Efficient Shadow Map Filtering

Thomas Annen

Max-Planck-Institut Informatik
Saarbrücken, Germany

Dissertation zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.–Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Eingereicht am 10. Oktober 2008 in Saarbrücken.

Betreuender Hochschullehrer — Supervisor

Prof. Dr. Hans-Peter Seidel	MPI Informatik Saarbrücken, Germany

Gutachter — Reviewers

Prof. Dr. Hans-Peter Seidel	MPI Informatik Saarbrücken, Germany
Prof. Dr. Jan Kautz	University College London London, UK
Prof. Dr. Frédo Durand	Massachusetts Institute of Technology Cambridge, USA

Dekan — Dean

Prof. Dr. Joachim Weickert	Universität des Saarlandes Saarbrücken, Germany

Datum des Kolloquiums — Date of Defense

12. Dezember 2008 in Saarbrücken

Prüfungsausschuss — Board of Examiners

Head of Colloquium	Prof. Dr. Christph Weidenbach	MPI Informatik Saarbrücken, Germany
Examiner	Prof. Dr. Hans-Peter Seidel	MPI Informatik Saarbrücken, Germany
Examiner	Prof. Dr. Jan Kautz	University College London London, UK
Protocol	Prof. Dr. Karol Myszkowski	MPI Informatik Saarbrücken, Germany

Thomas Annen
Max-Planck-Institut Informatik
Campus E1 4 (Room 226)
66123 Saarbrücken, Germany
tannen@mpi-inf.mpg.de

Abstract

Shadows provide the human visual system with important cues to sense spatial relationships in the environment we live in. As such they are an indispensable part of realistic computer-generated imagery. Unfortunately, visibility determination is computationally expensive. Image-based simplifications to the problem such as Shadow Maps perform well with increased scene complexity but produce artifacts both in the spatial and temporal domain because they lack efficient filtering support.

This dissertation presents novel real-time shadow algorithms to enable efficient filtering of Shadow Maps in order to increase the image quality and overall coherence characteristics. This is achieved by expressing the shadow test as a sum of products where the parameters of the shadow test are separated from each other. Ordinary Shadow Maps are then subject to a transformation into new so called *basis-images* which can, as opposed to Shadow Maps, be linearly filtered. The convolved basis images are equivalent to a pre-filtered shadow test and used to reconstruct anti-aliased as well as physically plausible all-frequency shadows.

Kurzfassung

Schatten liefern dem menschlichen Auge wichtige Informationen, um die räumlichen Beziehungen in der Umgebung in der wir leben wahrzunehmen. Sie sind somit ein unverzichtbarer Bestandteil der realistischen Bildsynthese. Leider ist die Sichtbatkeitsberechnung ein rechenintensiver Prozess. Bildbasierte Methoden, wie zum Beispiel Shadow Maps, verhalten sich positiv gegenüber einer wachsenden Szenenkomplexität, produzieren aber Artefakte sowohl in der räumlichen, als auch in der temporalen Domäne, da sie nicht wie herkömmliche Bilder gefiltert werden können.

Diese Dissertation präsentiert neue Echtzeit-Schattenverfahren die das effiziente Filtern von Shadow Maps ermöglichen, um die Bildqualität und das Kohärenzverhalten zu verbessern. Hierzu formulieren wir den Schattentest als eine Summe von Produkten, bei der die beiden Parameter der Schattenfunktion separiert werden. Shadow Maps werden dann in sogenannte *Basis-Bilder* transformiert, die im Gegensatz zu Shadow Maps linear gefiltert werden können. Die gefilterten Basis-Bilder sind äquivalent zu einem vorgefilterten Schattentest und werden verwendet, um geglättete Schattenkanten und realistische weiche Schatten zu berechnen.

Summary

Shadows provide the human visual system with important cues to sense spatial relationships in the environment we live in. As computer generated imagery has become an integral part of our lives, e.g. media, computer animated films or games, fast and high-quality shadow algorithms are key to realistic and efficient digital image synthesis. In particular, games and preview systems in movie production environments require real-time or interactive feedback to offer a player an enjoyable gaming experience or to equip artists with productive working tool sets.

Unfortunately, quality and feedback-time are two opposing objectives where either one often has to be compromised to achieve the other. As visibility computation takes a significant amount of the overall rendering time and as we witness a steady growth in geometric fidelity including dynamic and deformable objects, modern real-time shadow algorithm have to fulfill several requirements. They must be efficient, render high-quality shadows, and they must be flexible with regard to the input data. Even though the theoretical foundations for computing accurate shadows are well established, combining all aforementioned requirements renders visibility computation a challenging problem.

This dissertation is therefore dedicated to novel solutions for real-time shadow rendering and builds on Williams' Shadow Maps [Williams, 1978]. To this end, we propose a new mathematical framework to transform traditional Shadow Maps into a new representation which naturally affords Shadow Map filtering, an important property not available otherwise. Our new algorithms maintain the efficiency and flexibility of Shadow Maps but overcome their crucial limitations.

Part I reviews the filtering problem inherent in Shadow Mapping which stems from the *non-linearity* of the shadow test function. We explain why filtering the depth values is not equivalent to filtering the result of the shadow test and present a new solution to *linearize* the shadow function. We achieve this by expressing the shadow test as a sum of products where we separate the two parameters d and z of the shadow test function. Where d represents the distance from the shading point to the light source, and z encodes the closed blocker for that shading point. Thereby we can evaluate filtered shadows in constant-time through pre-filtering which leads to better shadow quality, performance, and temporal coherence. Based on this core idea we develop new techniques for anti-aliasing shadow discontinuities in Part II and introduce an extension to our linearization process for rendering efficient soft shadows in Part III.

Part II proposes two solutions to expand the shadow test function, i.e. into a Fourier and an Exponential series. The Fourier series approach transforms a depth map into a set of special *basis images*. We can then apply arbitrary linear filter kernels to each image. During rendering we evaluate the series expansion and effectively reconstruct a filtered shadow test.

The second solution we present relies on the same linearization pipeline as the previous method, but only requires a single basis image. We trade quality for speed and memory consumption by assuming that the shadow test domain is limited in order to simplify the problem. By imposing this assumption we can show that the shadow test can be approximated by a simple exponential function which yields a low memory footprint and an increased performance while the quality is still comparable to the Fourier series.

Part III focuses on realistic physically plausible shadows and derives an extended theory to harness our pre-filtering facilities for rapid and high-quality all-frequency shadow computation. The main idea is to replace an exhaustive and explicit blocker search which determines the softness of the shadow by a constant-time reconstruction. Not only allows this new method to render physically plausible shadows in real-time but it also supports dynamic objects and arbitrary distant environment illumination.

In summary, this dissertation contributes new ideas and solutions to an important and long standing problem in the field of Computer Graphics. Our algorithms cover a broad range of applications from real-time anti-aliasing of shadow discontinuities to rendering all-frequency shadows, in fully dynamic environments with multiple light sources simultaneously.

Zusammenfassung

Schatten liefern dem menschlichen Auge wichtige Informationen, um die räumlichen Beziehungen in der Umgebung in der wir leben wahrzunehmen. Da computergenerierte Bilder ein integraler Bestandteil unseres Lebens geworden sind, z.B. Medien, computeranimierte Filme oder Spiele, werden schnelle Schattenalgorithmen zum Schlüssel zur realistischen und effizienten digitalen Bildsynthese. Besonders Spiele oder Vorschausysteme in Filmstudios erfordern Echtzeit- oder interaktives Feedback, um Spielern eine unterhaltsame Spielerfahrung bieten zu können bzw. Künstler mit produktiven Arbeitsmitteln auszustatten.

Leider konkurrieren Qualität und Feedback-Verhalten der Software oft um Ressourcen, und nicht selten müssen bei einem der beiden Ziele Kompromisse eingegangen werden, um das andere zu erreichen. Da die Sichtbarkeitsberechnung alleine einen beachtlichen Teil der Renderingzeit in Anspruch nimmt, und geometrische Komplexität ständig wächst, z.B. dynamische oder deformierbare Objekte, sollten moderne Schattenverfahren mehrere Bedingungen erfüllen. Sie müssen effizient sein, Schatten in hoher Qualität erzeugen, und robust bezüglich der Eingabeprimitiven sein. Obwohl die theoretischen Grundlagen für die genaue Berechnug von Schatten ethabliert sind, bleibt die Schattengenierung eine grosse Herausforderung, wenn alle vorher genannten Bedingungen erfüllt werden sollen.

Diese Doktorarbeit widmet sich daher neuen Lösungen für Echtzeit Schattenverfahren und gründet auf Williams' Shadow Maps [Williams, 1978]. Hierzu präsentieren wir ein neues mathematischen Grundgerüst, das es ermöglicht traditionelle Shadow Maps in eine neue Repräsentation zu transformieren, die das effektive Filtern von Shadow Maps erlaubt. Dies ist eine wichtige Eigenschaft, die für herkömmliche Shadow Maps leider nicht gilt. Unsere neuen Algorithmen erhalten dabei die Effizienz und Flexibilität normaler Shadow Maps, löen aber einige ihrer kritischen Probleme.

Teil I gibt einen Einblick in das Problem des Shadow Map Filterns, welches auf der *Nicht-Linearität* des Schattentests beruht. Wir erklären im Detail, warum es das Filtern der Tiefenwerte nicht äquivalent zum Filtern der Ergebnisse des Schattentests ist, und stellen eine neue Methode vor, um die Schattenfunktion zu *linearisieren*. Wir erreichen dieses Ziel indem wir den Schattentest als eine Summe von Produkten ausdrücken mittels derer wir die beiden Parameter d und z der Testfunktion von einander trennen. d stellt dabei die Distanz vom Shadingpunkt zur Lichtquelle dar, und z kodiert die kleinste Blockerdistanz zur Lichtquelle für den Shadingpunkt. Dadurch gelingt es uns vorgefilterte Schatten in konstanter Zeit zu errechnen, was zu besserer Schattenqualität, Laufzeit und temporaler Kohärenz

führt. Darauf basierend entwickeln wir neue Methoden zum Filtern von Schattenkanten in Teil II und demonstrieren eine Erweiterung zum Erzeugen realistischer Schatten im Teil III.

Teil II präsentiert zwei konkrete Lösungen zur Linearisierung des Schattentests: eine Fourier und eine Exponential Reihe. Die Fourier Reihe transformiert eine normale Shadow Map in eine Menge sogenannter *Basis-Bilder*. Danach können wir beliebige lineare Glättungsfilter auf diese Basis-Bilder anwenden. Während des Renders werden diese Bilder dann genutzt, um die Reichenentwicklung auszuwerten und somit einen gefilterten Schatten zu rekonstruieren.

Die zweite Lösung, die wir präsentieren, basiert auf der gleichen Linearizierungspipeline wie zuvor, erfordert allerdings lediglich ein einziges Basis-Bild. Hierbei erlauben wir das die Qualität etwas vermindert wird, um im Gegenzug Speicher zu sparen und die Performance zu steigern. Hierfür machen wir ein Annahme bezüeglich des Definitionsbreichs des Schattentests, welche es uns erlaubt die Schattenfunktion mit einer einfachen Eponentialfunktion zu approximieren. Dies führt zu niedrigem Scheicherbedarf und einem verbesserten Laufzeitverhalten, wohingegen die Schattenqualität vergleichbar ist zu der, der Fourier Reihe.

Teil III dieser Dissertation konzentriert sich auf physikalisch plausible Schatten und leitet eine erweiterte Theorie her, die unsere speziellen Filtereigenschaften auch zur schnellen und hoch-qualitativen Berechnung von weichen Schatten verwendet. Die grundlegende Idee ist es die bisher aufwendige und explizite Blocker-Suche zum Bestimmen der Weichheit des Schattens, mit einem konstanten Lookup zu ersetzen. Das hat den Vorteil, dass es das Erzeugen von realistischen Schatten in Echtzeit erlaubt, voll dynamische Objekte, und sogar beliebige (weit entfernte) Emgebungsbeleuchtung unterstützt.

Kurz zusammengefasst, diese Dissertation trägt neue Lösungen zu einem wichtigen und lange existierenden Problem im Gebiet der Computer Graphik bei. Die hier präsentierten Algorithmen decken ein breites Spektrum an Anwendungsgebieten ab, das sich vom Filtern von Schattenkanten bis hin zum Erzeugen realistischer Schatten in Echtzeit in voll dynamischen Umgebungen und mehreren Lichtquellen erstreckt.

