformation.

The accommodation system is known to exhibit temporal variations even during fixation of a stationary stimulus which are due to adaptations of the ciliary muscle [Gray, Winn and Gilmartin 1993]. These contractions of the ciliary muscle are simulated by moving a single element in $\bar{f}$. The oscillations caused by this reflex can be characterized by two principal components in their power-spectrum, one low-frequency (< 0.6 Hz) and one narrow high-frequency (between 1.3 and 2.1

Hz). Since the low-frequency component is known to vary with pupil size (more pronounced for larger size), we use the pupil size as determined in Eq. 8.2 to set the mean power of a low-frequency component generated with three-octave value noise according to experimentally acquired data [Gray, Winn and Gilmartin 1993]. In practice, 17×17 elements for $\bar{f}$, 750 particles of varying size, and 200 gratings are used.

8.2.4 The Vitreous Humor

Scattering by the vitreous humor is usually ignored in glare research since it is rarely a strong discomforting or disabling effect. However, from a graphics perspective, the subtle scattering by the vitreous is important as it contributes some temporal fluctuations to the glare effect. For this reason, the vitreous is included in the proposed model. The vitreous humor contributes relatively little to the glare effect in terms of scattered light energy (cf. Table 8.1), and for this reason it is often ignored in glare research. However, the contribution of scattering on particles in the vitreous to the ciliary corona is clearly visible in less central (less saturated) glare regions. This visibility is reinforced due to temporal effects which are extremely strong attractors of human attention especially in the visual periphery.

The vitreous humor is a slightly scattering viscoelastic body [Zimmerman 1980]. This means that external forces (head movements, saccades) act on the vitreous humor such that it accelerates, rotates and comes to rest by strong damping. This behavior is modeled as a single rigid body with damped rotation dynamics, whose state is determined by angle and angular velocity. The damping is set to approximate the measurements by Zimmerman [1980]. Damped random forces are generated to mimic saccades and integrate this system using forward Euler integration. The final particles are embedded at random but fixed locations inside this rigid body and do not move relatively to each other. Ansari et al. [2001] have measured a three-dimensional map which describes the spatial distribution of particle sizes in the bovine vitreous which exhibits inhomogeneities similar to the human vitreous [Fankhauser 2006]. The scatterers are thus modeled to have uniform random distribution throughout the vitreous.

8.2.5 The Retina

There are convincing arguments that the retina contributes to the intraocular scattering [Vos and Bouman 1964]. It is unfortunately difficult to model the retinal forward scattering directly using a particle distribution because the particles are intermingled with the receptor cells. To find out how the retinal scattering affects the

glare pattern, [Navarro 1985; van den Berg, Hagenouw and Coppens 2005] fitted a model to experimentally acquired glare functions [CIE 1999] which measure the combined effect of all the intraocular scattering. It turned out that fewer and larger particles (compared to measured particle sizes and frequencies) in the preretinal parts of the eye are an excellent way to approximate the measured glare functions. For this reason, fewer and larger particles are used in the model.

8.2.6 Eyelashes and Blinking

Eyelashes and blinking can result in long *streaks* in the glare pattern. Blinking is modeled in the same way as done by Kakimoto et al. [2004]: using bitmaps of eyelids and eyelashes. However, given the dynamic framework, the proposed method can produce animated blinking by moving the eyelashes against the pupil. In addition, squinting which is a normal reaction to a strong, discomforting glare source can be simulated. It decreases the retinal illumination (and thus the discomfort) since the eyelids cut off the aperture and the eyelashes scatter the incident illumination [Sheedy, Truong and Hayes 2003]. Squinting is modeled by keeping the bitmaps closed by a constant amount.

8.3 Wave-Optics Simulation of Light-Scattering

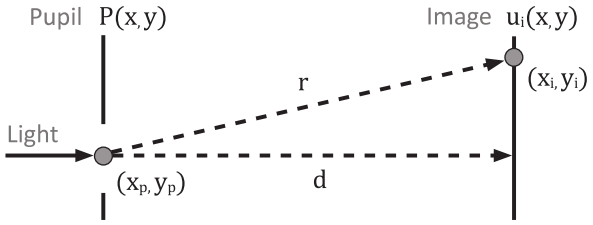

Figure 8.5: Schematic view of an optical system

To simulate the scattering at obstacles within the different anatomical structures of the eye, an approach based on wave-optics similar to van den Berg, Hagenouw and Coppens [2005] and Kakimoto et al. [2004] is used. Even though the underlying theory is rather complicated, the implementation of scattering even on complex apertures is often simple and generally boils down to taking the Fourier transform (FT) of an aperture function. Considering the human eye as a simplified optical system (Figure 8.5) with an aperture (pupil) and a image plane (retina, here assumed to be planar), this work obtains the diffraction pattern for particles and gratings within the lens and the cornea as well as for the eyelashes by computing

the incident radiance following the Fresnel approximation to Huygen's principle [Goodman 2005] given by

$$\begin{aligned} L_{\mathrm{i}}(x_{\mathrm{i}},y_{\mathrm{i}}) &= K\left|\mathcal{F}\left\{P(x_{\mathrm{p}},y_{\mathrm{p}})E(x_{\mathrm{p}},y_{\mathrm{p}})\right\}_{p=\frac{x_i}{\lambda d},q=\frac{y_i}{\lambda d}}\right|^2 \\ K &= 1/(\lambda d)^2 \\ E(x_{\mathrm{p}},y_{\mathrm{p}}) &= e^{i\frac{\pi}{\lambda d}(x_{\mathrm{p}}^2+y_{\mathrm{p}}^2)} \end{aligned} \tag{8.3}$$

for the coordinates $(x_{\mathrm{i}},y_{\mathrm{i}})$ at the retina assuming unit-amplitude, homogeneous incident light. Here $P(x_{\mathrm{p}},y_{\mathrm{p}})$ is the aperture function for the pupil, giving the opacity of each point in the pupil (0 transparent, 1 opaque), λ is the wavelength of the light, d the distance between pupil and retina and $\mathcal{F}$ denotes the Fourier transform that is evaluated at coordinates $(p,q)=\left(\frac{x_{\mathrm{i}}}{\lambda d},\frac{y_{\mathrm{i}}}{\lambda d}\right)$ (for a derivation, see Section 8.5).

In contrast to Kakimoto et al. [2004] who use Fraunhofer diffraction, the present work uses the more general Fresnel equation which contains Fraunhofer diffraction as a special case (Fresnel diffraction should be considered as more appropriate, due to the relatively short distance between pupil and retina [Hecht 1998]). In practice, the aperture function is modified before the FT by complex multiplication with the exponential given in Eq. 8.3 which produced more realistic results in the simulations (Figure 8.6).

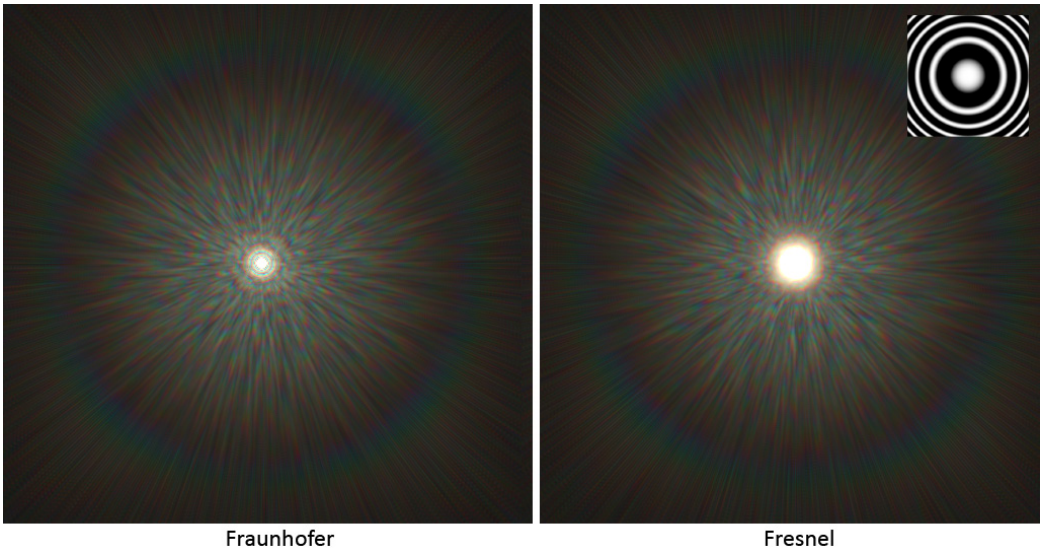

Figure 8.6: Comparison of Fraunhofer and Fresnel diffraction patterns. The Fresnel solution produces a more pronounced PSF. The term E from Eq. 8.3 is shown in the upper right corner.

Simulating the scattering at particles in the vitreous is more complicated because their distance to the retina varies much more. A physically correct approach would

be to compute separate Fresnel-diffraction patterns for each of these particles (because they differ in their distance to the retina) and add them up on an amplitude basis (J. Goodman, pers. comm.). However, this would involve costly operations that depend on the number of particles. Working towards a real-time technique, a single-plane (where we project the particles in the vitreous directly into the pupil-plane) and a more correct multiple-plane approach (where we compute and add multiple diffraction patterns at different distances) were compared. Figure 8.7 gives an impression of the difference of the two approaches. Judging the observable differences to be negligible, it was chosen to implement a single-plane approach to facilitate renderings that run in real-time even though this constitutes a physically incorrect simplification.

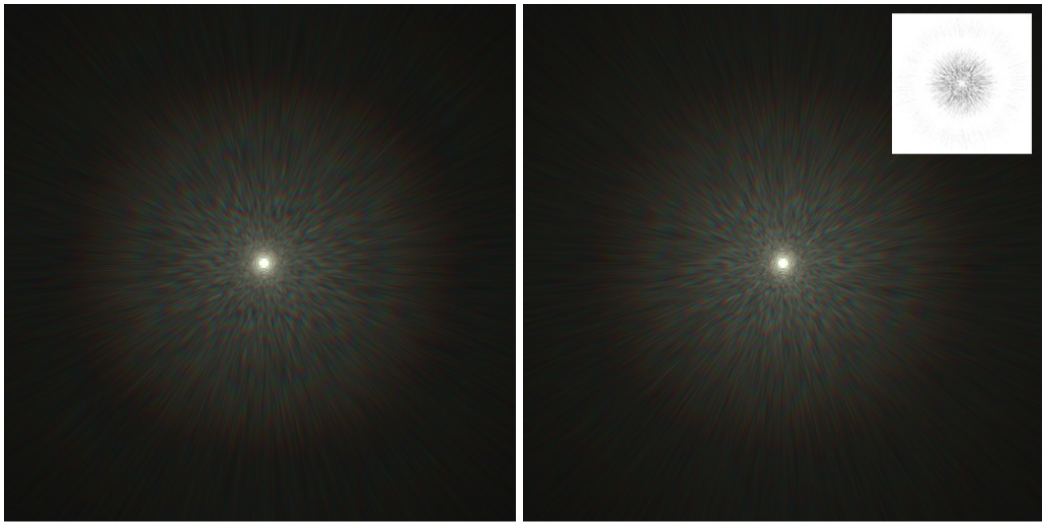

Figure 8.7: Single-plane (Left, 52.4 fps) and multiple-plane diffraction (Right, 32.1 fps) are perceptually equivalent (the difference is in the upper right corner). Using an individual plane for every particle would require 8 s/frame.

8.4 Implementation

The model was implemented to run in real-time entirely on recent graphics hardware (GPUs). Despite the involved theory, temporal glare is easily and efficiently implemented: Draw a few basic drawing primitives, apply an FFT, and do a special kind of blur. This already constitutes a temporal glare pipeline (cf. Figure 8.8). The following paragraphs provide the details.