Acknowledgements

This dissertation would not have been possible without the help and support of many people. First, I would like to thank my Ph.D. adviser Prof. Dr. Hans-Peter Seidel who has guided, motivated, and supported me throughout my entire time at MPI. I am grateful for an excellent work environment and that I had the opportunity to visit several research laboratories over the past five years.

I owe special thanks to Prof. Dr. Jan Kautz for being a reviewer of my thesis and Dr. Tom Mertens. Both have guided me and shared their experiences with me during my Ph.D. studies. Over the past few years we have not only closely cooperated but we have also become good friends.

I am also grateful to Prof. Dr. Frédo Durand from CSAIL at the Massachusetts Institute of Technology for being an external reviewer of this dissertation and for his inspiring supervision during my research visit in his group in winter 2003.

Special thanks to my former supervisor Stefan Brabec who introduced me into the field of Computer Graphics and shadow computation in particular.

I owe special thanks to Zhao Dong for his effort on both the theoretical background and the implementation of the algorithm presented in Chapter 8.

Furthermore, I would like to thank all my colleagues from the Computer Graphics group at MPI for their help, support, and for making MPI a great place to work. I can not name all of them but special thanks go to (alphabetical order and including former members): Tunc Ozan Aydin, Sabine Budde, Christian Fuchs, Martin Fuchs, Michael Goesele, Thorsten Grosch, Hendrik P.A. Lensch, Conny Liegl, Karol Myszkowski, Christian Rössl, Carsten Stoll, Kristina Scherbaum, Holger Theisel, Rhaleb Zayer, und Gernot Ziegler.

Finally, I would like to thank my family Oranna, Norbert, Daniela, Thomas, Jonas, Jana, and Sandra for their support throughout the years of this dissertation. Very special thanks also to my best friends Kaleigh Smith, Bernd (Fluff) Kiefer, Oliver Müller, Stefan Lauer, and Michi Becker, for true friendship, great times, and memorable moments.

Thomas Annen

Contents

1 Introduction **3**
- 1.1 Problem Statement . . . 5
- 1.2 Main Contributions . . . 7
- 1.3 Chapter Overview . . . 9

2 Background **11**
- 2.1 Notation . . . 11
- 2.2 Radiometry and Photometry . . . 12
  - 2.2.1 Radiometric Quantities . . . 12
  - 2.2.2 Photometric Quantities . . . 15
- 2.3 Common Illuminats . . . 15
  - 2.3.1 Light Source Models . . . 16
  - 2.3.2 Discussion . . . 19
  - 2.3.3 Near-Field and Far-Field Theory . . . 20
- 2.4 Concepts of Surface Reflections . . . 21
  - 2.4.1 Bidirectional Reflectance Distribution Function . . . 21
  - 2.4.2 Material Properties . . . 23
- 2.5 The Rendering Equation . . . 25
- 2.6 The Framebuffer: Final Image Assembly . . . 26
- 2.7 Hardware Accelerated Rendering . . . 27
- 2.8 Shadows and Human Perception . . . 30
- 2.9 Assumptions . . . 31
  - 2.9.1 Visibility Computation . . . 31

3 Shadow and Visibility Techniques **33**
- 3.1 Shadow Classification . . . 33
- 3.2 Image-Space Methods . . . 37
  - 3.2.1 Z-Buffer Algorithm . . . 37
  - 3.2.2 Shadow Maps . . . 38
- 3.3 Object-Space Methods . . . 44
- 3.4 Painter's Algorithm . . . 44

3.4.1 Shadow Volumes 45
3.4.2 Shadow Rays 46
3.5 Hybrid Methods 47
3.6 Pre-computation Methods 48

4 Related Work on Shadow Map Filtering 49
4.1 Anti-aliasing 49
4.2 Soft Shadows 50

I Linearization 53

5 Shadow Test Linearization 55
5.1 Shadow Test Function 55
5.2 Convolution 56

II Anti-aliasing of Shadows 61

6 Convolution Shadow Maps 63
6.1 Fourier Series Expansion 63
6.1.1 Discussion of Fourier Expansion 65
6.2 Anti-aliasing Using CSMs 67
6.2.1 GPU Implementation 67
6.3 Results 69
6.4 Discussion 73

7 Exponential Shadow Maps 77
7.1 Exponential Approximation 77
7.1.1 Choice of Exponent 79
7.2 Violation of Assumption 79
7.2.1 Frequency of Violation 80
7.3 Classification and Fall Back Solution 81
7.3.1 Z-Max Classification 82
7.3.2 Threshold Classification 83
7.4 Implementation 83
7.5 Results 83
7.5.1 Discussion 86

III Pre-filtered Soft Shadows 91

8 Convolution Soft Shadow Maps 93
8.1 Plausible Soft Shadows Using Convolution 93
8.2 Convolution for Soft Shadows 94
8.2.1 Estimating Average Blocker Depth 96
8.2.2 Initializing Average Depth Computation 98
8.2.3 CSM Order Reduction 98
8.3 Illumination with Soft Shadows 99
8.3.1 Rendering Prefiltered Soft Shadows 99
8.4 Applications and Results 101
8.5 Discussion 103

9 Summary and Conclusions 109
9.1 Summary 109
9.2 Conclusions 110

"Shadow is the obstruction of light. Shadows appear to me to be of supreme importance in perspective, because, without them opaque and solid bodies will be ill defined; that which is contained within their outlines and their boundaries themselves will be ill-understood unless they are shown against a background of a different tone from themselves."

Leonardo da Vinci (1452 to 1519)

Chapter 1

Introduction

The desire to comprehend and to describe the interaction between light and matter has always been a fascinating and challenging problem throughout history. It has been subject to intensive research in disciplines like scientific study, art and philosophy. Shadows, a natural phenomenon resulting from the interaction, have been of special interest ever since the original questions were asked. In his book, a "Short History of the Shadow", Victor I. Stoichita [1997] explains that shadows have often been an integral element of theories, knowledge, as well as our perception of reality. He reaches back to Plato (428 BC - 348 BC[1]) and Pliny the Elder (23 AD – 79 AD) to point out how philosophers used shadows in metaphors to communicate their view on truth and knowledge. A famous example is Plato's "Allegory of the Cave" from the 7th book of "The Republic" by Plato [1968].

In more recent history during the Italian renaissance (1420 – 1600), Leonardo da Vinci, the embodiment of a versatile genius, made significant contributions to a large number of different fields. Among other activities he experimented with the interplay between light and objects. Leonardo sketched his results and documented his conclusions in a number of famous notebooks [see da Vinci, 1970]. Although these notes contain some misconceptions on how shadows are formed, it is astonishing to see the rich detail in which da Vinci separated and classified different types of light sources as well as their corresponding shadows. Figure 1.1 shows an example of da Vinci's drawings where he describes the intensities of cast shadow. The citation at the beginning of this introduction is also an excerpt from da Vinci's notebooks. It emphasizes his belief that shadows add very important spatial information to our perspective perception. He states that if they are missing, paintings or portraits will appear amorphous and flat. This aspect was known among artist too.

[1] According to Jonathan Barnes, British historian of ancient philosophy.

They carefully incorporated changes in brightness to better simulate light and shade in their paintings or drawings. This popular artistic tool of the 16th century is referred to as "Chiaroscuro" [2]. It allows painters to express shapes of bodies and objects in a more plastic manner and thereby lends realism to pictorial representations.

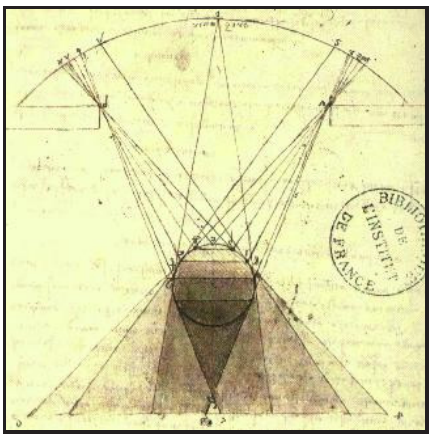


Figure 1.1: A sketch on shadows from da Vinci's notebooks.

Later in the last century along with the advent of modern computers came the wish to generate imagery digitally, which founded the field of Computer Graphics. One discipline of this field of research is (photo-) realistic image synthesis. There have been amazing advancements in the past decades which narrowed the gap between the real world and artificial environments. Computers are nowadays able to generate stunning results virtually indistinguishable from real images. Such technology has for instance been used to generate breathtaking visual effects in recent feature films.

A central part in realistic digital image synthesis is the evaluation of mutual visibility between primitives within virtual environments. Such entities are for instance, point samples, polygons, or arbitrarily shaped light sources. Knowing the visibility relation among any two of those items is a crucial factor and ultimately the key to physically accurate solutions. For example, a program needs to know if a surface point is fully, partially, or not visible at all to determine the amount of energy reaching this surface. Unfortunately, the importance of visibility computation to realistic image generation is accompanied by an enormous computational cost.

It was this computational burden that encouraged researchers to use approximations during lighting simulations to increase the overall rendering performance. The outcome was a separation into two major fields, local and global illumination techniques. Fully local methods consider only energy reaching a surface on a direct path from the light source. They usually neglect or imitate indirect effects such as shadows, inter-reflections, and caustics. Although these assumptions reduce the natural look of images and degrade quality, they grant local methods higher performance. Usually, images can be rendered in real-time or at interactive frame rates depending on the complexity of the shading model. Global methods do not rely on crude approximations, but instead involve physically based compu-

[2] Italian for clear/dark or light/dark.

tations. This yields superior and photo-realistic image quality but takes normally minutes or up to hours to produce a single frame.

This journey through the history of shadows in Computer Graphics was very brief and not exhaustive. Yet it displays the great relevance of shadows for conveying spatial information to make art work, technical illustrations, games, and computer generated imagery appear more realistic and natural.

We will now move on to a modern approach to resolve visibility and to compute shadows. The next section describes a critical problem in today's most popular and efficient real-time shadow procedure. We outline the difficulties and present a set of new solutions to overcome these limitations. By doing so we achieve more accurate, more efficient, and physically plausible shadows, which are valuable to many real-time applications.

1.1 Problem Statement

This dissertation focuses on efficient and high quality shadow computation in the context of real-time Computer Graphics and allows to include valuable spatial information into the local illumination domain. Today, the dominating shadow techniques in this field are *Shadow Volumes* by Crow [1977] and *Shadow Maps* by Williams [1978] (aka *depth maps*). The latter is far more popular partially due to its simplicity and robustness, but primarily due to its efficiency compared to shadow volumes. However, despite of several advantages, shadow mapping is plagued by aliasing artifacts originating from its image-based nature. Figure 1.2 (left) illustrates this problem, and shows the resulting poor shadow quality.

To lessen or resolve these artifacts has been the goal of many research articles. Plenty of these solutions tackle the problem by trying to increase the effective resolution of a shadow map in order to reduce discretization artifacts. However, so far little attention has been paid to filtering shadows even though it has a great impact regarding the image quality. The benefits of appropriate filtering are twofold. It conceals discretization artifacts and provides effective screen-space anti-aliasing. Second it drastically improves temporal coherence in animations. See Figure 1.2 (right) for a demonstration of the quality improvement when filtering is applied. One reason why filtering has drawn little attention so far may reside in the shadow test function itself. Shadow mapping is a non-linear operation with respect to the depth values stored in the shadow map. This fact reveals a fundamental problem in shadow mapping. It means that Shadow Maps **cannot** be filtered like ordinary texture maps and therefore it renders Shadow Map filtering a non-trivial operation. As a consequence filtering regular depth maps is very expensive. It requires

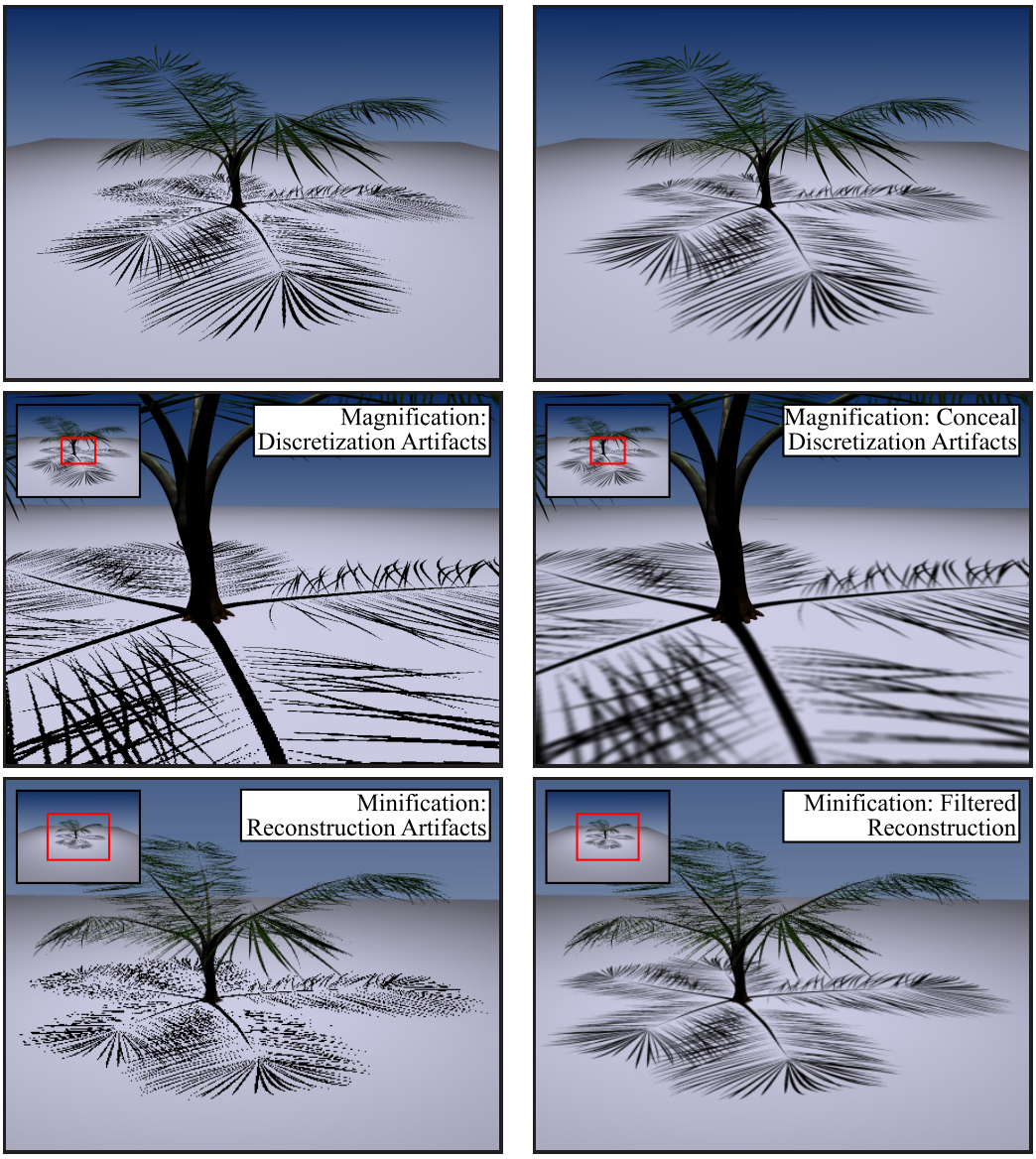


Figure 1.2: Williams' Shadow Maps are prone to aliasing. Two types of aliasing are shown on the left. Discretization and under-sampling artifacts. Our novel filtering methods reduce both problems and improve the overall shadow quality significantly show on the right.

explicit and extensive sampling of the depth values to achieve reasonable shadow quality.

Providing solutions to this challenging problem is essential because shadow mapping is widely used in games and in film production. This dissertation is

dedicated to this problem and provides new means for efficient Shadow Map filtering. But before we continue with the contributions we make, let us summarize some desirable and important properties which characterize a general, efficient, and high quality shadow algorithm. Such an algorithm should:

- be simple to implement and easy to integrate into existing software,
- general with respect to the rendering primitives (points, polygons, etc.),
- scale well with geometric complexity,
- allow pre-filtering to prevent expensive run-time sampling,
- yield high quality through efficient anti-aliasing,
- afford all-frequency shadow support.

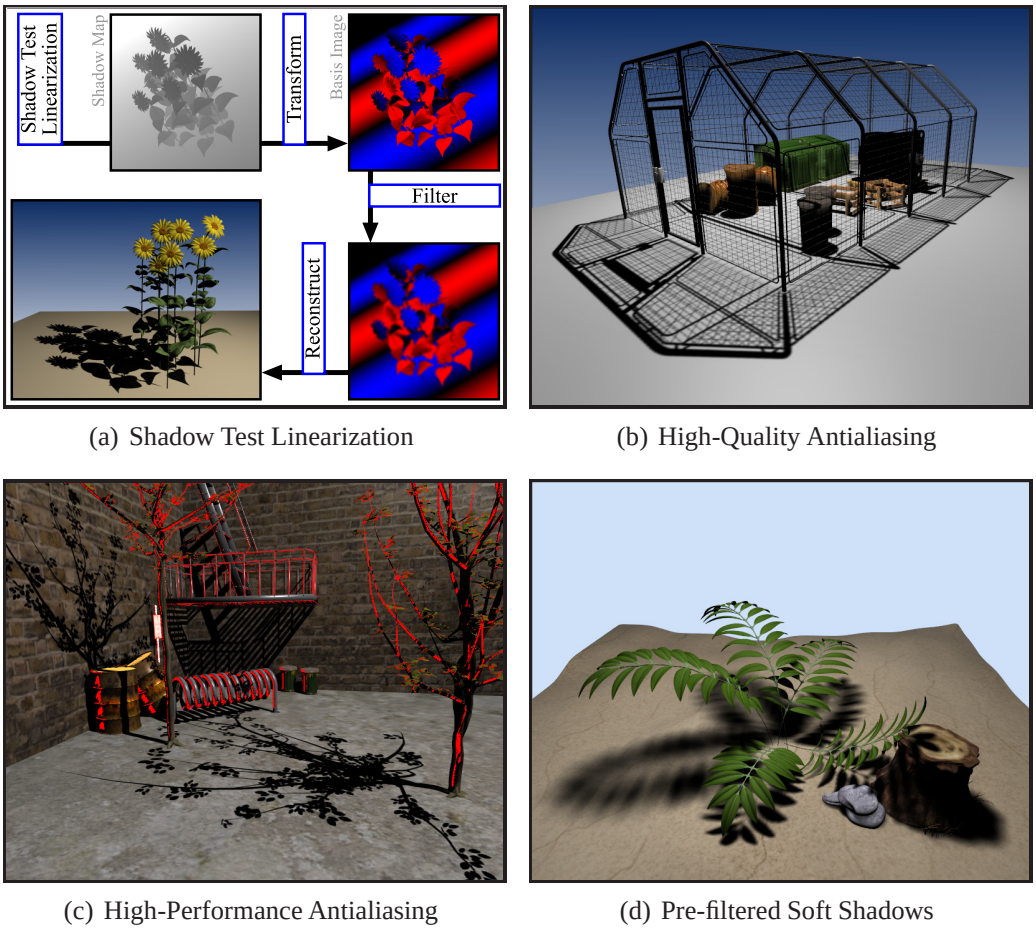
The first three qualities are provided by Shadow Maps and explain their popularity. They also motivate us to base our new ideas on Shadow Mapping to achieve the remaining three objectives. The next sections of this chapter discuss our contributions towards such an improved shadow technology and outline the remainder of this thesis.

1.2 Main Contributions

The contributions listed here have already been published in conference proceedings or journals. These publications are the central part of this dissertation in which we present:

- a new mathematical framework to decompose the shadow test into a sum of products to circumvent the filtering problem of Shadow Maps. By this we effectively enable filtering [Annen et al., 2007] the shadow test function before to the actual visibility evaluation is performed.
- *Convolution Shadow Maps* [Annen et al., 2007] a first solution to realize the decomposition of the shadow test into a Fourier Series expansion. This facilitates high-quality anti-aliasing of shadows boundaries in real-time even for large scenes and high-resolution Shadow Maps.
- *Exponential Shadow Maps* [Annen et al., 2008b] as a second approach which trades quality for high performance and memory savings by introducing an assumption on the parameter range of the shadow test in which case a sum of Exponentials suffices to compute a filtered shadow. This methods delivers very high frame-rates while preserving competitive quality compared to the Fourier Series.

- *Convolution Soft Shadow Maps* [Annen et al., 2008a] an extension of our mathematical framework to support more complex all-frequency shadows, e.g. penumbrae. We replace a costly explicit average blocker estimation used in many soft shadow algorithms by fast pre-filtering capabilities. An important step to render plausible all-frequency shadows[3] in real-time.



(a) Shadow Test Linearization (b) High-Quality Antialiasing

(c) High-Performance Antialiasing (d) Pre-filtered Soft Shadows

Figure 1.3: Contributions: (a) outline of our linearization process. (b) and (c) show two different solutions to anti-aliasing. This framework also delivers high quality and plausible all-frequency shadows in real-time depicted in (d).

[3]The frequency of shadows depends on the blocker, receiver, and light source configuration. An example that contains various frequencies is shown in Figure 1.3 (d).

1.3 Chapter Overview

After introducing the importance of shadows throughout history, we proceed in Chapter 2 with a background discussion to familiarize the reader with the basics of Computer Graphics necessary to understand the new shadow filtering techniques. This information will be instrumental when reviewing visibility and shadow techniques in Chapter 3. We then refer to the most related work on Shadow Map filtering in Chapter 4. Together these chapters give the context of this dissertation. In Part I we propose rethinking standard Shadow Mapping. We introduce an elegant process that involves a transformation from depth maps into a new kind of image which we call a **basis image**, from which shadows can be reconstructed. Because of the linearity of this procedure, we have effectively achieved the linearization of the non-linear standard shadow test. In contrast to regular Shadow Maps, this new image type can be filtered. Figure 1.3 (a) shows the entire process.

We develop two solutions for the linearization process in Part II. Our first solution is based on a Fourier Series expansion. It delivers high-quality anti-aliasing (see Figure 1.3 (b)) and its inherent properties permit further extensions. One limitation of this method is its memory consumption, which is why we propose a second algorithm based on an Exponential Series expansion. It is primarily designed to deliver very high frame rates and quality is of secondary importance. Even though it requires special treatment of a small amount of pixels (see pixels marked red in Figure 1.3 (c)) the overall quality is competitive compared to the Fourier Series solution. We will describe each method in detail and discuss their advantages and limitations.

Part III is dedicated to an extension of the theory presented in Part I and II. We can utilize our framework to formulate a highly efficient algorithm to render all-frequency shadows in real-time. It is based on the same theoretical foundations and achieves great speed-ups compared to previous procedures. Image (d) in Figure 1.3 is an example of high-quality all-frequency shadows. We then summarize this dissertation with conclusions on our approaches in Chapter 9.

Chapter 2
Background

Though efficient and high quality shadow computation is the primary objective of this dissertation, shadows only constitute one out of many complex natural phenomena emerging from light interacting with matter. In order to comprehend the versatile factors that cause or modify shadows, we first need to understand the physics of light. Specifically, how it propagates through space and the nature of its interplay with different material compositions. This chapter therefore strives to provide the reader with enough background information on rendering to comprehend our novel shadow ideas in the following chapters.

We first establish the notation we use throughout this dissertation, then layout properties of energy and conduct a light source classification. Once we have the light source models available we shift our focus to light-matter interaction and address the properties and geometry of surface reflectance. Eventually, this leads us to the fundamental equation in CG, the Rendering Equation. We conclude this chapter with a graphics hardware review, notes on how shadows impact our human perception, and a summary of assumptions our work is founded on.

2.1 Notation

This section describes the mathematical symbols we are going to use in this dissertation. We choose to denote spatial positions in $\mathbb{R}^3$ and $\mathbb{R}^2$ in bold font e.g. $\mathbf{x}$. We provide subscripts to further indicate specific coordinate frames. For example, when $\mathbf{x}$ has been transformed by the camera matrix it resides in *camera-space* $\mathbf{x_c}$. When $\mathbf{x}_c$ is projected onto a camera's image plane it is in *screen-space* (we use the term *texture- or Shadow Map space* when projecting onto the light source image plane) and we use an underline to indicate projected positions, e.g. $\underline{\mathbf{x}}_\mathbf{c}$.