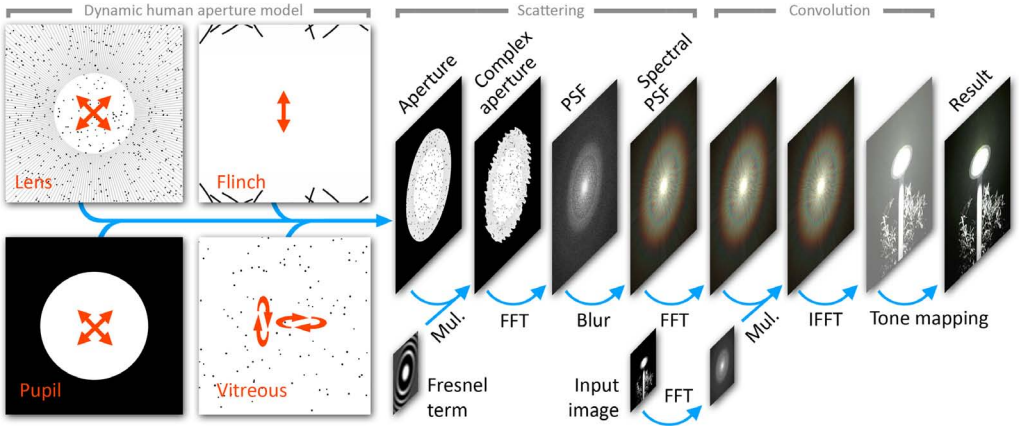

Figure 8.8: The temporal glare pipeline. The blue arrows are GPU shaders that transform one or multiple input textures (rectangles) into one or multiple output textures.

8.4.1 Human Aperture Model

The aperture of the human pupil was simulated by drawing basic primitives into a texture. All primitives are rendered using 2×2 or more super-sampling. This helps suppressing aliasing in time and space which would be exaggerated by the FFT. Figure 8.9 shows partial PSFs that include only subsets of the aperture model in order to show how the individual parts contribute to the result.

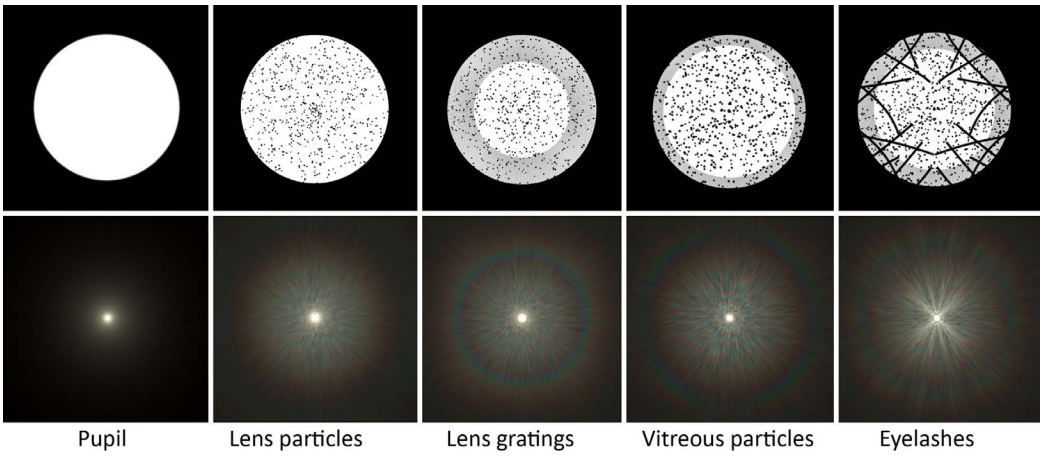

Figure 8.9: Adding scatterers from left to right. Vitreous and lens scattering looks similar, but have different dynamics.

Particles from the lens and the vitreous are projected orthogonally onto their aperture plane and drawn as 2D circles of equal size and color. The implementation uses two sets of parameters. Static parameters encode subject-dependent variables:

the number of particles, eye size and others. Dynamic parameters are updated for every frame: the blink state, the field luminance, the observer motion and others.

First, according to the blink state, the eyelashes and eyelids are drawn as textured quads. Next, the pupil (cf. Sec. 8.2.2) is drawn as a 2D white circle on top of a black background. The radius of this circle is computed by the hippus model (cf. Eq. 8.1) using the field luminance. The next pass adds the lens (cf. Sec. 8.2.3) particles. Then gratings (Sec. 8.2.2) are drawn as lines of a few pixels thickness. Finally, the particles of the vitreous humor are simulated (cf. Sec. 8.2.4) and drawn.

8.4.2 Fresnel Diffraction

The Fresnel diffraction of an aperture texture is computed in two steps. First, the aperture texture is multiplied by the complex exponential E (from Eq. 8.3) which was pre-computed and stored in a static texture. Second, a recent GPU FFT [Moreland and Angel 2003] is applied to the aperture texture computing the $\mathcal{F}$-term from Eq. 8.3. After final normalization by K, the output is the monochromatic PSF F_λ.

8.4.3 Chromatic Blur

The colorful appearance of glare can be modeled in a simple and efficient way [van den Berg, Hagenouw and Coppens 2005]. The system exploits the fact that a monochromatic PSF F_{λ_2} at wavelength λ_2 equals another monochromatic PSF F_{λ_1} for wavelength λ_1 whose argument is scaled by $\frac{\lambda_1}{\lambda_2}$ (cf. Eqn. 8.3):

$$F_{\lambda_2}(\mathbf{x}) = F_{\lambda_1}\left(\frac{\lambda_1}{\lambda_2}\mathbf{x}\right).$$

This equation is only strictly true in the Fraunhofer approximation. In the more exact Fresnel approximation, it would be required to recompute the term E of Eq. 8.3, and thus also the FFT, for each wavelength. To avoid an FFT per wavelength and uphold real-time frame rates, the system computes E for a single wavelength $\lambda_1 = 575\,\text{nm}$ (in the middle of the visual spectrum since the error is smaller the closer λ_2 is to λ_1) and uses this E for all wavelengths. The result is still a significant improvement over Frauenhofer diffraction (which assumes $E = 1$ for all λ).

The spectral PSF F_s for a light with spectrum $s(\lambda)$ is computed as a sum of n scaled

copies of $F_{575\,\mathrm{nm}}$ as:

$$\begin{aligned} F_s(\mathbf{x}) = &\quad \textstyle\sum_{i=0}^{n-1} s(\lambda_i) F_{575\,\mathrm{nm}}(\mathbf{x}_i) \\ \lambda_i = &\quad 380\,\mathrm{nm} + i\frac{770\,\mathrm{nm}-380\,\mathrm{nm}}{n} \\ \mathbf{x}_i = &\quad\quad \mathbf{x}\,\tfrac{575\,\mathrm{nm}}{\lambda_i} \quad . \end{aligned}$$

A fragment program computes F_s by summing n look-ups at different locations in the texture holding $F_{575\,\mathrm{nm}}$. Each look-up is multiplied by the corresponding spectral value $s(\lambda_i)$. Bi-linear interpolation is used to read the discrete $F_{575\,\mathrm{nm}}$ at continuous sample locations $\mathbf{x}_i$. The implementation uses XYZ color space to represent s and F_s, and $n = 32$ resulted in the best tradeoff between quality and speed. Figure 8.1 shows a RGB PSF.

8.4.4 Convolution and Final Display

The system convolves the output image with this RGB PSF. This is different from previous methods, that only used billboards composed onto single bright pixels. Using the FT convolution theorem the convolution is computed as the multiplication of the FFT of the PSF and an FFT of the output image. The computation is done for all RGB channels in parallel using suitable internal texture formats. Applying a final inverse FFT to all channels yields the HDR input image "as seen" from the dynamic human eye model. Figure 8.10 compares the convolution and billboard-based approaches. Note that such distinct appearance of the ciliary corona needles as shown in Figs. 8.1 and 8.10 (left) is typical for bright light sources with angular extent below 20 minutes of arc (the ray-formation angle [Simpson 1953]). Larger light sources superimpose the fine diffraction patterns that constitute the needles of the ciliary corona. This leads to a washing out of their structure as shown in Figure 8.10 (right). However, the temporal glare effect is still visible because the superimposed needles fluctuate incoherently in time.

One simple performance optimization is to pre-compute an animation cycle of RGB PSFs or to omit the convolution [Kakimoto et al. 2004]. Dynamic textures [Soatto, Doretto and Wu 2001] could be used to generate a faithful billboard approximation to dynamic glare with marginal overhead over static glare patterns e. g. for a game. Nevertheless, convolution is used in all results.

For final display gamma mapping is used ($\gamma = 2.2$). While more advanced tone mapping operators could be used instead, choosing this simple approach gives the most reliable and consistent results.

Figure 8.10: PSF applied using (left, 52.4 fps) billboards and (right, 30.2 fps) the convolution proposed. The results are substantially different for this candle. Using convolution, the horizontal needles of the ciliary corona blur out while the vertical needles remain unaffected.

8.5 Derivation of the Fresnel Approximation

One way to derive the Fresnel approximation is using Huygen's principle, which states that a wavefront can be extended by assuming an infinite number of wave-emitters along the wavefront. Assuming the setup in Figure 8.5, the intensity of the light in the image plane $u_{\mathrm{i}}(x_{\mathrm{i}}, y_{\mathrm{i}})$ is expressed as a double integral over the light incident on the pupil $u_p(x_p, y_p)$ where a new harmonic, spherical wave is emitted at each point

$$u_{\mathrm{i}}(x_{\mathrm{i}}, y_{\mathrm{i}}) = \frac{d}{i\lambda} \iint_P u_{\mathrm{p}}(x_{\mathrm{p}}, y_{\mathrm{p}}) \frac{\exp(ikr)}{r^2} \, \mathrm{d}x_{\mathrm{p}} \mathrm{d}y_{\mathrm{p}} \ ,$$

where d is the distance between the pupil and image planes, r is the distance to the source, P the pupil-area over which is integrated, and $k = 2\pi/\lambda$ is the wave number. The Fresnel approximation follows if r is substituted by the first terms of its binomial expansion such that the terms under the integral can be rearranged and expressed as the Fourier transform (FT) of the product of the pupil function and the complex exponential. Assuming unit-amplitude, homogeneous light, we have:

$$u_{\mathrm{i}}(x_{\mathrm{i}}, y_{\mathrm{i}}) = -\frac{i \exp(ikd) \exp(\frac{ik}{2d}(x_i^2 + y_i^2))}{\lambda d} \mathcal{F}\left\{P(x_{\mathrm{p}}, y_{\mathrm{p}}) E(x_{\mathrm{p}}, y_{\mathrm{p}})\right\}$$

Because we are interested in radiance $L_i(x_{\mathrm{i}}, y_{\mathrm{i}}) = |u_{\mathrm{i}}(x_{\mathrm{i}}, y_{\mathrm{i}})|^2$, the term in front of the FT becomes a constant, yielding Eq. 8.3.

8.6 Results

The method results in perceptual improvements as shown by a user study (cf. Sec. 8.6.1) with stimuli running at real-time frame rates (cf. Sec. 8.6.2).

8.6.1 Perceptual Study

To investigate the perceptual quality of the simulations, a psychological study was conducted, comparing the effect of dynamic and static glare renderings in terms of realism, attractiveness and brightness. To provide a standardized setting for the preference study, subjects were simultaneously presented with two images of the same scene where each of the displays was either enhanced by application of dynamic or static glare or not. All possible combinations were presented for each scene. A range of different scenes representing divergent applications (natural images and computer-generated scenes) was chosen to ensure external validity (cf. Figure 8.11). Subjects were asked to choose one of the two displays in a classical two-alternative-forced-choice (2-AFC) task according to one of three instructions: they had to judge which of the two images was brighter, more attractive or more realistic separately and in randomized order. The currently required task was indicated above the display in each trial. 10 naïve and paid subjects participated in the experiment and were seated at distance of 1m from the display. Each image occupied a visual angle of ≈ 10 vis. deg.