For vectors in $\mathbb{R}^3$ we use the standard arrow sign e.g. $\vec{n}$ and their normalized (unit) counterparts are indicated using a hat symbol like $\hat{n}$. Subscripts pro-

Quantity	Description
$\mathbf{x}$	Point in $\mathbb{R}^3$
$\underline{\mathbf{x}}$	Point in $\mathbb{R}^2$
$\vec{v}, \hat{v}$	Vector and its normalized version in $\mathbb{R}^3$.
$_{\mathbf{e},\mathbf{o},\mathbf{i}}$	Subscripts refer to *emitted, outgoing,* and *incident*
$_{\mathbf{c},\mathbf{l}}$	Subscripts denote a variable in camera- or light-space
$\Omega, \hat{\Omega}$	Solid angle and unit solid angle.
w	Spatial convolution kernel in $\mathbb{R}^2$

Table 2.1: A list of quantities and their description used in this dissertation. Directional subscripts can be combined with camera- and light-space subscripts.

vide information on whether energy is incident (i) at their associated locations, or outgoing (o) from that point. We use (e) for emitted energy from light sources. Table 2.1 summarizes our notation.

2.2 Radiometry and Photometry

Radiometry is the scientific discipline concerned with the measurement of electromagnetic radiation including spectra like microwaves, infrared light, visible light, and ultraviolet light.

Photometry was established by Pierre Bouguer in 1760 and is the psychophysical measurement of electromagnetic radiation only taking into account energy perceptible by the human eye. It is typically limited to wavelengths between 380 and 740 nanometers (nm). The range visible to our human visual system is merely a narrow band in the electromagnetic spectrum.

2.2.1 Radiometric Quantities

The subsequent paragraphs layout all radiometric quantities and their individual SI units listed in Table 2.2. Subscripts in the table distinguish between radiometric and photometric symbols but we consider the full electromagnetic spectrum in our discussion.

Radiant Energy Q is electromagnetic radiation and can be seen as energy quantized into finite entities called photons. The energy carried by a single photon according to Planck's hypothesis is $Q = h$, where h is the Planck constant and is the frequency of radiation. The total radiant energy is the contribution of all photons over all wavelengths. Q is measured in *Joule* $[J = N \cdot m = kg \cdot m^2/s^2 = W \cdot s]$.

Radiometry			Photometry		
Quantity	Unit	Sym.	Quantity	Unit	Sym.
Radiant Energy	J	Q_r	Luminous Energy	$lm \cdot s$	Q_v
Radiant Flux	W	Φ	Luminous Flux	lm	F
Radiosity Irradiance	W/m^2	B_r E_r	Luminosity Illuminance	lm/m^2	B_v E_v
Radiant Intensity	W/sr	I_r	Luminous Intensity	lm/sr	I_v
Radiance	$W/(m^2 sr)$	L_r	Luminance	$lm/(m^2 sr)$	L_v

Table 2.2: Radiometric and photometric quantities and units. We denote radiometric terms with subscript r for *radiometry* and their photometric counterparts with subscript v for *visible* as they take the sensitivity of the human eye into account.

Radiant Flux or **Radiant Power** Φ is the energy transmitted over unit time and is defined by the following equation:

$$\Phi(\mathbf{x}, \Omega) := \frac{dQ(\mathbf{x}, \Omega)}{dt}. \tag{2.1}$$

When we integrate Φ over time we obtain the total radiant energy output Q. The physical unit of radiant flux is measured in *Watt* $[W = kg \cdot m^2/s^3 = J/s]$.

Radiance L is one of the most important quantities encountered in computer graphics. Radiance describes the differential flux per unit projected area, per unit solid angle, either incoming at a surface point $\mathbf{x}$ or leaving $\mathbf{x}$:

$$L(\mathbf{x}, \Omega) := \frac{d^2\Phi(\mathbf{x}, \Omega)}{dA_x \, d\Omega \, cos}, \tag{2.2}$$

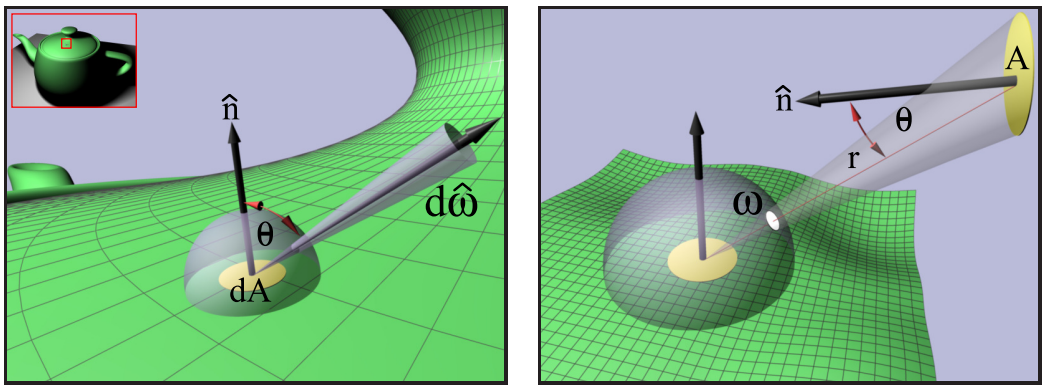
where is the angle between the surface normal at $\mathbf{x}$ and solid angle Ω. The solid angle is the 3D extension of planar 2D angles and it is proportional to the coverage of the projected surface A onto the sphere which translates to:

$$\Omega := \frac{A \, cos}{r^2}, \tag{2.3}$$

where r is the distance from the surface patch to the area being projected[1]. Note we use *cos* to describes the angle between a surface normal and a direction. Please see Figure 2.1 for an illustration of radiance and solid angle.

A very important characteristic of radiance is that it does not change when traveling through empty space. The reason why this is important in graphics is

[1] Integrating the solid angle over the unit sphere results in a total solid angle of 4 *sr*.



(a) Geometry of radiance (b) Geometry of the solid angle

Figure 2.1: A close-up of a differential surface area on a teapot surface is used to illustrate the geometry of radiance (a). The solid angle is given in (b) where an area is projected onto a sphere.

that the reflected radiance of a surface point moves along a ray towards the capturing device (virtual camera or eye) to fully determine the object appearance. The unit of radiance is $[W/(m^2 sr)]$.

Radiant Flux Density is the amount of energy per unit area that either arrives or leaves a differential surface area measured with respect to the upper hemisphere Ω^+ centered at that differential surface patch and aligned with the normal at that patch. Incident energy is called *irradiance* E, energy that is emitted is called *radiant exitance* M. Mathematically, they are written as:

$$E(\mathbf{x}) := \frac{d\Phi_i(\mathbf{x})}{dA_x} \quad , \quad M(\mathbf{x}) := \frac{d\Phi_o(\mathbf{x})}{dA_x}. \tag{2.4}$$

When multiplying radiance by the denominator in Equation 2.2 to get the differential flux and then inserting it into the above formula we can derive:

$$\begin{aligned} dE(\mathbf{x})\, dA_x &= L_i(\mathbf{x}, \Omega_i)\, dA_x\, d\Omega_i \cos\theta_i \\ E(\mathbf{x}) &= \int_{\Omega^+} L_i(\mathbf{x}, \Omega_i) \cos\theta_i \, d\Omega_i \\ M(\mathbf{x}) &= \int_{\Omega^+} L_o(\mathbf{x}, \Omega_o) \cos\theta_o \, d\Omega_o. \end{aligned} \tag{2.5}$$

Computer Graphics often refers to radiant exitance as *radiosity* $B = M$. The radiant flux density is given in units of $[W/m^2]$.

Radiant Intensity I is the ratio of flux per unit solid angle and can be used to describe the intensity of light sources. It is defined as the derivative of radiant flux divided by the differential solid angle:

$$I(\mathbf{x}, \Omega) := \frac{d\Phi(\mathbf{x}, \Omega)}{d\Omega}, \tag{2.6}$$

and quantified in units of $[W/sr]$ (sr = steradian). We will get back to intensity in Section 2.3 to describe light source intensities.

2.2.2 Photometric Quantities

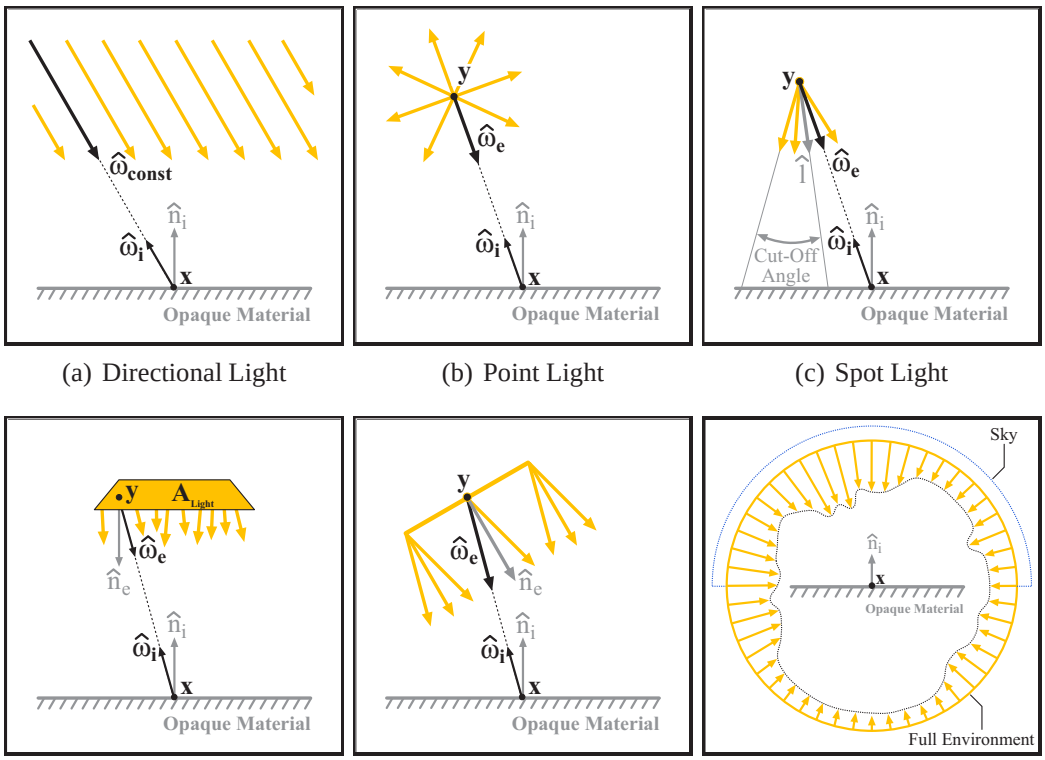
To obtain the photometric counterparts of radiometric quantities, the spectrum needs to be factored by the sensitivity of the human visual system. Photometry therefore describes radiant energy with respect to the receptive capabilities of the human eye. Our retina consists of two different photo receptors: *rods* $(\approx 120\ mio.)$ and *cones* $(\approx 6-7\ mio.)$. Rods are insensitive to color and responsible for night vision, whereas cones provide color sensation. As a result from measurements cones can be classified according to their response to different wavelengths into ”red” (64%), ”green” (32%), and ”blue” cones (2%) [see Nave, 2006].

When we now convolve the energy within the spectrum from 380 and 740 nanometers (nm) with these three color sensitivity curves, we obtain colors in the ranges of violet $(380-435\ nm)$, green $(520-565\ nm)$, to red $(625-740\ nm)$ respectively.

2.3 Common Illuminats

We address the energy emission characteristics and partition sources of visible light into the most common light models used in Computer Graphics.

In an endeavor to generate realistic imagery, adequate description of illuminants is indispensable. Among other elements (e.g. light-matter interaction see Section 2.4), physically correct simulation of real world lighting conditions also requires physically accurate models of any such source. Unfortunately, this is often impossible or infeasible with regard to both acquisition and rendering time. Hence, most rendering systems resort to simple light source models to sufficiently mimic the behavior of natural or manufactured lights (e.g. the sun or light bulbs). An important theory to enable simplifications is the *near-field* and *far-field* theory. As it also has an important influence on secondary lighting effects such as accurate shadows we present more information on this theory after the light classification.



(a) Directional Light (b) Point Light (c) Spot Light

(d) Area Light (e) Linear Light (f) Environment Light

Figure 2.2: Various light source types commonly used in Computer Graphics. Light source from (a)–(c) emit energy either from a singular or an infinitely distant location. In contrast, lights in (d) and (e) have a spatial extent and (f) is a special case using an entire sphere of illumination.

2.3.1 Light Source Models

Typically, graphics systems categorize lights into a few classes. We illustrate the most important of these models relevant to our shadow algorithms in Figure 2.2 along with their individual geometric configuration. Please note our discussion **does not** take visibility into account. Hence, equations describe intensity calculations only and omit shadow effects. Due to the relevance of visibility we devote a separate section to this topic and how it integrates into the rendering process in Sections 2.5 and 2.9.1

Directional Lights are considered to be infinitely far away from a receiving surface. This has two major consequences. The emitted light rays can be considered as parallel, and as a directional light would have to have infinite intensity to account for its infinite distance such lights use a constant intensity term per light

ray. Therefore, they are completely defined by their light direction and constant intensity. Figure 2.2 (a) gives an example. Its intensity is:

$$E(\mathbf{x}) = I_{const}(\infty, \Omega_{const}) \ \langle \Omega_i \cdot \hat{n}_i \rangle \, . \tag{2.7}$$

Point Lights describe light emission which is radiated from a single point in space into all directions (see Figure 2.2 (b)). They are slightly more expensive than directional light as the renderer has to compute the vector to the light per shading point. The total intensity of a uniform point light is given as:

$$\begin{aligned} \Phi(\mathbf{y}) &= \int_{S^2} I(\mathbf{y}, \Omega) \, d\Omega_o \\ \Phi(\mathbf{y}) &= I(\mathbf{y}) \, 4 \\ I(\mathbf{y}) &= \frac{\Phi(\mathbf{y})}{4}. \end{aligned} \tag{2.8}$$

The irradiance at $\mathbf{x}$ due to a single point light is derived as follows:

$$\begin{aligned} E(\mathbf{x}) \, dA &= I(\mathbf{y}, \Omega_e) \, d\Omega_e \\ E(\mathbf{x}) &= I(\mathbf{y}, \Omega_e) \, \frac{cos}{r^2} \\ E(\mathbf{x}) &= \frac{\Phi(\mathbf{y}, \Omega_e)}{4} \frac{\langle \Omega_i \cdot \hat{n}_i \rangle}{r^2}, \end{aligned} \tag{2.9}$$

where r is the distance between the light source and receiver point. We implicitly assume to take the $max(\langle \Omega_i \cdot \hat{n}_i \rangle, 0)$ to avoid lighting surfaces which are actually back-facing.

Spot Lights were introduced by Warn [1983] and are similar to point lights but offer more control over the light distribution than just a position. Spot lights are steerable with respect to an illumination direction and cone and resemble real spot lights used to light theater stages for example. Figure 2.2 (c) illustrates a spot with a given *cut-off* angle that defines the light opening. The irradiance due to a spot light cause is given as:

$$E(\mathbf{x}) = \frac{\Phi(\mathbf{y}, \Omega_e)}{4} \frac{\langle \Omega_i \cdot \hat{n}_i \rangle}{r^2} \, s_e(\hat{l}, \Omega_e), \tag{2.10}$$

where the $s_e(\hat{l}, \Omega_e)$ term computes if $\mathbf{x}$ actually falls within the cone of illumination or not and $\hat{l}$ is the spot direction. This is often called the spot light factor:

$$s_e(\hat{l}, \Omega_e) = \begin{cases} 1 & \text{if} \quad \langle l \cdot \Omega_e \rangle \ \leq \\ 0 & \text{otherwise} \end{cases} \quad . \tag{2.11}$$

Area Lights are the most important class of lights for realistic image synthesis because practically every real light source has a spatial extent. A solution to area light source support was first presented by Nishita & Nakamae [1983] and enhances the look of virtual scenes by adding a natural appearance due to realistic shadow effects such as umbra and penumbra (see Chapter 3.1). We show a simple rectangular example in Figure 2.2(d), however the illuminant can have an arbitrary shape with a finite spatial dimension. The irradiance at $\mathbf{x}$ incident from an area light is the integral over the light surface/area:

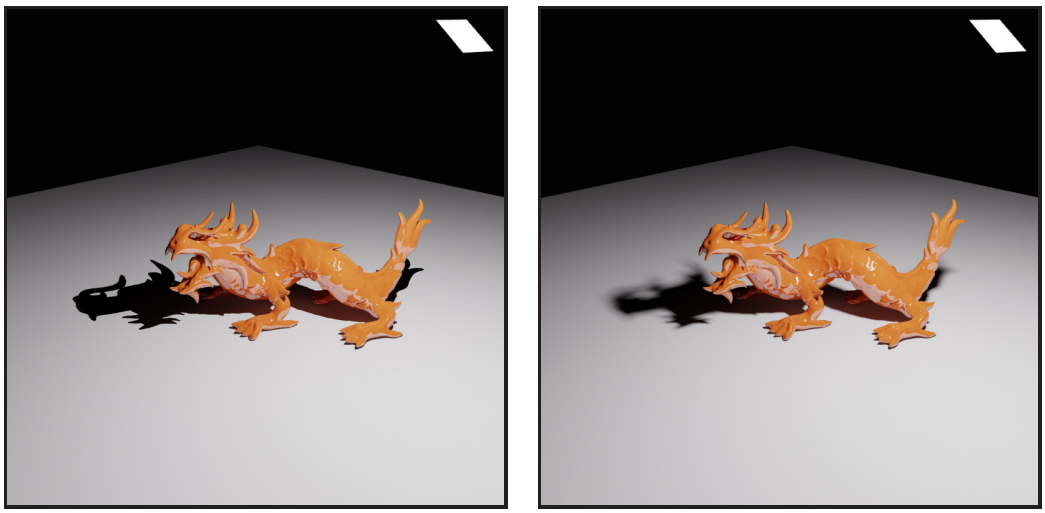
$$E(\mathbf{x}) = \int_{A_{Light}} L_e(\mathbf{y}, \Omega_e) \frac{\langle \Omega_e \cdot \hat{n}_e \rangle \langle \Omega_i \cdot \hat{n}_i \rangle}{\|\mathbf{x} - \mathbf{y}\|^2} dA_{\mathbf{y}}. \tag{2.12}$$

Linear Lights are similar to area lights and often used to represent long thin light sources. The difference between area lights is that their irradiance at $\mathbf{x}$ integrates over a line segment instead of an area. Nishita et al. [Nishita et al., 1985] however present a method to integrate over a long and very thin rectangle. A depiction is shown in Figure 2.2 (e) and their formula can be derived, as afore mentioned, by replacing the integration domain in Equation 2.12 by the a line segment $\mathscr{L}$ instead of an area.

Environment Lights are ideal light models to represent illumination arriving at a point $\mathbf{x}$ from an entire environment or sky. Usually, the environment is captured by taking pictures of a perfectly specular ball. Here, images are taken at different exposure times to later reconstruct high-dynamic range (HDR) images [Wyckoff & Feigenbaum, 1962; Debevec & Malik, 1997] for more realistic representation of the surroundings. Figure 2.2 (f) presents a complete environment (Ω) and the upper hemisphere for sky lighting Ω^+. Formula 2.13 computes its irradiance.

$$E(\mathbf{x}) = \int_{\Omega^+} L_i(\mathbf{x}, \Omega_i) \langle \Omega_i \cdot \hat{n}_i \rangle \; d\Omega_i \tag{2.13}$$

Goniometric Diagram are not very common in real-time rendering but a popular method to represent emission characteristics of realistic light sources. Goniometric diagrams [Kaufman, 1987; Verbeck & Greenberg, 1984] capture a single planar slice through the light's energy distribution. Each diagram describes the radiation with respect to a certain angle. Most point lights (e.g. light bulbs) can be described by one goniometric diagram owing to their rotational symmetric geometry. Note that only a light source's far-field is measured because only a single point of energy emittance is considered.



(a) Far-Field Shadows (b) Near-Field Shadows

Figure 2.3: Near-field and far-field effects on shadows for extended light sources. Far-field assumption reduces light emission to a single point (direction) and causes hard shadow. The near-field description properly accounts for the light's extent and produces correct soft shadows. Note that the overall illumination in fully visible regions is equivalent for the near- and far-field.

2.3.2 Discussion

Most light sources used for real-time/interactive shading pipelines are simple models (e.g. directional and point or spot lights) because of their efficient lighting evaluation. These models are particularly useful for computer animation film studios because their scenes are entirely computer generated and do not contain any real characters or environments. The shots in such movies are often lit by dozens or even hundreds of lights to create the desired ambiance in a sequence. Even with the use of indirect lighting or bounce lighting the production needs to have cinematographic freedom to tweak lighting in such a way that it integrates with story telling and overall style choice. Having many simple but efficient lights becomes then more important than having physically accurate lighting.

We will exploit this fact and implement our methods using spot lights mostly though our algorithms are applicable to other light types (directional and point lights) too.

2.3.3 Near-Field and Far-Field Theory

As mentioned before, virtually every light source we normally encounter has some finite spatial extend and their emission characteristics can be defined by a function $L(\mathbf{y}, \Omega_e)$. Here $\mathbf{y}$ is any position within the light's extent (see Figure 2.2) and Ω_e defines the radiation direction at $\mathbf{y}$ (we adopt a notation similar to Gösele [2004]). $L(\mathbf{y}, \Omega_e)$ defines the light source's *near-field*.

Unlike this, when a light and receiving surface are at least five times [Ashdown, 1995] the light's maximum extend apart from each other, the spatial dependence of $L(\mathbf{y}, \Omega_e)$ can be dropped without discernible differences [Murdoch, 1981] in the illumination (however it does affect secondary effects like shadows, see next paragraph). This yields a new function $L(\Omega_e)$ only depending on the angular distribution of energy. $L(\Omega_e)$ then encodes a light's *far-field*.

Before we proceed we need to discuss the impact of the near- and far-field theory on secondary lighting effects such as shadows. For directional, point, and spot light models, a far-field description is sufficient for computing the irradiance as well as shadows because the visibility computation only involves a simple binary function[2]. Either the ray from the surface to the light source position $\mathbf{y}$ (for directional light there is only one direction for all surface locations) towards $\mathbf{x}$ is blocked by an obstacle or not. As a result their shadows form sharp discontinuities without any penumbra areas[3].

For extended luminaires the far-field is not sufficient to model shadow effects correctly. When visibility is computed at a receiver point, the light can not only be visible or blocked but also partially visible. This is the reason why extended light sources cause penumbra and explains their importance for realistic image generation. To illustrate the difference we show a dragon model in a scene equipped with a quadratic area light at more than five times the distance of its maximum extend away from the dragon in Figure 2.3. Shadows in Figure 2.3 (b) are computed using the light's near-filed description and exhibit the expected shadows including umbra[4] and penumbra. Figure 2.3 (a) shows the same rendering using the far-field and contains crisp shadows only.

These final remarks conclude our discussion on light sources and we will present more information on the nature of shadows in Chapter 3. We would like to point the reader interested in more details on light source acquisition and representation to the Ph.D. thesis of Gösele [2004] and the work of Poulin [1993] as valuable resources of this filed of research. We are now going to review fundamental reflection properties.

[2] Chapter 3 provides a detailed discussion on shadow computation.
[3] Penumbra is the transition between fully lit and completely dark regions. See Chapter 3.
[4] Umbra is an entirely dark region where no light arrives. See Chapter 3.

2.4 Concepts of Surface Reflections

In rendering images are generated as if the scene was observed by a virtual camera. To obtain the color of the individual pixels of a raster image, the renderer has to compute the radiance reflected from the matter visible through each of these pixels along the direction towards the camera. To attain an insight on how surfaces interact with light we need to take a closer look at their material properties and how these material compositions alter or reflect incoming light.

2.4.1 Bidirectional Reflectance Distribution Function

The mathematical formulation of surface reflectance is accomplished by the *bidirectional reflectance distribution function*, abbreviated as BRDF. It was introduced by Edward Nicodemus [Nicodemus, 1965] in 1965 and describes the relation between reflected and incoming energy at a surface position $\mathbf{x}$ with respect to an incident and outgoing direction.

More formally, a BRDF describes the ratio of differential outgoing radiance $dL_o(\mathbf{x}, \Omega_o)$ to the differential irradiance $dE(\mathbf{x})$ at $\mathbf{x}$. An important fact inherent in this definition is that a BRDF is only capable of modeling energy reflected from opaque surfaces. It is not suitable to model matter with transmittance or scattering behavior. Such materials require a more advanced description called a bidirectional scattering-surface reflectance distribution function or BSSRDF. We only review the BRDF and refer to more sophisticated reflectance models later at the end of this section.