The study shows mixed results in terms of realism and attractiveness (see Figure 8.12). The ratings within the individual observer were quite reliable (typically $r = .8$), but they show a large inter-individual variance. This is well explained by the fact that the glare phenomenon is a very personal experience that differs vastly between subjects (due to varying anatomy). This emphasizes, that a final glare rendering is never going to be completely realistic for the whole audience and that care should be taken to choose moderate parameter values for optimal effects. Another moderating factor was the scene to which the glare was applied. While the majority found dynamic glare most attractive for “Gem” and “Trees”, it was mostly disliked for “Park” and “Bridge” (see Figure 8.11).

In the vast majority of trials, subjects reported to perceive dynamic glare as brighter than static ($\chi^2 = 72.2$, $p < .01$) and control ($\chi^2 = 145.8$, $p < .01$) in all scenes. Because of the clear advantage of stronger perceived brightness for dynamic glare rendering that was established in the study qualitatively, additional preliminary data was collected from 4 subjects to quantify this effect. The psychophysical measurement of perceived glare brightness is methodologically difficult and results in a large intra- and inter-subject variance. The present work therefore applies an

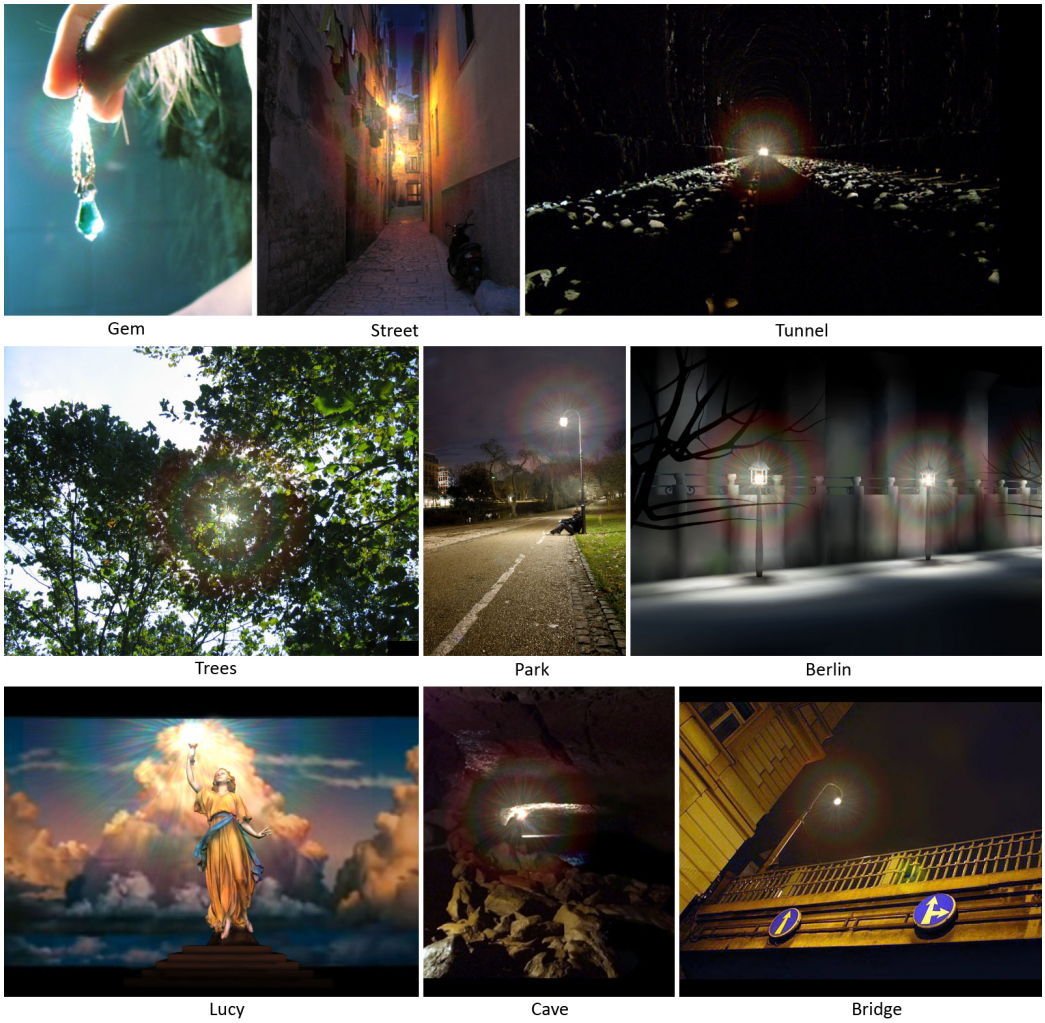

Figure 8.11: The nine stimuli used in the experiment. Please see the video for animated versions.

optimized double-adjustment method as previously proposed [Yoshida et al. 2008]. Subjects had to adjust the brightness of two spots simultaneously to match the perceived brightness of the centered target (which was either without, with static, or with dynamic glare). The mean value of both adjustments was used as the measured perceived luminance of the target. Subjects performed this task for the three different conditions (static, dynamic, control) and two different intensities of the target glare rendering. The results are plotted in Figure 8.12, and they show that both dynamic and static glare produced a boost in increased brightness relative to the control condition ($F(2,4) = 8.22, p < .05$). The apparent advantage of the dynamical over the static glare rendering is not significant due to missing statistical power.

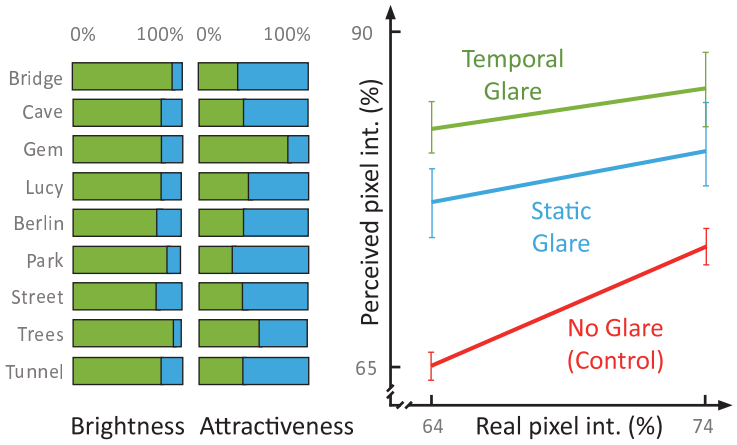

Figure 8.12: *Left:* Relative frequency of decisions for dynamic (green) and static (blue) glare. While dynamic glare is generally perceived as brighter, perceived attractiveness depends strongly on the individual scenes. *Right:* Subjects were well able to adjust the brightness to match control stimuli but perceived a boost in brightness for static and dynamic glare.

8.6.2 Performance

Table 8.2 presents performance numbers for the technique on an 2.4 GHz CPU with an NVIDIA GeForce 8800 GTX.

Size	Aperture	Diffraction	Display	Overall
256	5.2 ms	2.7 ms	1.1 ms	9.0 ms
512	11.8 ms	4.8 ms	1.8 ms	18.4 ms
1024	36.5 ms	13.8 ms	4.8 ms	55.1 ms

Table 8.2: Varying the PSF resolution and timing breakdown. For these timings, 2×2 supersampling of the aperture, 32 samples for the RGB PSF, and no convolution was used

9 Conclusion

The present thesis concludes with this chapter. Following a summary of the three main tracks of contribution – Interactive indirect illumination, Interactive Reflection Editing and Temporal Glare – in Section 9.1, we will discuss their potential relation in a combined computer graphics pipeline in Section 9.2. An outlook into possible future work in Section 9.3 closes this thesis.

9.1 Closing Remarks

In this section we give some concluding remarks on the five contributions made from Chapter 5 to Chapter 8. Because Chapter 4, Chapter 5 and Chapter 6 have similar conclusions and overlapping future work, we will commonly refer to them as "interactive global illumination" in the remaining sections.

9.1.1 Interactive Global Illumination

We have presented three approaches to interactive global illumination: Image Space Directional Occlusion (Chapter 4), Imperfect Shadow Maps (Chapter 5), and Micro-Rendering (Chapter 6), for which concluding remarks are found in this section. All methods have in common to compute global illumination in large and fully dynamic scenes allowing for light, geometry, and material changes at interactive (more than one image per second) or real-time rates. Further, they are temporally coherent, prefer smooth and consistent bias over noise, in contrast to Monte Carlo methods which are unbiased but noisy. Finally, they work without relying too much on geometric constructions like parametrizations or meshing

which are often required by PRT or radiosity.

Micro-Rendering – in contrast to SSDO and ISM, as well as to all the other techniques presented in this thesis – is not mainly motivated by a perceptual intuition. It is more founded in exact reproduction and computation of signals. Not surprisingly it is also the most time-consuming method. Including a perceptual component, e. g. considering materials [Pellacini, Ferwerda and Greenberg 2000] into Micro-Rendering is future work from Section 9.3.

There is a clear order in quality amongst the global illumination techniques proposed: Image Space Directional Occlusion, Imperfect Shadow Maps and Micro-Rendering; a decreasing order in terms of performance but an increasing order in terms of quality. While all of them are interactive for current GPUs, their speed differs by an order of magnitude. SSDO takes some milliseconds ISMs run with many frames per second and Micro-Rendering performs at interactive speed. In terms of quality, it must be admitted that SSDO is a sophisticated image filter, but by definition limited to local effects. Objects that are distant in image space will not exchange light or occlusion, even if they would do in reality. Only ISM can resolve visibility over arbitrary distances, but is limited to mid-sized scenes and diffuse materials. Finally, Micro-Rendering advances over ISM in terms of scalability to huge scenes and arbitrary materials.

It is worth noting, how some combinations of these techniques are more useful compared to others.

Imperfect shadow maps and SSDO are an excellent combination: While ISM catches global visibility and global bounces via VPL lighting, but misses details, SSDO misses global visibility and bounces, but can add arbitrarily small details in image space. Combining both techniques is as easy as multiplying the ISM output with the SSDO output, ideally adjusting ISM's depth bias to match ISDOs sample radius. Note, how this relation is similar to the relation of bump maps [Blinn 1978] and geometry: In a world that is locally flat, geometry details are best described locally by a flat, regular data structure, like a bump map. In the same way, light transport for locally flat surfaces is efficiently simulated in flat, regular image space.

Combinations of SSDO and Micro-Rendering can be used, but do not make as much sense as combining SSDO and ISM. The reason for this is, that Micro-Rendering can have at least one order of magnitude more details in the geometry than ISMs and therefore image space or "flatness" in terms of the last paragraph starts much deeper into the details. Still at this point, image space directional occlusion and bounces could added in a simple ad-hoc post process. A detail that casts an approximate shadow when zooming in is still perceptually more plausible than no light transport at all. More substantial however, this could also lead to

advanced future work, where blockers from image space are combined with a micro-buffer, as discussed in Section 9.3.

ISM and Micro-Rendering are not orthogonal, and do not make much sense together. It would be worthwhile to advance ISMs towards Micro-Rendering. The biggest difference is the missing scalability in ISM: If the scene gets twice as big, it will also render twice as slow ($O(n)$ time complexity, where n is geometric elements). Micro-Rendering however is fully $O(\log n)$ sub-linear. Only comparing the time complexity of $O(\log n)$ vs. $O(n)$ hides the considerable constant factor of Micro-Rendering. The cost for traversing on node in Micro-Rendering can become one order of magnitude bigger than splatting a point in ISM. Also advanced Micro-rendering features (i. e. support for arbitrary materials) increase it's cost. Therefore, making ISM more scalable, such as Micro-Rendering, but with lower constants seems also worthwhile.

9.1.2 Interactive Reflection Editing

Chapter 7 introduced a system for reflection editing. It puts appearance editing in a new perspective, by formally linking artistic goals with physical laws via perception as practiced by traditional artists for centuries. With this system the digital artist of today can specify constraints for reflection positions via an intuitive user interface. If multiple constraints are given, the system optimizes a Moving Least Squares cost function for each pixel to generate the optimal interpolation field of edited reflection directions. The system is easy to implement, works without pre-processing, runs in real-time on modern graphics hardware, and is independent of the underlying reflection rendering algorithm. It is not limited to simple reflections and will also allow more general specular light transport such as glossy reflections and refractions. The applicability of the system was successfully verified by a user study including feedback from professional 3D designers.