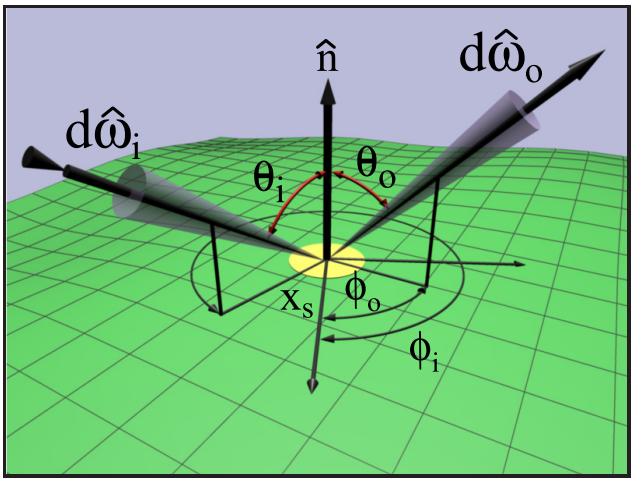
The BRDF, neglecting wavelength , is a six-dimensional function, two dimensions for each, the position on the surface, the incoming and outgoing directions Ω_i and Ω_o respectively. It is measured in $[1/sr]$, and formally written as:

$$f_r(\mathbf{x}_s, \Omega_i \rightarrow \Omega_o) = \frac{dL_o(\mathbf{x}_s, \Omega_o)}{dE(\mathbf{x}_s)} = \frac{dL_o(\mathbf{x}_s, \Omega_o)}{L_i(\mathbf{x}_s, \Omega_i)\ cos\ _i\ d\Omega_i}, \tag{2.14}$$

where $\mathbf{x}_s$ defines the $2D$ position on the surface and the directional dependence for both $\Omega_i = (\ _i, \Phi_i)$ and $\Omega_o = (\ _o, \Phi_o)$ is expressed in polar coordinates (being the polar and Φbeing the azimuth angle). The diagram in Figure 2.4 (a) illustrates the geometry of the BRDF. This general model accounts for both spatially and rotational variation and is called a *shift-variant* anisotropic BRDF.

Two important conditions must hold in order to make a BRDF a physically correct model and to enable simulation of realistic materials [Nicodemus, 1965; Wolff et al., 1992]. The first condition is the Helmholz reciprocity or symmetry condition:

$$f_r(\mathbf{x}_s, \Omega_i \rightarrow \Omega_o) = f_r(\mathbf{x}_s, \Omega_i \leftarrow \Omega_i) \tag{2.15}$$



(a) Geometry of a BRDF

Figure 2.4: The nomenclature of a BRDF and its parameters.

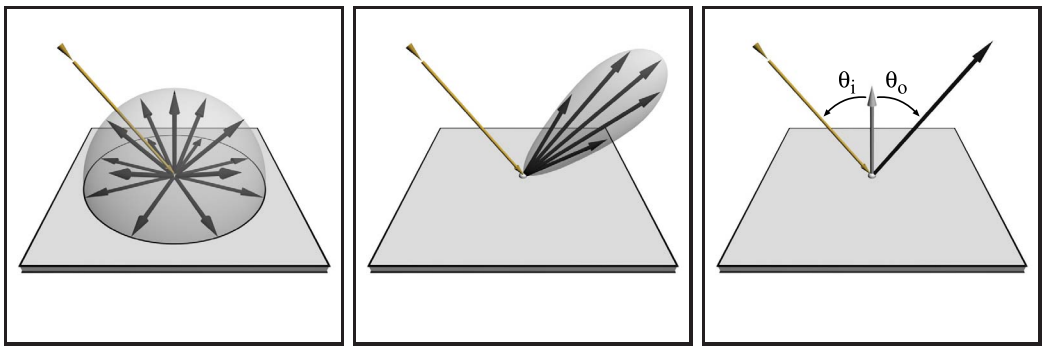
which states that the incoming and outgoing directions for the light transport can be exchanged ($\Omega_i \leftrightarrow \Omega_i$). The second condition that must hold is energy conservation:

$$\int_{\Omega^+} f_r(\mathbf{x}_s, \Omega_i \rightarrow \Omega_o) \leq 1. \tag{2.16}$$

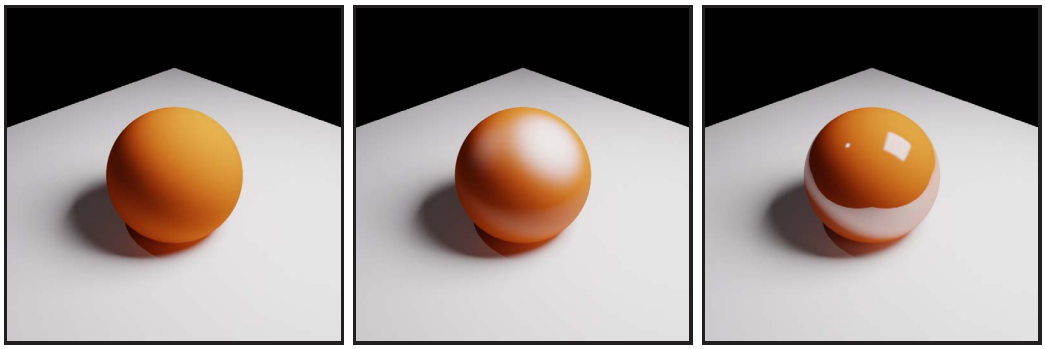
Energy conservation is important because it respects the fact that real materials do not reflect more energy than they receive. This means when we integrate the reflected energy over the upper hemisphere at $\mathbf{x}_s$ the total amount of energy must be less or equal to the incident energy.

The dimensionality of a BRDF as defined in Equation 2.14 can be reduced by two dimensions from six to four if the material is homogeneous. In other words the reflectance properties remain the same when its spatial location changes. Such BRDFs are called *shift-invariant* (as opposed to spatially varying) and only require directional information: $f_r(\Omega_i \rightarrow \Omega_o)$. Yet another dimension can be dropped in case the BRDF that does not change its reflection characteristics when the surface is rotated around the surface normal $\vec{n}$ at $\mathbf{x}_s$. This is called an *isotropic* BRDF $f_r(\mathbf{x}_s, \theta_i, \theta_o, \Phi_o - \Phi_i)$.

This review on BRDF theory is just a brief overview rather than an exhaustive discussion. More details on the complex subject of BRDF representations can be found, amongst many other sources, in the works of Nicodemus [1965]; Blinn [1977]; Wolff et al. [1992]; Glassner [1994]; Koenderink et al. [1996] as well as Kautz [2002], Lensch [2003], and the SIGGRAPH course from 2005 organized by Lensch and Gösele [Lensch et al., 2005]. Now, before we continue with more theory we would like to present a few examples of the most common BRDF models



(a) Diffuse BRDF (b) Glossy BRDF (c) Specular BRDF



(d) Diffuse Ball (e) Glossy Ball (f) Specular Ball

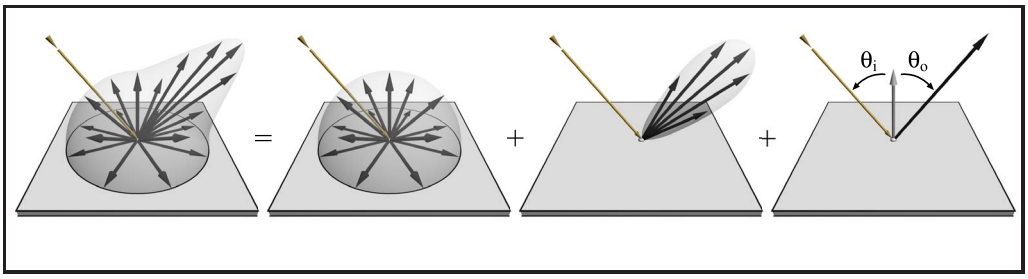
Figure 2.5: Material examples. We show plots of the reflectance model in the top row and rendered examples for each model in the row below. The rendered images where lit by a point (from left) and an area light (from right). Both light sources are clearly visible in image (f), blur out in (e), and finally vanish in (d).

found in real-time graphics.

2.4.2 Material Properties

Today, nearly every renderer supports materials with: *diffuse*, *glossy*, and *specular* reflections. More complex materials such as translucent, transparent objects or even human skin have very sophisticated properties and are known to be difficult to model and render (see Mertens et al. [2003a], Mertens et al. [2003b], Gösele et al. [2004], Jensen & Christensen [1998], Kautz [2003] and Hullin et al. [2008] for more information).

Diffuse Objects have a surface which shows a certain roughness such that incoming light is scattered (almost) uniformly in all directions and therefore do not



(a) Example of BRDF components

Figure 2.6: BRDF plot of a material that combines multiple reflectance properties.

cause any specular highlights. Consequently, diffuse materials are invariant under view direction changes. Often, diffuse surfaces are also referred to as Lambertian materials named after Johann Heinrich Lambert (August 26, 1728 - September 25, 1777). Chalk is a good example for a an almost Lambertian material. Figures 2.5 (a) and (d) show the emission characteristics of an ideal Lambertian surface as well as a rendered example.

Glossy Objects have a much smoother surface layer compared to its rough diffuse counterpart. Glossiness is caused by light leaving such objects scattered into a preferred direction and therefore created a shiny appearance. In contrast to diffuse materials glossy matter does have view-dependency. Examples for glossy sheen are finished wood or matte paint. Figures 2.5 (b) and (e) display the glossy lobe in which light reflected and a example rendering.

Specular Objects are view-dependent as glossy materials but have a completely smooth surface. The law of reflection (for opaque surfaces) states that a single ray incident at a surface point $\mathbf{x}_s$ under an angle , with respect to the normal at $\mathbf{x}_s$, is reflected (mirrored) off the surface under the same angle . Therefore, $_i = _o$. A mirror is a real world examples for perfect specularity. The reflectance properties and a computer generated example is shown in Figure 2.5 (c) and (f).

BRDF models can be very complex and are usually obtained by physically measuring material samples. Several characteristics can often be found in a single BRDF. Figure 2.6 shows an example where all three models from figure 2.5 have been merged.

Now that we have seen some example renderings of different BRDFs we turn back to the theory part again and discuss the formula that all realistic rendering systems try to solve.

2.5 The Rendering Equation

In 1986 Jim Kajiya [Kajiya, 1986] and David Immel [Immel et al., 1986] simultaneously presented the *Rendering Equation* to the Computer Graphics world. The rendering equation is an integral equation that describes the radiance equilibrium leaving a point as the sum of emitted and reflected radiance at that point. Similar to the BRDF it can be evaluated with respect to a given wavelength only. We will omit wavelength in our discussion and begin with its most basic form being defined as:

$$L_o(\mathbf{x}, \Omega_o) = L_e(\mathbf{x}, \Omega_o) + L_r(\mathbf{x}, \Omega_o). \tag{2.17}$$

The term L_e is only non-zero for surfaces that emit energy and are hence classified as light sources in the rendering process. L_r relates to the energy reflected off surfaces and must account for all incident illumination at $\mathbf{x}$. Filling in the integration over incident illumination at $\mathbf{x}$ reflected into Ω_o this formula expands into:

$$L_o(\mathbf{x}, \Omega_o) = L_e(\mathbf{x}, \Omega_o) + \int_{\Omega^+} L_i(\mathbf{x}, \Omega_i)\ f_r(\mathbf{x}, \Omega_i \rightarrow \Omega_o)\ cos\ _i\ d\Omega_i \tag{2.18}$$

where $cos\ _i$ is the cosine weighting term $\langle \hat{n}_i \cdot \Omega_i \rangle$. This form of the rendering equation integrates over the solid angle. Another viable option is to parameterize the equation over surfaces instead. To do so, the invariance of radiance is utilized:

$$L_i(\mathbf{x}, \Omega_i) = L_o(\mathbf{x}', \Omega_o) = L_o(\Psi(\mathbf{x}, \Omega_i), -\Omega_i). \tag{2.19}$$

Here Ψ is a ray-casting operator. It *casts* a ray from $\mathbf{x}$ into Ω_i and returns the closes surface point $\mathbf{x}'$ that was hit or infinity if nothing was intersected. Hence:

$$\mathbf{x}' = \Psi(\mathbf{x}, \Omega_i). \tag{2.20}$$

We assume $\mathbf{x}'$ to be implicitly given in the following formulation and omit the ray casting operator. To avoid confusion between Ω_o, which defines the direction of outgoing radiance from $\mathbf{x}$ with the outgoing energy from the surface $\mathbf{x}'$ we use the arrow ($\rightarrow$) notation similar the notation for angles in the BRDF. Then the rendering equation becomes:

$$L_o(\mathbf{x}, \Omega_o) = L_e(\mathbf{x}, \Omega_o) + \int_{\mathbf{x}' \in S} L_i(\mathbf{x} \leftarrow \mathbf{x}')\ f_r(\mathbf{x}, \Omega_i \rightarrow \Omega_o)\ G(\mathbf{x}, \mathbf{x}')\ dA_{\mathbf{x}'}, \tag{2.21}$$

where $G(\mathbf{x}, \mathbf{x}')$ is the *geometric term* which is responsible for the geometric arrangement of both differential surfaces taking their distance to each other, their orientation, as well as their mutual visibility into account:

$$G(\mathbf{x}, \mathbf{x}') = \frac{cos\ _{\mathbf{x}}\ cos\ _{\mathbf{x}'}}{\|\mathbf{x} - \mathbf{x}'\|^2}\ V(\mathbf{x}, \mathbf{x}'). \tag{2.22}$$

The visibility term $V(\mathbf{x},\mathbf{x}')$ computes whether $\mathbf{x}$ and $\mathbf{x}'$ can see each other or if their sight is occluded. V is piecewise function given as:

$$V(\mathbf{x},\mathbf{x}') = \begin{cases} 1 & \text{if} \quad \mathbf{x} \text{ and } \mathbf{x}' \text{ are mutually visible.} \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

In the case where the rendering equation is evaluated for one bounce only, all L_i's consist of L_e's which represent the light sources in a scene and basically corresponds to direct illumination. It only uses lights as sources of energy and neglects secondary effects such as color bleeding or caustics both effects of multiple light bounces. When these effects need to be reproduced it becomes necessary to recursively compute Equation 2.17 where lit surfaces after the first bounce become light sources themselves.