9.1.3 Temporal Glare

In Chapter 8 we proposed a model for the dynamics of the human eye and how they add temporal variation to the perceived glare-pattern. The method has a solid basis in eye anatomy and wave optics but remains practical as we demonstrate by implementing it in real-time on recent GPU hardware. A psychophysical study has shown that it improves perceived brightness, suggesting its applicability in a tone-mapping context. Since glare is a phenomenon intrinsic to the individual eye for which no ground truth can be obtained, its validation remains a challenging task.

9.2 Combinations

Interactive global illumination methods, Interactive Reflection Editing and Temporal Glare are orthogonal and form building blocks in a computer graphics pipeline. Certain combinations are worth to be considered in detail.

As Temporal Glare uses an HDR input, and is only perceptually relevant with high contrast, HDR global illumination rendering output as computed by some of our methods is an ideal input to Temporal Glare. For example Micro-Rendering reproduces also indirect specular highlights effectively, that will give rise to (temporal) glare.

A second fruitful combination is to use reflection editing, to edit glossy and indirect light rendered by Micro-Rendering. This can be done, by simply making Micro-Rendering do the gathering in a twisted local frame, as computed by reflection editing.

9.3 Future Work

In this section, possible avenues of future research are discussed.

9.3.1 Interactive Global Illumination

Besides plenty of technical problems that are worth solving, a more in-depth perceptual research of indirect light – or mutual light in perception literature – might also lead to better algorithms. Such research would try to identify which parts of light transport are really understood, and which are not. Further, improving understanding of how the HVS makes use (e. g. for color constancy [Bloj, Kersten and Hurlbert 1999]) of indirect illumination. Until now it has only been considered “eye candy” that was improved using perceptual findings (e. g. [Myszkowski et al. 2001]) that are not strictly addressing perception of indirect illumination but findings of visual perception in general. Are there specific properties of bounced light worth exploration? Is perception of indirect light more related to the “what” system that recognizes and categorizes or is it closer to the “where” pathway which resolves spatial locality?

Screen Space Occlusion An extensions of SSDO to multiple indirect bounces, specular materials, subsurface scattering and importance sampling would be very useful. In future work, it is worth investigation, if an less biased computation

of indirect light is possible in screen space. One could think about omitting the projection step entirely: Instead of using the frame-buffer to find proximity between pixels, an acceleration structure like a hash table could be used to find proximity between points in space. Also a combination between voxels and image space similar to Boubekeur et al. [2006]'s combination of volumes and height fields could be an useful representation of meso-scale bouncers and blockers. The combination of screen-space and object-space approaches remains an interesting avenue of further research bringing together physically plausible illumination of dynamic complex scenes and real-time frame rates.

Imperfect Shadow Maps Imperfect shadow maps are limited to moderately-sized scenes. For larger scenes, performance starts to decrease substantially and without bound. However, the geometry that contributes significantly to indirect shadows seems to be bound. In a future technique, one would need to identify the relevant blockers and only use these. This can be done lazily: The importance of blockers will not change drastically, even when they move quickly. Aila and Laine [2004] and in a recent GPU implementation Sintorn, Eisemann and Assarsson [2008] have used alias-free shadow maps to efficiently deal with the "right" blockers for one light and a primary camera – what is an efficient way to deal with the "right" blockers concerning a camera and a high number of VPLs?

While Micro-Rendering (Chapter 6) uses warping for gathering, the same concept could also be applied to (effective) scattering as in depth maps that belong to VPLs. Depending on the surface material, a glossy VPL sends more light in some direction, in the case of a mirror, in only one. However, current ISMs are agnostic to that fact and store depth maps in a non-adapted parametrization, namely a paraboloid mapping. While this mapping has low distortion, that is desirable for the original context where it was proposed by Brabec, Annen and Seidel [2002] – shadows from (artificial) omni-lights – it might be unsuitable for VPLs. As ISMs usually deal with diffuse materials that correspond to Lambertian emitters, simple replacement of the paraboloid projection by a Lambert projection would make the error proportional to the distributed light. In future work, depth map creation should become similar to the warping in Micro-Rendering, saving texture space or improving accuracy.

As of now, imperfect shadow maps gather from all virtual point lights, regardless if they are important or not. Lightcuts is an efficient method to gather more detailed from important lights [Walter et. al. 2005]. Extensions to lightcuts, try to avoid to compute to many shadow maps [Hašan, Pellacini and Bala 2007]. ISM however, is a method to render many shadow map (rows or columns in terms of Hašan, Pellacini and Bala [2007]'s Matrix Row-Column Sampling) efficiently. Previous

attempts to use lightcuts were limited, because local memory to store and work on cuts is limited for GPUs [Ritschel et. al. 2008a]. Are there, maybe, other ways to achieve the same effect as lightcuts? A starting point could be, to address not all bounds and all functions that constitute the rendering equation, but only sample VPLs according to their distance.

Micro-Rendering Micro-Rendering is geared towards diffuse and glossy surfaces; highly specular surfaces require a sufficient number of gather locations as well as point samples to capture the illumination. We would like to investigate alternatives such as radiance caching on the GPU to handle specular surfaces.

Our hierarchical point representation allows for large and complex geometry; however, scenes with a high depth complexity, such as buildings with many rooms, would benefit from potential visibility techniques [Bittner and Wonka 2003] and efficient culling techniques.

Micro-Rendering has a number of different opportunities for further acceleration. Our implementation is until now only a straightforward implementation of the algorithm without any in-detail performance analysis, beyond what has to be kept in mind for GPU programming generally. For example it is not yet understand if the bottleneck is computation or data transfer when finding the cut.

Another optimization opportunity for Micro-Rendering is to get rid of the micro-framebuffer completely. The micro-framebuffer resides in local memory, which is limited, any potentially is the limiting factor altogether. In other words: Micro-Rendering has a lot of *state* (the micro-framebuffer) while fine-grained parallelism for current GPUs would prefer to have an as-small-as-possible state. To reduce the state-size, we envision to sort the points before splatting them into a micro-buffer, which is possible when using a suitable tree. In such a future approach, we would not store the full micro-framebuffer but just an RGB accumulator that accumulates light, every time a strata is splatted for the first time. Only a small bit vector with one bit (instead of a node index or RGB values) per pixel would be required if the pixel is used for the first time. A pixel set function would be implemented as a bit shift, a logical ors and a conditional add. This is possible, because we only required the convolved end-product, whereas hidden surface removal for usual frame-buffers requires to see reproduce all surface elements resp. pixels.

Micro-Rendering uses a spatial hierarchical data structures (a binary tree) that needs to update in dynamic scenes. In our results, we present dynamic scenes, but limited to deformations that do not drastically change the tree. For this reason, recent results on efficient parallel tree building [Shevtsov, Soupikov and Kapustin 2007; Zhou et al. 2008; Lauterbach et al. 2009] should be applied to Micro-Rendering as

well. Furthermore, as Micro-Rendering is different from ray-tracing in many ways, it is worth to investigate, what the optimal Micro-Rendering tree is. Is SAH really the best approach for Micro-Rendering and ray-tracing at the same time?

Micro-Rendering is currently limited to solid surface and doesn't support transparency. Supporting transparency would become possible by inserting splats in back-to-front-order and alpha blending, but that would require to order the splats. In a different approach, a surface with 25 % opacity, would only generate 25 % of their micro-pixels. After a convolution of the micro-framebuffer with a BRDF, this should be indistinguishable from true transparency in most scenes.

Currently, Micro-Rendering considers all geometry in the upper half-space (Ω^+ in Section 2.3.1) above a piece of surface. Subsurface scattering considers the lower half-space, that is, geometry below a piece of surface, inside an object. Jensen and Buhler [2005] already used a hierarchy of point lights that model subsurface light transport. Their work addresses visibility based on ray-tracing. Combining scattering and visibility in Micro-Rendering, the micro-pixels could be interpreted as differential surface patches that distribute subsurface scattering based on some BSSRDF [Jensen et al. 2001] model, including efficient blocking.

While Micro-Rendering captures many geometric details and scales to much bigger geometry than ISMs, the geometric resolution is still limited. Consequently, surface meso- or micro-structure is not included in light transport, as it is in SSDO. A simple combination of Micro-Rendering would just multiply or add SSDO occlusion or SSDO bounces and Micro-Rendering. In an advanced combination of both techniques, in a first pass one could first run Micro-Rendering to fill a micro-framebuffer with macro-geometry. In a second pass, meso- and micro-structures from image space could be actively combined with the micro-framebuffer, i. e. properly blocking previous macro-information. Only after the second pass, the result is convolved with the BRDF. This is compatible with BRDF warping (cf. Section 6.2.3): Pixels are un-projected in 3D-space and then inserted into the warping before they are written to the micro-framebuffer. By doing so, (procedural) displacement and bump maps would be included in light transport. When done carefully, the proposed extension to transparency is orthogonal.

Micro-Rendering generates one micro-framebuffer in each thread by now. While this allows a lot of parallelism, it might not be the end of the story and much more testing is needed. Recently, approaches based on scheduling have become popular, and surprisingly, although somewhat complex, are able to outperform non-scheduled parallelism, for example in ray-tracing, including incoherent rays [Aila and Laine 2009]. Scheduling here means, that a thread doesn't execute an entire sequence of operations required to arrive at an result. The problem with such an approach is, that inside a warp, all threads will have to wait until the longest

sequence has finished. For ray-tracing, a traversal is broken into individual steps, that inserted and taken out of a work queue. In the same fashion, traversal of the micro-rendering tree could be broken into steps that are inserted into a queue.

Recently, Wang et al. [2009a] used signed distance functions for high-quality interpolation of binary, distant visibility. In Micro-Rendering, an advanced interpolation scheme of the entire incoming radiance (a spherical function) instead of the convolved result (a single value) could be used. One option would be a re-projection of one micro-framebuffer into a new location and orientation, followed by the reflection operator.

9.3.2 Interactive Reflection Editing

A number of opportunities for future work on Interactive Reflection Editing exist. With our real-time system, the user can easily explore the space of possible solutions between physically correct rendering and the artistic goals. Developing criteria for acceptable edits would be challenging future work. It has been shown in practice, that careful placement of region borders, e. g., locating them in areas of high surface curvature, gives more pleasant results. Automatic or guided placement of regions to make edits less objectionable is a possible avenue of future work.

Our system does not support the bending of reflection rays, which means that it is impossible to reflect objects that are occluded. However this would require a renderer that allows to cast curved rays [Gröller 1995]. Using a real-time ray-tracer as the underlying renderer would allow to experiment with multiple bounces of reflections, multiple refractions, or mixtures of them.

In future, it would be possible to generalize the idea of local constraint-based editing beyond physical laws also to other phenomena, such as soft-shadow penumbrae or caustics.

9.3.3 Temporal Glare

For a better understanding of Temporal Glare, we would like to conduct a more detailed study, including more participants to better understand in which conditions temporal glare is important and where not and how individual components of our model contribute to the final result. Future work could elaborate the glare model, e. g. by including real-time subject information like gaze direction. Finally, we would like to consider the relation between depth of field and our glare model.

9.4 Messages

This thesis argued, how synthesizing, editing and enhancing the depiction of three-dimensional virtual scenes can benefit from insight in perception and artistic practice. Most applications targeted by this work — be it feature films, games or visualization — seek to convey a certain message. Such messages are not limited to pure factual information, but more: The relation of lighting mood, related sensations and global illumination might serve as an example. Artistic practice has a long tradition that found efficient ways to convey messages and perception can help to focus computational effort on where it is perceived most. Therefore, as computation time, storage, input data and the media used will always remain limited, combining art and perception into new computer graphics algorithms, such as this thesis did, is a promising visual approach to efficiently convey a message.

Bibliography (Own work)

DONG, Z., GROSCH, T., RITSCHEL, T., KAUTZ, J. AND SEIDEL, H.-P. (2009): Real-time Indirect Illumination with Clustered Visibility. In *Proc. Vision, Modeling and Visualization*

RITSCHEL, T., GROSCH, T., KAUTZ, J. AND MÜLLER, S. (2007): Interactive Illumination with Coherent Shadow Maps. In *Proc. Eurographics Symposium on Rendering*, 61–72

RITSCHEL, T. (2007): Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets. In CIGNONI, P. AND SOCHOR, J., EDITORS: *Short Paper Proceedings of Eurographics 2007*, 17–20

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J. AND DACHSBACHER, C. (2009a): Micro-Rendering for Scalable, Parallel Final Gathering. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 28 (5)

RITSCHEL, T., GROSCH, T., KAUTZ, J. AND SEIDEL, H.-P. (2008a): Interactive Global Illumination based on Coherent Surface Shadow Maps. In SHAW, C. AND BARTRAM, L., EDITORS: *Proc. Graphics Interface*, 185–192

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C. AND KAUTZ, J. (2008b): Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 27 (5)

RITSCHEL, T., GROSCH, T. AND SEIDEL, H.-P. (2009): Approximating Dynamic Global Illumination in Image Space. In SPENCER, S. N., EDITOR: *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 75–82

RITSCHEL, T., IHRKE, M., FRISVAD, J. R., COPPENS, J., MYSZKOWSKI, K. AND SEIDEL, H.-P. (2009b): Temporal Glare: Real-Time Dynamic Simulation of the Scattering in the Human Eye. *Computer Graphics Forum (Proceedings Eurographics 2009)* 28 (2)

RITSCHEL, T., OKABE, M., THORMÄHLEN, T. AND SEIDEL, H.-P. (2009c): Interactive Reflection Editing. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH Asia)* 28 (5)

RITSCHEL, T., SMITH, K., IHRKE, M., GROSCH, T., MYSZKOWSKI, K. AND SEIDEL, H.-P. (2008c): 3D Unsharp Masking for Scene Coherent Enhancement. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 27 (3)

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 Trans. on Applied Perception (Proc. APGV)*, 6, 1–14

Bibliography

AGARWAL, S., RAMAMOORTHI, R., BELONGIE, S. AND JENSEN, H. W. (2003): Structured Importance Sampling of Environment Maps. *ACM Trans. on Graphics*, 22 (3), 605–612

AILA, T. AND LAINE, S. (2004): Alias-Free Shadow Maps. In *Proc. Eurographics Symposium on Rendering*, 161–166

AILA, T. AND LAINE, S. (2009): Understanding the Efficiency of Ray Traversal on GPUs. In *Proc. ACM SIGGRAPH / Eurographics Conference on High Performance Graphics*, 145–149

AKENINE-MÖLLER, T., HAINES, E. AND HOFFMAN, N. (2008): Real-Time Rendering 3rd Edition.

ANJYO, K.-I. AND HIRAMITSU, K. (2003): Stylized Highlights for Cartoon Rendering and Animation. *IEEE Comput. Graph. Appl.* 23 (4), 54–61

ANNEN, T., KAUTZ, J., DURAND, F. AND SEIDEL, H.-P. (2004): Spherical Harmonic Gradients for Mid-Range Illumination. In *Proc. Eurographics Symposium on Rendering*, 331–336

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., SEIDEL, H.-P. AND KAUTZ, J. (2008): Real-Time, All-Frequency Shadows in Dynamic Scenes. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 27 (3), 34:1–34:8

ANSARI, R. R., SUH, K. I., DUNKER, S., KITAYA, N. AND SEBAG, J. (2001): Quantitative Molecular Characterization of Bovine Vitreous and Lens with Non-invasive Dynamic Light Scattering. *Experimental Eye Research*, 73 (6), 859–866

APPEL, A. (1968): Some Techniques for Shading Machine Renderings of Solids. In *AFIPS '68 (Spring): Proc. of the April 30–May 2, 1968, spring joint computer conference*, 37–45

ARIKAN, O., FORSYTH, D. A. AND O'BRIEN, J. F. (2005): Fast and Detailed Approximate Global Illumination by Irradiance Decomposition. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 1108–1114

ARVO, J., TORRANCE, K. AND SMITS, B. (1994): A Framework for the Analysis of Error in Global Illumination Algorithms. In *Proc. ACM SIGGRAPH*, 75–84

AUPPERLE, L. AND HANRAHAN, P. (1993): A Hierarchical Illumination Algorithm for Surfaces with Glossy Reflection. In *Proc. ACM SIGGRAPH*, 155–162

BAUM, D. R., RUSHMEIER, H. E. AND WINGET, J. M. (1989): Improving Radiosity Solutions Through the Use of Analytically Determined Form-factors. In *Proc. ACM SIGGRAPH*, 325–334

BAVOIL, L., SAINZ, M. AND DIMITROV, R. (2008): Image-space Horizon-based Ambient Occlusion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 talks*, 1–1

BENEDEK, G. B. (1971): Theory of Transparency of the Eye. *Applied Optics*, 10 (3), 459–473

BERG, T. J. T. P. VAN DEN (1995): Analysis of Intraocular Straylight, Especially in Relation to Age. *Optometry and Vision Science*, 72 (2), 52–59

BERG, T. J. T. P. VAN DEN, HAGENOUW, M. P. J. AND COPPENS, J. E. (2005): The Ciliary Corona: Physical Model and Simulation of the Fine Needles Radiating from Point Light Sources. *Investigative Ophthalmology & Visual Science*, 46, 2627–2632

BERG, T. J. T. P. VAN DEN, IJSPEERT, J. K. AND WAARD, P. W. T. DE (1991): Dependence of Intraocular Straylight on Pigmentation and Light Transmission Through the Ocular Wall. *Vision Research*, 31 (7-8), 1361–1367

BITTNER, J. AND WONKA, P. (2003): Visibility in Computer Graphics. *Environment and Planning B: Planning and Design*, 30 (5), 729–756

BLAKE, A. AND BÜLTHOFF, H. (1990): Does the Brain Know the Physics of Specular Reflection? *Nature*, 343, 165–168

BLINN, J. F. (1978): Simulation of Wrinkled Surfaces. *Computer Graphics (Proc. ACM SIGGRAPH)*, 12 (3), 286–292

BLINN, J. F. AND NEWELL, M. E. (1976): Texture and Reflection in Computer Generated Images. *Commun. ACM*, 19 (10), 542–547

BLOJ, M. AND RUPPERTSBERG, A. I. (2006): The Role of Mutual Illumination in Gradient Formation. *J. Vision*, 6, 246

BLOJ, M. G. AND HURLBERT, A. C. (1995): Does Mutual Illumination Improve Human Colour Constancy? *Invest. Ophthalmol. Vis. Sci.* 36, 639

BLOJ, M. G., KERSTEN, D. AND HURLBERT, A. (1999): Perception of Three-dimensional Shape Influences Colour Perception Through Mutual Illumination. *Nature*, 402, 877–879

BLOJ, M., RIPAMONTI, C., MITHA, K., HAUCK, R., GREENWALD, S. AND BRAINARD, D. H. (2004): An Equivalent Illuminant Model for the Effect of Surface Slant on Perceived Lightness. *Journal of Vision*, 4 (9), 735–746

BOUBEKEUR, T., HEIDRICH, W., GRANIER, X. AND SCHLICK, C. (2006): Volume-Surface Trees. *Comput. Graph. Forum (Proc. Eurographics)*, 25 (3), 399–406

BOYNTON, R. M. AND CLARKE, F. J. J. (1964): Sources of Entoptic Scatter in the Human Eye. *Journal of the Optical Society of America*, 54 (1), 110–119

BRABEC, S., ANNEN, T. AND SEIDEL, H.-P. (2002): Shadow Mapping for Hemispherical and Omnidirectional Light Sources. In *Proc. Computer Graphics International*, 397–408

BRAHAM, A. (1976): The Rokeby Venus, Velázquez.

BUNNELL, M. (2005): Dynamic Ambient Occlusion and Indirect Lighting. In PHARR, M., EDITOR: *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*

BURD, H., JUDGE, S. AND CROSS, J. (2002): Numerical Modelling of the Accommodating Lens. *Vision Research*, 42 (18), 2235–2251

CHESLACK-POSTAVA, E., WANG, R., AKERLUND, O. AND PELLACINI, F. (2008): Fast, Realistic Lighting and Material Design using Nonlinear Cut Approximation. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 27 (5), 128

CHRISTENSEN, P., LAUR, D., FONG, J., WOOTEN, W. AND BATALI, D. (2003): Ray Differentials and Multiresolution Geometry Caching for Distribution Ray Tracing in Complex Scenes. In *Proc. Eurographics Symposium on Rendering*, 543–552

CHRISTENSEN, P. H. (2008): Point-Based Approximate Color Bleeding. Pixar – Technical report

CIE (1999): Collection 1999: Vision and Colour, Physical Measurement of Light and Radiation. Commission Internationale de l'Eclairage (135–1999). – Technical report

COHEN, M. F. AND GREENBERG, D. P. (1985): The Hemi-Cube: A Radiosity Solution for Complex Environments. In *Proc. ACM SIGGRAPH*, 31–40

COHEN, M. F., WALLACE, J. AND HANRAHAN, P. (1993): Radiosity and Realistic Image Synthesis.

COHEN-OR, D., CHRYSANTHOU, Y., SILVA, C. AND DURAND, F. (2003): A Survey of Visibility for Walkthrough Applications. *IEEE Trans. on Visualization and Computer Graphics*, 9 (3), 412–431

COLBERT, M., PATTANAIK, S. AND KRIVÁNEK, J. (2006): BRDF-Shop: Creating Physically Correct Bidirectional Reflectance Distribution Functions. *IEEE Comput. Graph. Appl.* 26 (1), 30–36

CORBALLIS, M. C. AND MCLAREN, R. (1984): Winding One's Ps and Qs: Mental Rotation and Mirror-Imagediscrimination. *Journal of Experimental Psychology: Human Perception and Performance*, 10, 318–327

CROW, F. (1977): Shadow Algorithms for Computer Graphics. In *Computer Graphics (Proc. ACM SIGGRAPH)*, 242–248

DACHSBACHER, C. AND STAMMINGER, M. (2005): Reflective Shadow Maps. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 203–213

DACHSBACHER, C. AND STAMMINGER, M. (2006): Splatting Indirect Illumination. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 93–100

DACHSBACHER, C., STAMMINGER, M., DRETTAKIS, G. AND DURAND, F. (2007): Implicit Visibility and Antiradiance for Interactive Global Illumination. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), 1–10

DACHSBACHER, C. AND STAMMINGER, M. (2003): Translucent Shadow Maps. In *Proc. Eurographics Workshop on Rendering*, 197–201

DACHSBACHER, C., VOGELGSANG, C. AND STAMMINGER, M. (2003): Sequential Point Trees. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 22 (3), 657–662

DALY, S. (1993): The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity. In *Proc. SPIE: Digital Images and Human Vision*, 179–206

DEBATTISTA, K., SUNDSTEDT, V., SANTOS, L. P. AND CHALMERS, A. (2005): Selective Component Based Rendering. In *Proc. Graphite 2005*

DONG, Z., KAUTZ, J., THEOBALT, C. AND SEIDEL, H.-P. (2007): Interactive Global Illumination Using Implicit Visibility. In *Proc. Pacific Graphics*, 77–86

DÖRSCHNER, K., B. H. M. L. T. (2004): Human Observers Compensate for Secondary Illumination Originating in Nearby Chromatic Surfaces. *Journal of Vision*, 4, 92–105

DURAND, F. (2000): A Multidisciplinary Survey of Visibility.

DURAND, F. AND DORSEY, J. (2000): Interactive Tone Mapping. In *Proc. Eurographics Workshop on Rendering*, 219–230

DURAND, F., HOLZSCHUCH, N., SOLER, C., CHAN, E. AND SILLION, F. X. (2005): A Frequency Analysis of Light Transport. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 1115–1126

DUTRÉ, P., BEKAERT, P. AND BALA, K. (2003): Advanced Global Illumination., 340

EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K. AND WORLEY, S. (2003): Texturing & Modeling: A Procedural Approach. 3rd edition.