Given the various factors of the rendering equation one can imagine the heavy computational load involved solving the radiance equilibrium. Among all terms the ray casting operator $\Psi(\mathbf{x},\mathbf{x}')$ is by far the most expensive operation because it is needed to sample the surroundings to collect all surfaces $\mathbf{x}'$ from which radiance is either emitted or reflected towards $\mathbf{x}$.

Unfortunately, solving the rendering equation to an extent that reaches photorealism can easily take several hours just to compute a single picture even on todays most powerful workstations. This has led to several approximations in order to gain performance. The part of the rendering equation that has the highest potential for computational savings is the visibility term on which we focus on Part II and III.

2.6 The Framebuffer: Final Image Assembly

In Section 2.5 we have seen that the rendering equation needs to determine surfaces from which illumination emanates and surfaces which are not visible due to occlusion. In graphics two fundamental methods are available to sample surfaces and to implement the ray-casting operator to transforms a three dimensional scene description into a 2D image. One is *ray-tracing* and the other one is *scan-conversion* or *rasterization*.

Ray-tracing was introduced by Turner Whitted [Whitted, 1979] in 1979 and is an extension of ray casting (see Appel [1968] and Roth [1982] for ray-casting which has more restrictions compared to ray-tracing). Ray-tracing traces virtual rays of light from the observers eye position (the camera) through the image plane's pixel centers and intersects each ray with the scene geometry. As intersecting all polygons in a scene is not feasible, researchers have designed various forms of

hierarchical acceleration structures to increase intersection testing. However, the majority of these structure is no suited for dynamic objects and requires a rebuild process whenever objects change their position or shape. The ability to trace reflection, refraction, numerous shadow rays to light sources, and natural handling of transparencies [Porter & Duff, 1984] are the striking advantages of ray-tracing.

Rasterization or scan-conversion is based on Edwin Catmull's [Catmull, 1974] sorting technique called the *z-buffer* and operates in a different way than ray-tracing. First a view matrix is applied to every vertex of the scene representation to transform all objects into camera space. Then a second matrix transformation projects all polygons from view- into into screen-space. A scan-line algorithm then processes each polygon and computes its coverage on the pixel or raster grid. Each picture element is shaded by linearly interpolating the lighting results from the vertices. Here, the vertices' depth value is stored in an additional buffer, the *z-buffer* to resolve visibility.

Both techniques are point sampling algorithms and suffer from aliasing due to insufficient sampling rates and requires suitable anti-aliasing methods to reduce unpleasant artifacts.

2.7 Hardware Accelerated Rendering

Initially, dedicated hardware support was only available through expensive high-end graphics systems, such as the SGI Onyx system fitted with an Infinite Reality graphics accelerator. Yet even then the hardware support of this equipment was limited to certain parts of the rendering pipeline. This changed in August 1999 when the first *graphics processing unit* (GPU) was introduced to the consumer level hardware market. It integrates the entire graphics pipeline in one graphics chip and supports user programmability for some stages. Without making too much of a generalization these chips followed more or less the computational paradigm given in Figure 2.7.

The massive parallelism offered by GPUs is mainly due to the fact that their computation kernels work on single vertices or pixels only. As a result no connectivity information between vertices for instance is accessible. The same holds for pixel data too because a pixel's neighbors may not have been computed yet and imposing this dependency would ultimately hinder parallelism. It should be noted that this restriction has only been recently loosened by a new extension called a *geometry shader*.

Vertex Operations. Input data (graphics hardware processes polygonal data e.g.

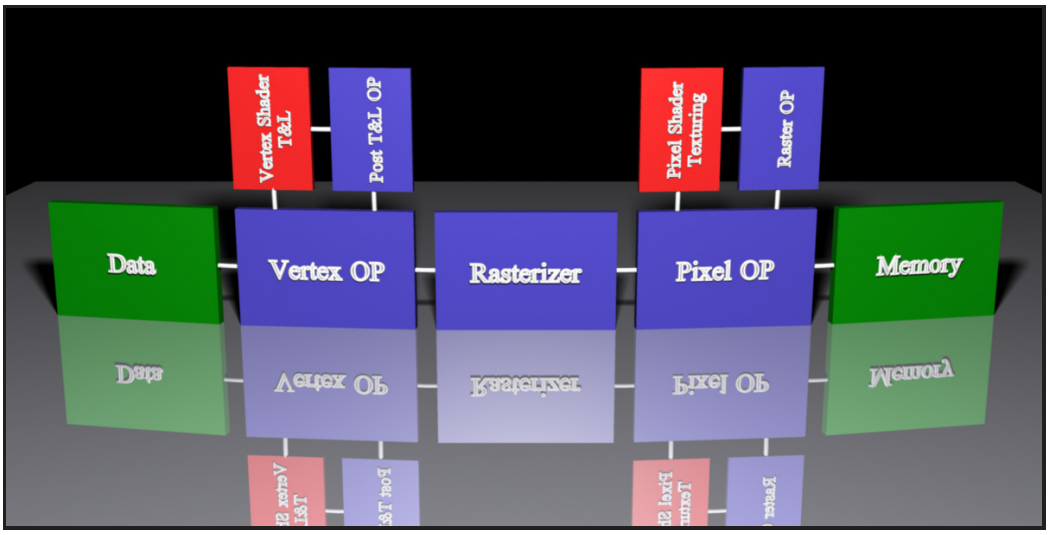


Figure 2.7: Graphics pipeline and hardware support for the pas two decades. Input primitives are streamed through the pipeline until the final rendered image is output to the framebuffer.

points, lines, and triangles) is streamed into the first block of the pipeline where all vertex related operations take place. This block can be divided into two major units, a *vertex shader* that performs transform and lighting, short (T&L), operations and a post T&L stage. T&L includes for example model/view transformations, texture coordinate assignment, and lighting. This unit was the first to allow the user to replace fixed wired functionality by customized programs. It opened the door for new vertex transformations, more flexible lighting, and even vertex-texture access. Post T&L functionality includes perspective correction, viewport mapping, and clipping. Note that if the user decides to bypass the hard-wired pipeline he is responsible to implement all hardware operations the vertex processing stage usual performs.

Rasterization. After all vertex operations are complete, the processed data is streamed into the *rasterization* unit. Here, all polygons are setup for conversion into a 2D raster image. This is sometimes called *triangle setup* and prepares vertex properties such as color, the perspective correction coordinate, and texture coordinates for interpolation. This is an important step because each pixel that the rasterizer generates has to carry a set of data that represents the interpolated values within the polygon at the pixel's 2D position to enable proper processing in subsequent pipeline modules. When all vertices have bee set up they are ready for scan-conversion and rasterized into pixels. The generated fragments, as pixels are also often called, then continue their journey to the next processing kernel.

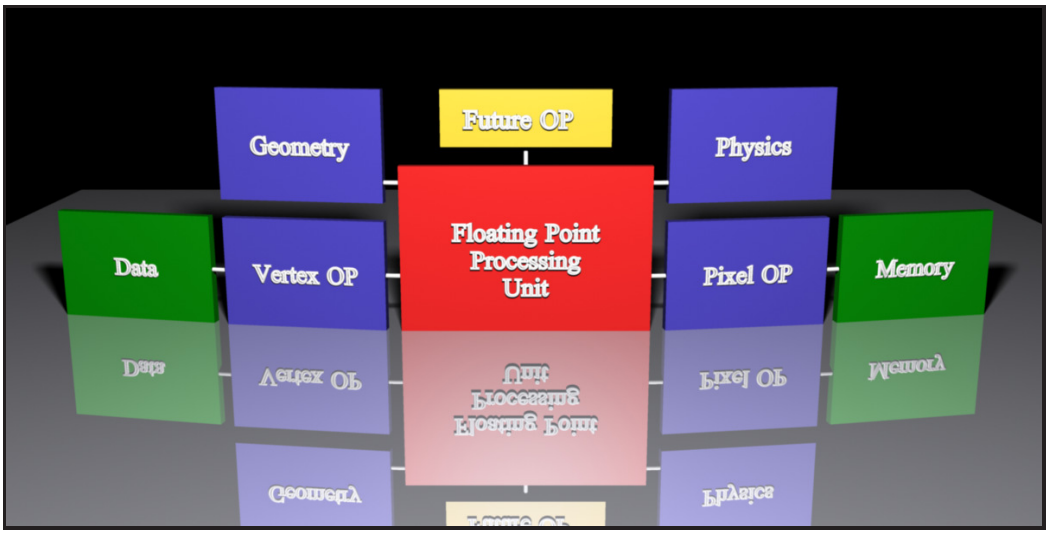


Figure 2.8: NVIDIA's latest graphics hardware chip offers more general processing power to accomplish graphics computations and supplies more flexibility.

Pixel Operations. This is the final unit before an actual result is eventually written to memory. All fragments generated during rasterization are subject to certain pixel operations which can be summarized as a *pixel shader* part and a *raster operations* part. The *pixel shader* was the second stage that became programmable and allows for customized (dependend) texturing, per-pixel lighting, and many other shading features. Though pixels leaving the pixel shader are properly shaded they are not yet sorted according to their spatial depth. Also, pixels can be transparent in which case their coverage must be accumulated when overlapping. This is why pixels are subject to further raster operations after shading. Raster operations include visibility tests, proper blending with color entries already resident in memory, as well as anti-aliasing and stencil tests.

The traditional pipeline from Figure 2.7 has experienced a conceptual change during the past few years. As hardware features advance and the demand for more flexibility grows, GPU designs change towards a more general processing unit as shown in Figure 2.8. In the old design a fixed number of processors with even different instruction set was dedicated to the vertex and pixel shaders. Nowadays however, GPUs possess a large number of processing units with almost the same instructions and they allocate these resources dynamically depending on where they are currently required. This naturally provides an automatic load balancing between computation units. It further increases the flexibility to add new operations and allows for more general purpose programming (GPGPU) to facilitate

the GPU as powerful multi-processor devices rather than a graphics accelerator.

The interested reader should consult the manufacturer's websites for more insights and white papers on modern graphics architectures at www.nvidia.com and www.amd.com. Intel is currently working on their new GPU called *Larrabee* which is going to use *x*86 instruction www.intel.com.

Mip-maps. All algorithms throughout Part II and III are entirely implemented on the GPU and utilize hardware accelerated features that are not dedicated to the rendering pipeline directly. The most important one is efficient *on chip* generation of mip-maps[5] introduced by Williams [1983]. It avoids streaming image data back and forth between the CPU and GPU and utilizes high performance imaging subsets on the GPU. Multi-resolution image pyramids [Burt, 1981; Williams, 1983] have a long history in the graphics and vision community and proven a great tool for pre-filtering to avoid disturbing popping artifacts, thereby increasing temporal coherence in renderings. Basically, an image pyramid encodes the repetitively pre-convolved results in each sub-level for the its parent image.

2.8 Shadows and Human Perception

Shadows are known to be among the most important visual cues for our spatial coordination. However, the role that the shadow quality plays in our humans judgment of spatial relationship is still a controversy among scientists.