EISEMANN, E. AND DÉCORET, X. (2006): Fast Scene Voxelization and Applications. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 71–78

EISEMANN, E. AND DURAND, F. (2004): Flash Photography Enhancement via Intrinsic Relighting. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 23 (3), 673–678

EVERITT, C. (2001): Introduction Interactive Order-Independent Transparency. NVidia Technical Report

FANKHAUSER, F. (2006): Dynamic Light Scattering in Ophthalmology: Results of In Vitro and In Vivo experiments. *Technology & Health Care*, 14 (6), 521–535

FERWERDA, J. A., SHIRLEY, P., PATTANAIK, S. N. AND GREENBERG, D. P. (1997): A Model of Visual Masking for Computer Graphics. In *Proc. ACM SIGGRAPH*, 143–152

FILION, D. AND MCNAUGHTON, R. (2008): Effects & Techniques. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, 133–164

FLEMING, R. AND BÜLTHOFF, H. (2005): Low-Level Images Cues in the Perception of Translucent Materials. *ACM Trans. on Applied Perception*, 2, 346–382

FLEMING, R. W., DROR, R. O. AND ADELSON, E. H. (2003): Real-world Illumination and the Perception of Surface Reflectance Properties. *J. Vis.* 3 (5), 347–368

FORSYTH, D. AND ZISSERMAN, A. (1990): Shape From Shading in the Light of Mutual Illumination. *Image Vision Comput.* 8 (1), 42–49, ISSN 0262–8856

FRANKLIN, A. (2006): Hardware-based Ambient Occlusion. In ENGEL, W., EDITOR: *ShaderX4: Advanced Rendering Techniques*, 91–100

FREUND, D. E., MCCALLY, R. L. AND FARRELL, R. A. (1986): Effects of Fibril Orientations on Light Scattering in the Cornea. *Journal of the Optical Society of America A*, 3 (11), 1970–1982

FRY, G. A. (1991): The Relation of Pupil Constriction Experienced Under Discomfort Glare. In ADRIAN, W., EDITOR: *Proc. of the First International Symposium on Glare*, 173–181

FUNKHOUSER, T., JOT, J.-M. AND TSINGOS, N. (2002): "Sounds Good to Me!" Computational Sound for Graphics, Virtual Reality, and Interactive Systems. SIGGRAPH Course Notes

GASSENBAUER, V., KŘIVÁNEK, J. AND BOUATOUCH, K. (2009): Spatial Directional Radiance Caching. *Computer Graphics Forum*, 28 (4), 1189–1198

GAUTRON, P., KRIVÁNEK, J., BOUATOUCH, K. AND PATTANAIK, S. (2005): Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm. In *Proc. Eurographics Symposium on Rendering*, 55–64

GIBSON, J. J. (1950): The Perception of the Visual World.

GINGOLD, Y. AND ZORIN, D. (2008): Shading-based Surface Editing. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 27 (3), 1–9

GOODMAN, J. (2005): Introduction To Fourier Optics.

GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P. AND BATTAILE, B. (1984): Modeling the Interaction of Light Between Diffuse Surfaces. In *Proc. ACM SIGGRAPH*, 213–222

GORTLER, S. J., SCHRÖDER, P., COHEN, M. F. AND HANRAHAN, P. (1993): Wavelet Radiosity. In *Proc. ACM SIGGRAPH*, 221–230

GRAY, L. S., WINN, B. AND GILMARTIN, B. (1993): Accommodative Microfluctuations and Pupil Diameter. *Vision Research*, 33 (15), 2083–2090

GREEN, P., KAUTZ, J. AND DURAND, F. (2007): Efficient Reflectance and Visibility Approximations for Environment Map Rendering. *Comput. Graph. Forum (Proc. Eurographics)*, 26 (3), 495–502

GRÖLLER, E. (1995): Nonlinear Ray Tracing: Visualizing Strange Worlds. *The Visual Computer*, 11 (5), 263–274

GROSSMAN, J. AND DALLY, W. (1998): Point Sample Rendering. In *Proc. Eurographics Workshop on Rendering*, 181–192

GUENNEBAUD, G., BARTHE, L. AND PAULIN, M. (2006): Real-time Soft Shadow Mapping by Backprojection. In *Proc. Eurographics Symposium on Rendering*, 227–234

GUTIERREZ, D., SERON, F. J., LOPEZ-MORENO, J., SANCHEZ, M. P., FANDOS, J. AND REINHARD, E. (2008): Depicting Procedural Caustics in Single Images. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 27 (5), 120

HAGEN, H., HAHMANN, S., SCHREIBER, T., NAKAJIMA, Y., WORDENWEBER, B. AND HOLLEMANN-GRUNDSTEDT, P. (1992): Surface Interrogation Algorithms. *IEEE Comput. Graph. Appl.* 12 (5), 53–60

HAIDINGER, W. (1844): Über das directe Erkennen des polarisirten Lichts und der Lage der Polarisationsebene. *Annalen der Physik und Chemie*, 63, 29–39

HANRAHAN, P., SALZMAN, D. AND AUPPERLE, L. (1991): A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (Proc. ACM SIGGRAPH)*, 25 (4), 197–206

HASENFRATZ, J.-M., LAPIERRE, M., HOLZSCHUCH, N. AND SILLION, F. (2003): A Survey of Real-Time Soft Shadows Algorithms. *Computer Graphics Forum*, 22 (4), 753–774

HAŠAN, M., KRIVÁNEK, J., WALTER, B. AND BALA, K. (2009): Virtual Spherical Lights for Many-Light Rendering of Glossy Scenes. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 28 (5), 1–6

HAŠAN, M., PELLACINI, F. AND BALA, K. (2007): Matrix Row-Column Sampling for the Many-Light Problem. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), 26

HECHT, E. (1998): Optics. 3rd edition.

HEGEMAN, K., PREMOZE, S., ASHIKHMIN, M. AND DRETTAKIS, G. (2006): Approximate Ambient Occlusion For Trees. In SEQUIN, C. AND OLANO, M., EDITORS: *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* ACM SIGGRAPH

HEIDRICH, W. AND SEIDEL, H. (1999): Realistic, Hardware-accelerated Shading and Lighting. In *Proc. ACM SIGGRAPH*, 171–178

HELMHOLTZ, H. (1867): Handbuch der Physiologischen Optik.

HEMENGER, R. (1992): Sources of Intraocular Light Scatter from Inversion of an Empirical Glare Function. *Applied Optics*, 31 (19), 3687–3693

HERZOG, R., KINUWAKI, S., MYSZKOWSKI, K. AND SEIDEL, H.-P. (2008): Render2MPEG: A Perception-based Framework towards Integrating Rendering and Video Compression. *Comput. Graph. Forum (Proc. Eurographics)*, 27, 183–192

HOBEROCK, J. AND JIA, Y. (2007): High-Quality Ambient Occlusion. In *GPU Gems 3* . – chapter 12

HOPPE, H. (1996): Progressive Meshes. In *Proc. ACM SIGGRAPH*, 99–108

HORN, B. K. P. (1987): Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4 (4), 629–642

HUBONA, G. S., WHEELER, P. N., SHIRAH, G. W. AND BRANDT, M. (1999): The Role of Object Shadows in Promoting 3D Visualization. *ACM Trans. on Computer-Human Interaction*, 6, 214–242

HUNTER, R. S. AND HAROLD, R. W. (1987): The Measurement of Appearance.

IGARASHI, T., MOSCOVICH, T. AND HUGHES, J. F. (2005): As-rigid-as-possible Shape Manipulation. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 1134–1141

ITTELSON, W. H., MOWAFY, L. AND MAGID, D. (1991): The Perception of Mirror-reflected Objects. *Perception*, 20, 567–584

IWASAKI, K., DOBASHI, Y., YOSHIMOTO, F. AND NISHITA, T. (2007): Precomputed Radiance Transfer for Dynamic Scenes Taking into Account Light Interreflection. In *Proc. Eurographics Symposium on Rendering*, 35–44

JENSEN, H. W. (1995): Importance Driven Path Tracing using the Photon Map. In *Proc. ESGR*, 326–335

JENSEN, H. W. (1996): Global Illumination Using Photon Maps. In *Proc. Eurographics Symposium on Rendering*, 21–30

JENSEN, H. W. AND BUHLER, J. (2005): A Rapid Hierarchical Rendering Technique for Translucent Materials. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, 12

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M. AND HANRAHAN, P. (2001): A Practical Model for Subsurface Light Transport. In *Proc. ACM SIGGRAPH*, 511–518

KAJIYA, J. (1986): The Rendering Equation. *Computer Graphics (Proc. ACM SIGGRAPH)*, 20 (4), 143–150

KAKIMOTO, M., MATSUOKA, K., NISHITA, T., NAEMURA, T. AND HARASHIMA, H. (2004): Glare Generation Based on Wave Optics. In *Proc. Pacific Graphics*, 133–142

KAUTZ, J., BOULOS, S. AND DURAND, F. (2007): Interactive Editing and Modeling of Bidirectional Texture Functions. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), 53

KAWASE, M. (2005): Practical Implementation of High Dynamic Range Rendering. In *Game Developers Conference*

KELLER, A. (1997): Instant Radiosity. In *Proc. ACM SIGGRAPH*, 49–56

KERR, W. B. AND PELLACINI, F. (2009): Toward Evaluating Lighting Design Interface Paradigms for Novice Users. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 28 (3), 1–9

KERSTEN, D., KNILL, D. C., MAMASSIAN, P. AND BÜLTHOFF, I. (1996): Illusory Motion from Shadows. *Nature*, 379, 31

KERSTEN, D., MAMASSIAN, P. AND KNILL, D. C. (1997): Moving Cast Shadows and the Perception of Relative Depth. *Perception*, 26, 171–192

KHAN, E. A., REINHARD, E., FLEMING, R. W. AND BÜLTHOFF, H. H. (2006): Image-based Material Editing. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 25 (3), 654–663

KIRK, A. G. AND ARIKAN, O. (2007): Real-Time Ambient Occlusion for Dynamic Character Skins. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*

KONTKANEN, J. AND AILA, T. (2006): Ambient Occlusion for Animated Characters. In *Proc. Eurographics Symposium on Rendering*

KONTKANEN, J. AND LAINE, S. (2005): Ambient Occlusion Fields. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 41–48

KÖNZ, F., RICKA, J., FRENZ, M. AND FANKHAUSER, F. (1995): Dynamic Light Scattering in the Vitreous: Performance of the Single-Mode Fiber Technique. *Optical Engineering*, 34 (8), 2390–2395

KOPF, J., COHEN, M., LISCHINSKI, D. AND UYTTENDAELE, M. (2007): Joint Bilateral Upsampling. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), Article No. 96

KOPRA, A. (2007): Writing Mental Ray Shaders: A Perceptual Introduction.

KOZLOWSKI, O. AND KAUTZ, J. (2007): Is Accurate Occlusion of Glossy Reflections Necessary? In *Proc. ACM Symposium on Applied Perception in Graphics and Visualization*, 91–98

KUFFLER, S. W. (1953): Discharge Patterns and Functional Organization of Mammalian Retina. *J. Neurophysiol.* 16, 37–68

KŘIVÁNEK, J., BOUATOUCH, K., PATTANAIK, S. N. AND ZARA, J. (2006): Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping. In *Proc. Eurographics Symposium on Rendering*

KŘIVÁNEK, J., GAUTRON, P., PATTANAIK, S. AND BOUATOUCH, K. (2005): Radiance Caching for Efficient Global Illumination Computation. *IEEE TVCG*, 11 (5), 550–561

LAFORTUNE, E., FOO, S.-C., TORRANCE, K. AND GREENBERG, D. (1997): Non-Linear Approximation of Reflectance Functions. In *Proc. ACM SIGGRAPH*, 117–126

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J. AND AILA, T. (2007): Incremental Instant Radiosity for Real-Time Indirect Illumination. In *Proc. Eurographics Symposium on Rendering*, 277–286

LANDIS, H. (2002): RenderMan in Production. In *ACM SIGGRAPH 2002 Course 16*

LANGER, M. S. AND BÜLTHOFF, H. H. (2000): Depth Discrimination From Shading Under Diffuse Lighting. *Perception*, 29 (6), 649 – 660

LARSON, G. W., RUSHMEIER, H. AND PIATKO, C. (1997): A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. *IEEE Trans. on Visualization and Computer Graphics*, 3 (4), 291–306

LAUTERBACH, C., GARLAND, M., SENGUPTA, S., LUEBKE, D. AND MANOCHA, D. (2009): Fast BVH Construction on GPUs. *Comput. Graph. Forum (Proc. Eurographics)*, 28 (2), 375–384

LEHTINEN, J. AND KAUTZ, J. (2003): Matrix Radiance Transfer. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 59–64

LEHTINEN, J., ZWICKER, M., TURQUIN, E., KONTKANEN, J., DURAND, F., SILLION, F. AND AILA, T. (2008): A Meshless Hierarchical Representation for Light Transport. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 27 (3), 37

LEWIS, R. (1993): Making Shaders More Physically Plausible. In *Proc. Eurographics Workshop on Rendering*, 47–62

LISCHINSKI, D. AND RAPPOPORT, A. (1998): Image-Based Rendering for Non-Diffuse Synthetic Scenes. In *Proc. Eurographics Symposium on Rendering*, 301–314

LUFT, T., COLDITZ, C. AND DEUSSEN, O. (2006): Image Enhancement by Unsharp Masking the Depth Buffer. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 25 (3), 1206–1213

MA, W.-C., HSIAO, C.-T., LEE, K.-Y., CHUANG, Y.-Y. AND CHEN, B.-Y. (2006): Real-time Triple Product Relighting Using Spherical Local-frame Parameterization. *Vis. Comput.* 22 (9), 682–692

MALONEY, L. T. (1999): Physics-based Approaches to Modelling Surface Color Perception. In GEGENFURTNER, K. T. AND SHARPE, L. T., EDITORS: *Color Vision: From Genes to Perception*, 387–422

MAMASSIAN, P., KNILL, D. C. AND KERSTEN, D. (1998): The Perception of Cast Shadows. *Trends in Cognitive Sciences*, 8, 288–295

MANTIUK, R., PATTANAIK, S. AND MYSZKOWSKI, K. (2002): Using Environment Mapping for Interactive Global Illumination Computation in Dynamic Diffuse Environments. *Archives of Theoretical and Applied Computer Science*, 14 (4), 263–275

MARR, D. (1982): Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.

MARROQUIM, R., KRAUS, M. AND CAVALCANTI, P. R. (2007): Efficient Point-Based Rendering Using Image Reconstruction. In *Symposium on Point-Based Graphics 2007*, 189–203

MAX, N. (1995): Optimal Sampling for Hemicubes. *IEEE Trans. on Visualization and Computer Graphics*, 1 (1), 60–76

MCGUIRE, M. AND LUEBKE, D. (2009): Hardware-Accelerated Global Illumination by Image Space Photon Mapping. In *Proc. ACM SIGGRAPH / Eurographics Conference on High Performance Graphics*

MÉNDEZ, A., SBERT, M. AND CATÁ, J. (2003): Real-time Obscurances with Color Bleeding. In *Proc. SCCG*, 171–176

MERTENS, T., KAUTZ, J., BEKAERT, P., REETH, F. V. AND SEIDEL, H.-P. (2005): Efficient Rendering of Local Subsurface Scattering. *Computer Graphics Forum*, 24 (1), 41–50

MEYER, Q., EISENACHER, C., STAMMINGER, M. AND DACHSBACHER, C. (2009): Data-Parallel Hierarchical Link Creation for Radiosity. In *Proc. EGPGV*, 65–70

MICHAEL, R. AND BARRAQUER, R. I. (2006): Refractive Index of Lens Fiber Membranes in Different Parts of the Crystalline Lens: Impact on Lens Transparency. In *New Developments in Eye Research* . – chapter 7, 167–181

MICHAEL, R., MARLE, J. VAN, VRENSEN, G. F. AND BERG, T. J. VAN DEN (2002): Refractive Index of Lens Fiber Membranes in Different Parts of the Crystalline Lens. In *Proc. of SPIE* Volume 4611,, 159–164

MILLER, G. (1994): Efficient Algorithms for Local and Global Accessibility Shading. In *Proc. ACM SIGGRAPH*, 319–326

MILLER, G. AND HOFFMAN, C. (1984): Illumination and Reflection maps: Simulated Objects in Simulated and Real Environments. SIGGRAPH Seminar Notes

MITTRING, M. (2007): Finding Next-Gen: CryEngine 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, 97–121

MOON, P. AND SPENCER, D. E. (1944): On the Stiles-Crawford Effect. *Journal of the Optical Society of America*, 34 (6), 319–329

MORELAND, K. AND ANGEL, E. (2003): The FFT on a GPU. In *Proc. Graphics Hardware*, 112–119

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M. AND GROSS, M. (2005): Meshless Deformations Based on Shape Matching. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 471–478

MURRAY, I. J., PLAINIS, S. AND CARDEN, D. (2002): The Ocular Stress Monitor: A New Device for Measuring Discomfort Glare. *Lighting Research & Technology*, 34 (3), 231–242

MYSZKOWSKI, K., TAWARA, T., AKAMINE, H. AND SEIDEL, H.-P. (2001): Perception-Guided Global Illumination Solution for Animation Rendering. In *Proc. ACM SIGGRAPH*, 221–230

NAKAMAE, E., KANEDA, K., OKAMOTO, T. AND NISHITA, T. (1990): A Lighting Model Aiming at Drive Simulators. *Computer Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 395–404

NAVARRO, R. (1985): Incorporation of Intraocular Scattering in Schematic Eye Models. *Journal of the Optical Society of America A*, 2 (11), 1891–1894

NAYAR, S. K., IKEUCHI, K. AND KANADE, T. (1991): Shape From Interreflections. *Int. J. Comput. Vision*, 6 (3), 173–195, ISSN 0920–5691

NG, R., RAMAMOORTHI, R. AND HANRAHAN, P. (2003): All-frequency Shadows Using Non-linear Wavelet Lighting Approximation. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 22 (3), 376–381

NG, R., RAMAMOORTHI, R. AND HANRAHAN, P. (2004): Triple Product Wavelet Integrals for All-frequency Relighting. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 23 (3), 477–487

NICHOLS, G. AND WYMAN, C. (2009): Multiresolution Splatting for Indirect Illumination. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 83–90

NISHITA, T. AND NAKAMAE, E. (1985): Continuous Tone Representation of Three-dimensional Objects Taking Account of Shadows and Interreflection. In *Proc. ACM SIGGRAPH*, 23–30

NOWROUZEZAHRAI, D., KALOGERAKIS, E. AND FIUME, E. (2009): Shadowing Dynamic Scenes with Arbitrary BRDFs. *Comput. Graph. Forum (Proc. Eurographics)*, 28, 249–258

NOWROUZEZAHRAI, D. AND SNYDER, J. (2009): Fast Global Illumination on Dynamic Height Fields. *Comput. Graph. Forum (Proc. EGSR)*, 28 (4), 1131–1139

OBERT, J., KRIVÁNEK, J., PELLACINI, F., SÝKORA, D. AND PATTANAIK, S. N. (2008): iCheat: A Representation for Artistic Control of Indirect Cinematic Lighting. *Computer Graphics Forum (Proc. EGSR)*, 27 (4), 1217–1223

OKABE, M., MATSUSHITA, Y., SHEN, L. AND IGARASHI, T. (2007): Illumination Brush: Interactive Design of All-Frequency Lighting. In *Proc. Pacific Graphics 2007*, 171–180

OSTROVSKY, Y., CAVANAGH, P. AND SINHA, P. (2005): Perceiving Illumination Inconsistencies in Scenes. *Perception*, 34 (11), 1301–1314

PALMER, S. E. (1999): Vision Science – Photons to Phenomenology.

PAMPLONA, V. F., OLIVEIRA, M. M. AND BARANOSKI, G. V. G. (2009): Photorealistic Models for Pupil Light Reflex and Iridal Pattern Deformation. *ACM Trans. on Graphics*, 28 (4), 1–12

PATTANAIK, S. N., FERWERDA, J. A., FAIRCHILD, M. D. AND GREENBERG, D. P. (1998): A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display. In *Proc. ACM SIGGRAPH*, 287–298

PELLACINI, F., BATTAGLIA, F., MORLEY, K. AND FINKELSTEIN, A. (2007): Lighting with Paint. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (2), 9

PELLACINI, F., FERWERDA, J. A. AND GREENBERG, D. P. (2000): Toward a Psychophysically-based Light Reflection Model for Image Synthesis. In *Proc. ACM SIGGRAPH*, 55–64

PHARR, M. AND HUMPHREYS, G. (2004a): Infinite Area Light Source With Importance Sampling.

PHARR, M. AND HUMPHREYS, G. (2004b): Physically Based Rendering: From Theory to Implementation.

PHONG, B.-T. (1975): Illumination for Computer Generated Pictures. *Communications of the ACM*, 18 (6), 311–317

POULIN, P., RATIB, K. AND JACQUES, M. (1997): Sketching Shadows and Highlights to Position Lights. In *Proc. Computer Graphics International '97*, 56–63

PURCELL, T., DONNER, C., CAMMARANO, M., JENSEN, H. AND HANRAHAN, P. (2003): Photon Mapping on Programmable Graphics Hardware. In *Proc. Graphics Hardware*, 41–50

RAMAMOORTHI, R. AND HANRAHAN, P. (2001): A Signal-processing Framework for Inverse Rendering. In *Proc. ACM SIGGRAPH*, 117–128

RAMAMOORTHI, R., MAHAJAN, D. AND BELHUMEUR, P. (2007): A First-order Analysis of Lighting, Shading, and Shadows. *ACM Trans. on Graphics*, 26 (1), 2

RAMANARAYANAN, G., FERWERDA, J., WALTER, B. AND BALA, K. (2007): Visual Equivalence: Towards a New Standard for Image Fidelity. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), 76

RASKAR, R., AGRAWAL, A., WILSON, C. A. AND VEERARAGHAVAN, A. (2008): Glare Aware Photography: 4D Ray Sampling for Reducing Glare Effects of Camera Lenses. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 27 (3), 1–10

REINBOTHE, C., BOUBEKEUR, T. AND ALEXA, M. (2009): Hybrid Ambient Occlusion. In *Proc. Eurographics (Area Papers)*

REINHARD, E., WARD, G., PATTANAIK, S. AND DEBEVEC, P. (2005): High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics).

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q. AND GUO, B. (2006): Real-Time Soft Shadows in Dynamic Scenes using Spherical Harmonic Exponentiation. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 25 (3), 977–986

ROBISON, A. AND SHIRLEY, P. (2009): Image Space Gathering. In *Proc. ACM SIGGRAPH / Eurographics Conference on High Performance Graphics*, 91–98

ROKITA, P. (1993): A Model for Rendering High Intensity Lights. *Computers & Graphics*, 17 (4), 431–437

RUSHMEIER, H., PATTERSON, C. AND VEERASAMY, A. (1993): Geometric Simplification for Indirect Illumination Calculations. In *Proc. Graphics Interface*, 227–236

RUSINKIEWICZ, S., BURNS, M. AND DECARLO, D. (2006): Exaggerated Shading for Depicting Shape and Detail. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 25 (3), 1199–1205

RUSINKIEWICZ, S. AND LEVOY, M. (2000): QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proc. ACM SIGGRAPH*, 343–352

SAGAN, H. (1994): Space-Filling Curves.

SAITO, T. AND TAKAHASHI, T. (1990): Comprehensible Rendering of 3-D Shapes. In BASKETT, F., EDITOR: *Proc. ACM SIGGRAPH*, 197–206

SATTLER, M., SARLETTE, R., ZACHMANN, G. AND KLEIN, R. (2004): Hardware-accelerated Ambient Occlusion Computation. In *Proc. VMV*, 331–338

SATTLER, M., SARLETTE, R., MÜCKEN, T. AND KLEIN, R. (2005): Exploitation of Human Shadow Perception for Fast Shadow Rendering. In *Proc. ACM Symposium on Applied Perception in Graphics and Visualization*, 131–134

SCHAEFER, S., MCPHAIL, T. AND WARREN, J. (2006): Image Deformation Using Moving Least Squares. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 25 (3), 533–540

SCHMITZ, A., TAVENRATH, M. AND KOBBELT, L. (2008): Interactive Global Illumination for Deformable Geometry in CUDA. *Computer Graphics Forum*, 27 (7), 1979–1986

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J. AND GREENBERG, D. (1993): Painting With Light. In *Proc. ACM SIGGRAPH*, 143–146

SCHRÖDER, P. AND HANRAHAN, P. (1993): On the Form Factor Between Two Polygons. In *Proc. ACM SIGGRAPH*, 163–164

SEGOVIA, B., IEHL, J.-C., MITANCHEY, R. AND PÉROCHE, B. (2006): Non-interleaved Deferred Shading of Interleaved Sample Patterns. In *Proc. Graphics Hardware*

SHACKED, R. AND LISCHINSKI, D. (2001): Automatic Lighting Design Using a Perceptual Quality Metric. *Comput. Graph. Forum*, 20 (3), 215 – 226

SHANMUGAM, P. AND ARIKAN, O. (2007): Hardware Accelerated Ambient Occlusion Techniques on GPUs. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 73–80

SHEEDY, J. E., TRUONG, S. D. AND HAYES, J. R. (2003): What are the Visual Benefits of Eyelid Squinting? *Optometry & Vision Science*, 80 (11), 740–744

SHEVTSOV, M., SOUPIKOV, A. AND KAPUSTIN, A. (2007): Highly Parallel Fast KD-tree Construction for Interactive Ray Tracing of Dynamic Scenes. *Comput. Graph. Forum (Proc. Eurographics)*, 26 (3), 395–404

SILLION, F. (1995): A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. *IEEE Trans. on Visualization and Computer Graphics*, 1 (3), 240–254

SILLION, F. AND DRETTAKIS, G. (1995): Feature-based Control of Visibility Error: A Multi-Resolution Clustering Algorithm for Global Illumination. In *Proc. ACM SIGGRAPH*, 145–152

SIMPSON, G. (1953): Ocular Haloes and Coronas. *Br. J. Ophthalmol.* 37 (8), 450–486

SINTORN, E., EISEMANN, E. AND ASSARSSON, U. (2008): Sample-based Visibility for Soft Shadows Using Alias-free Shadow Maps. *Comput. Graph. Forum (Proc. EGSR)*, 27 (4), 1285–1292

SLOAN, P.-P., KAUTZ, J. AND SNYDER, J. (2002): Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 21 (3), 527–536

SLOAN, P.-P., GOVINDARAJU, N., NOWROUZEZAHRAI, D. AND SNYDER, J. (2007): Image-Based Proxy Accumulation for Real-Time Soft Global Illumination. In *Proc. Pacific Graphics*, 97–105

SLOAN, P.-P., HALL, J., HART, J. AND SNYDER, J. (2003a): Clustered Principal Components for Precomputed Radiance Transfer. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 22 (3), 382–391

SLOAN, P.-P., LIU, X., SHUM, H.-Y. AND SNYDER, J. (2003b): Bi-scale Radiance Transfer. *ACM Trans. on Graphics*, 22 (3), 370–375

SLOAN, P.-P., LUNA, B. AND SNYDER, J. (2005): Local, Deformable Precomputed Radiance Transfer. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 1216–1224

SOATTO, S., DORETTO, G. AND WU, Y. N. (2001): Dynamic Textures. In *International Journal of Computer Vision*, 439–446

SORKINE, O. AND ALEXA, M. (2007): As-Rigid-As-Possible Surface Modeling. In *Proc. SGP '07*, 109–116

SPENCER, G., SHIRLEY, P., ZIMMERMAN, K. AND GREENBERG, D. P. (1995): Physically-Based Glare Effects for Digital Images. *Computer Graphics Forum*, 29 (Annual Conference Series), 325–334

STEWART, A. J. AND LANGER, M. S. (1997): Towards Accurate Recovery of Shape from Shading under Diffuse Lighting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19, 1020–1025

STOKES, W. A., FERWERDA, J. A., WALTER, B. AND GREENBERG, D. P. (2004): Perceptual Illumination Components: A New Approach to Efficient, High Quality Global Illumination Rendering. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 23 (3), 742–749

STOLLNITZ, E. J., ROSE, T. D. AND SALESIN, D. H. (1995): Wavelets for Computer Graphics: Theory and Applications.

STRASSER, W. (1974): Schnelle Kurven- und Flächendarstellung auf graphischen Sichtgeräten. Ph. D thesis, TU Berlin, Germany

SUN, B. AND RAMAMOORTHI, R. (2009): Affine Double- and Triple-product Wavelet Integrals for Rendering. *ACM Trans. on Graphics*, 28 (2), 1–17

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J. AND HOPPE, H. (2005): Fast Exact and Approximate Geodesics on Meshes. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 553–560

TABELLION, E. AND LAMORLETTE, A. (2004): An Approximate Global Illumination System for Computer Generated Films. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 23 (3), 469–476

THEISEL, H. (2001): Are Isophotes and Reflection Lines the Same? *Comput. Aided Geom. Des.* 18 (7), 711–722

TODO, H., ANJYO, K.-I., BAXTER, W. AND IGARASHI, T. (2007): Locally Controllable Stylized Shading. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), 17

TOMASI, C. AND MANDUCHI, R. (1998): Bilateral Filtering for Gray and Color Images. In *Proc. ICCV*, 839

TOSUN, E., GINGOLD, Y. I., REISMAN, J. AND ZORIN, D. (2007): Shape Optimization Using Reflection Lines. In *Proc. SGP '07*, 193–202

VANGORP, P., LAURIJSSEN, J. AND DUTRÉ, P. (2007): The Influence of Shape on the Perception of Material Reflectance. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 26 (3), 77

VEACH, E. AND GUIBAS, L. J. (1995): Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proc. ACM SIGGRAPH*, 419–428

VOLEVICH, V., MYSZKOWSKI, K., KHODULEV, A. AND KOPYLOV, E. A. (2000): Using the Visual Differences Predictor to Improve Performance of Progressive Global Illumination Computation. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 19 (2), 122–161

VOS, J. J. AND BOOGAARD, J. (1963): Contribution of the Cornea to Entoptic Scatter. *Journal of the Optical Society of America*, 53 (7), 869–873

VOS, J. J. AND BOUMAN, M. A. (1964): Contribution of the Retina to Entoptic Scatter. *Journal of the Optical Society of America*, 54 (1), 95–100

WALD, I. (2004): Realtime Ray Tracing and Interactive Global Illumination. Ph. D thesis, Saarland University, Germany

WALD, I., KOLLIG, T., BENTHIN, C., KELLER, A. AND SLUSALLEK, P. (2002): Interactive Global Illumination. In *Proc. Eurographics Workshop on Rendering*, 9–20

WALLACE, J. R., ELMQUIST, K. A. AND HAINES, E. A. (1989): A Ray Tracing Algorithm for Progressive Radiosity. *Computer Graphics (Proc. ACM SIGGRAPH)*, 23 (3), 315–324

WALTER, B., ARBREE, A., BALA, K. AND GREENBERG, D. P. (2006): Multidimensional Lightcuts. *ACM Trans. on Graphics (Proc. ACM 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 Trans. on Graphics (Proc. ACM SIGGRAPH)*, 24 (3), 1098–1107

WAND, M., FISCHER, M., PETER, I., HEIDE, F. M. AUF DER AND STRASSER, W. (2001): The Randomized z-Buffer Algorithm: Interactive Rendering of Highly Complex Scenes. In *Proc. ACM SIGGRAPH*, 361–370

WANG, J., REN, P., GONG, M., SNYDER, J. AND GUO, B. (2009a): All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 28 (5), 1–10

WANG, R., ZHU, J.-J. AND HUMPHREYS, G. (2007): Precomputed Radiance Transfer for Real-time Indirect Lighting Using a Spectral Mesh Basis. In *Proc. Eurographics Symposium on Rendering*, 13–21

WANG, R., NG, R., LUEBKE, D. AND HUMPHREYS, G. (2006): Efficient Wavelet Rotation for Environment Map Rendering. In *Proc. Eurographics Symposium on Rendering*, 173–183

WANG, R., TRAN, J. AND LUEBKE, D. P. (2004): All-Frequency Relighting of Non-Diffuse Objects using Separable BRDF Approximation. In *Proc. Eurographics Symposium on Rendering*, 345–354

WANG, R., WANG, R., ZHOUN, K., PAN, M. AND BAO, H. (2009b): An Efficient GPU-based Approach for Interactive Global Illumination. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH)*, 28 (3), 91:1–91:8

WANGER, L. (1992): The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 39–42

WARD, G., RUBINSTEIN, F. AND CLEAR, R. (1988): A Ray Tracing Solution for Diffuse Interreflection., 85–92

WARD, G. AND HECKBERT, P. (1992): Irradiance Gradients. In *Proc. Eurographics Symposium on Rendering*, 85–98

WHITTED, T. (1980): An improved Illumination Model for Shaded Display. *Commun. ACM*, 23 (6), 343–349

WILLIAMS, L. (1978): Casting Curved Shadows on Curved Surfaces. *Computer Graphics (Proc. ACM SIGGRAPH)*, 12 (3), 270–274

WOLBERG, G. (1998): Image Morphing: A Survey. *The Visual Computer*, 14 (8), 360–372

WYMAN, C. (2008): Hierarchical Caustic Maps. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 163–171

WYSZECKI, G. AND STILES, W. S. (1982): Color Science: Concepts and Methods, Quantitative Data and Formulae. 2nd edition.

YEE, H., PATTANAIK, S. AND GREENBERG, D. P. (2001): Spatiotemporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments. *ACM Trans. on Graphics*, 20 (1), 39–65

YELLOTT, J. I. J. (1983): Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina. *Science*, 221, 382–385

YOSHIDA, A., IHRKE, M., MANTIUK, R. AND SEIDEL, H.-P. (2008): Brightness of Glare Illusion. In *Proc. ACM Symposium on Applied Perception in Graphics and Visualization*, 83–90

YUAN, R., YAGER, D., GUETHLEIN, M., OLIVER, G., KAPOOR, N. AND ZHONG, R. (1993): Controlling Unwanted Sources of Threshold Change in Disability Glare Studies: A Prototype Apparatus and Procedure. *Optometry & Vision Science*, 70 (11), 976–981

YUE, Y., IWASAKI, K., CHEN, B.-Y., DOBASHI, Y. AND NISHITA, T. (2009): Interactive Rendering of Interior Scenes with Dynamic Environment Illumination. *Comput. Graph. Forum (Proc. Pacific Graphics)*, 28, 1935–1944

ZATZ, H. R. (1993): Galerkin radiosity: A higher order solution method for global illumination. In *Proc. ACM SIGGRAPH*, 213–220

ZHOU, K., HOU, Q., WANG, R. AND GUO, B. (2008): Real-time KD-tree Construction on Graphics Hardware. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 27 (5), 126

ZHUKOV, S., IONES, A. AND KRONIN, G. (1998): An Ambient Light Illumination Model. In *Proc. Eurographics Workshop on Rendering*, 45–56

ZIMMERMAN, R. L. (1980): In Vivo Measurements of the Viscoelasticity of the Human Vitreous Humor. *Biophysical Journal*, 29 (3), 539–544