Wanger [1992] conducted three experiments where test subjects where run through tests to analyze the effect shadow quality would have on certain tasks the users were asked to perform. The first two experiments focused on the subject's ability to estimate objects properties like size and position when shadows vary in shape and sharpness. The third experiment was designed to check if shadow sharpness had an influence on shape matching tasks. Wanger [1992] concludes that the shape and sharpness had no significant impact on the size and position estimation task but shadow sharpness did have an influence on the shape matching performance.

Similar to Wanger [1992], Kersten et al. [1994] conducted experiments that are concerned with the perception and influence of shadows in motion and presents remarkable results. Their results reveal that motion is shadow overrides other visual cues (object size consistency) and that the human system poses the constraint of a stationary light source. The latter is very interesting and happens because displaced objects and their moving shadows provide ambiguous information to the perceptual system. Our brain has to make implicit assumptions, e.g. the light

[5] Mip-map is an acronym for *multum in pravo* or *many things in a small place*.

source is fixed in space, in order to infer the spatial location of an object when moving.

In a more recent experiment Kozlowski & Kautz [2007] investigate if accurate occlusion is needed when rendering glossy reflections. Their experiment reveals that the geometry complexity of objects can be reduced enormously without producing noticeable differences in specular and glossy reflections.

2.9 Assumptions

We are now reaching the end of the background chapter and we would like to summarize the assumptions we incorporate into our shadow algorithms. We begin with the most important one, separating the visibility function from light source integration, followed by the remaining assumptions.

2.9.1 Visibility Computation

In Section 2.21 when we discussed the rendering equation we encountered the visibility term which computes mutual visibility between the current shading point $\mathbf{x}$ and the sampled surface position $\mathbf{y}$. This computation turns out to be the most expensive part of the rendering equation and causes a major bottleneck in the rendering process. The reason for this is that visibility has unfortunately no locality. Theoretically, any scene object can cause occlusion along the ray from $\mathbf{y}$ to $\mathbf{x}$.

Now, let us consult the rendering equation defined over surfaces 2.21 once more, in order to compute the irradiance at a certain point due to an area light source, now including the visibility term:

$$L_o(\mathbf{x},\Omega_o) = L_e(\mathbf{x},\Omega_o) + \int_{\mathbf{x}'\in S} L_i(\mathbf{x}\leftarrow\mathbf{x}')\ f_r(\mathbf{x},\Omega_i\rightarrow\Omega_o)\ G(\mathbf{x},\mathbf{x}')\ dA_{\mathbf{x}'}.$$

As we seek to compute the irradiance at $\mathbf{x}$ due to a single area light source, the rendering equation can discard the emitted energy term L_e, as $\mathbf{x}$ would in that case itself act as a light source, as well as the reflected energy f_r, leading us to:

$$E(\mathbf{x}) = \int_{\mathbf{x}'\in S} L_i(\mathbf{x}\leftarrow\mathbf{x}')G(\mathbf{x},\mathbf{x}')dA_{\mathbf{x}'}.$$

The invariance of radiance (Equation 2.19) and the ray casting operator (Equation 2.20) allow us to re-write the incident radiance L_i in terms of outgoing or emitted radiance as: $L_i(\mathbf{x}\leftarrow\mathbf{x}') = L_e(\Psi(\mathbf{x},\Omega_i),-\Omega_i)$, where the ray casting operator samples the light source area. From $\mathbf{y} = \Psi(\mathbf{x},\Omega_i)$ and $\Omega_e = -\Omega_i$ then follows:

$$E(\mathbf{x}) = \int_{A_{Light}} L_e(\mathbf{y},\Omega_e)\frac{\langle\Omega_e\cdot\hat{n}_e\rangle\,\langle\Omega_i\cdot\hat{n}_i\rangle}{\|\mathbf{x}-\mathbf{y}\|^2}\,V(\mathbf{x},\mathbf{y})\,dA_{\mathbf{y}}. \tag{2.24}$$

This means we have to integrate over the light's area and for each sample we must compute if energy from this sample actually arrives at $\mathbf{x}$ or if it is blocked and poses an enormous load on the shading process.

An assumption often made to reduce the expensive visibility evaluation is to factor the visibility out of the irradiance computation [Agrawala et al., 2000] and to multiply an approximate attenuation or shadow factor afterwards. Even though this is not physically correct it allows to control the overall visibility cost and the difference is often not noticeable. We can formalize this approximation as follows:

$$E(\mathbf{x}) = \tilde{V}(\mathbf{x}) \int_{A_{Light}} L_e(\mathbf{y}, \Omega_e) \frac{\langle \Omega_e \cdot \hat{n}_e \rangle \langle \Omega_i \cdot \hat{n}_i \rangle}{\|\mathbf{x} - \mathbf{y}\|^2} \, dA_{\mathbf{y}}. \tag{2.25}$$

We are then left with the approximate visibility factor $\tilde{V}$ which can be defined as:

$$\begin{aligned} \tilde{V}(\mathbf{x}) &= \frac{1}{A} \int_{A_{Light}} V(\mathbf{x}, \mathbf{y}) \, dA_{\mathbf{y}} \\ &\approx \frac{1}{N} \sum_{i}^{N} V(\mathbf{x}, \mathbf{y})_i, \end{aligned} \tag{2.26}$$

and allows to reduce the computational cost for shadowing drastically. Especially in real-time applications such as games the number of samples is often very low (sometimes it falls down to four or even just one sample) to achieve the highest performance. We will examine the shadow quality for varying number of visibility samples in Chapter 3.

The shadowing techniques we present all build on top of approximation from Equation 2.26 and we will show that our algorithms provide higher quality shadows and prevent exhaustive sampling through efficient pre-filtering capabilities compared to standard shadow mapping based techniques.
Besides an approximate visibility term we also rely on the following assumptions:

- we can currently not integrate the BRDF into our visibility sampling ,
- we discard indirect lighting effects and therefore also indirect shadows as recursive evaluation of the rendering equation would be necessary,
- our algorithms are currently not able to support textured light sources as described by Segal et al. [1992],
- we currently do not support volumetric or transparent shadow effects.

We deem these assumptions as an acceptable solution and a requirement to reduce computational expenses in order to move closer to real-time high quality shadow computation.

Chapter 3

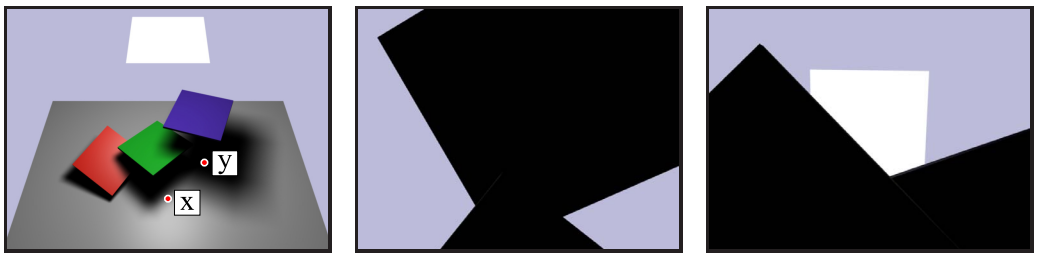
Shadow and Visibility Techniques

The computation and interpretation of visibility characteristics has a long history in the computer graphics community, and over the course of the past decades numerous articles on this topic have been published. The problem is especially complicated because of the non-local nature inherent in visibility computation which means that a small change in the spatial location or shape of any object potentially influences its entire surroundings.

In this chapter we will discuss the properties of shadows and review solutions to resolve visibility for shadow generation which are not directly related to Shadow Map filtering. We will review these methods separately in Chapter 4 where we elaborate on the differences of other filtering methods compared to your approaches. Even though it is not always possible to strictly divide these approaches into separate domains we strive to classify them into four categories. Algorithms that work in *image-space* or *object-space*, *hybrid methods* which combine algorithms and techniques based on *precomputation*. For the Image-Space section we will explain the major problem inherent in William's Shadow Maps in more detail to found a solid understanding for the difficulties of filtering Shadow Maps. In addition we would like to refer to Woo et al. [1990] and Hasenfratz et al. [2003] for excellent overviews on shadow algorithms.

3.1 Shadow Classification

By the words of Leonardo da Vinci "Shadow is the obstruction of light" da Vinci [1970]. The question that rises then is: "How much light is blocked at a given location in space?" To answer this question let us take a look at a simple example.



(a) Shadow Rendering (b) View towards light from **y** (c) View towards light from **x**

Figure 3.1: Principle shadow regions. A rendering of overlapping geometry lit by a large area light source (a). (b) shows that the light source is completely concealed when seen from **y** (umbra) whereas in (c) the light is only partially blocked when seen from **x** (penumbra).

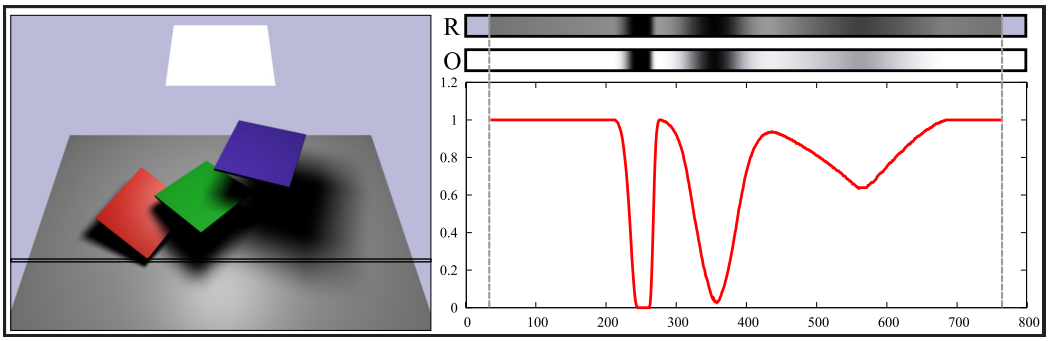
Figure 3.1 (a) shows three objects casting shadow on each other and on the ground plane. Note how the shadows vary depending on the relation between the occluder (an object that blocks light rays) and receiver. Now, let us take a closer look at two points **x** and **y**, marked in Figure 3.1 (a). If we were to place cameras at **x** and **y** and aim them towards the light source we would receive the images shown in Figure 3.1 (b) and (c). From **y** the light source is completely occluded as opposed to the rendering from **x** where the light source is partially visible. Both locations fall in two principle regions of shadow: **y** is in the "*umbra*" and **x** resides in the "*penumbra*" region.

Umbra is the Latin word for shadow and defines the region in shadow from where all light is blocked and is therefore the darkest part in shadow.

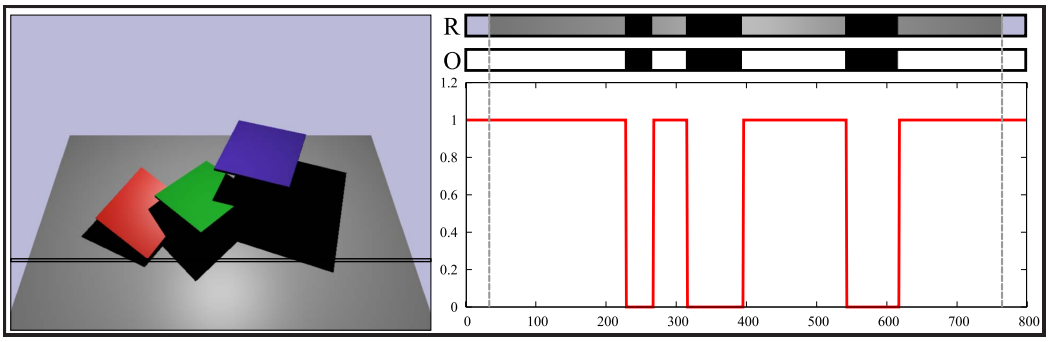
Penumbra is a combination of the Latin words *paenes* ("almost, nearly") and *umbra*. It defines shadow regions from where the light source is at least partially visible.

We can also translate the images from Figure 3.1 (b) into intensity plots where the percentage of the light area that is visible from a surface point **x** is given as a function of **x**. In Figure 3.2 (a) we show a single row of pixels R taken from Figure 3.1. For each pixel we determine its 3D world-space position **x** and plot the occlusion O of the light source as seen from that point.

This allows us to identify shadow regions simply by looking at the occlusion plot. Function values of $f(\mathbf{x}) = 0$ correspond to umbra regions, function values in the range of $0 < f(\mathbf{x}) < 1$ represent locations in penumbra areas, and values of $f(\mathbf{x}) = 1$ mark fully lit regions from where the entire light source is visible.



(a) Shadow rendering and occlusion plot for an area light source

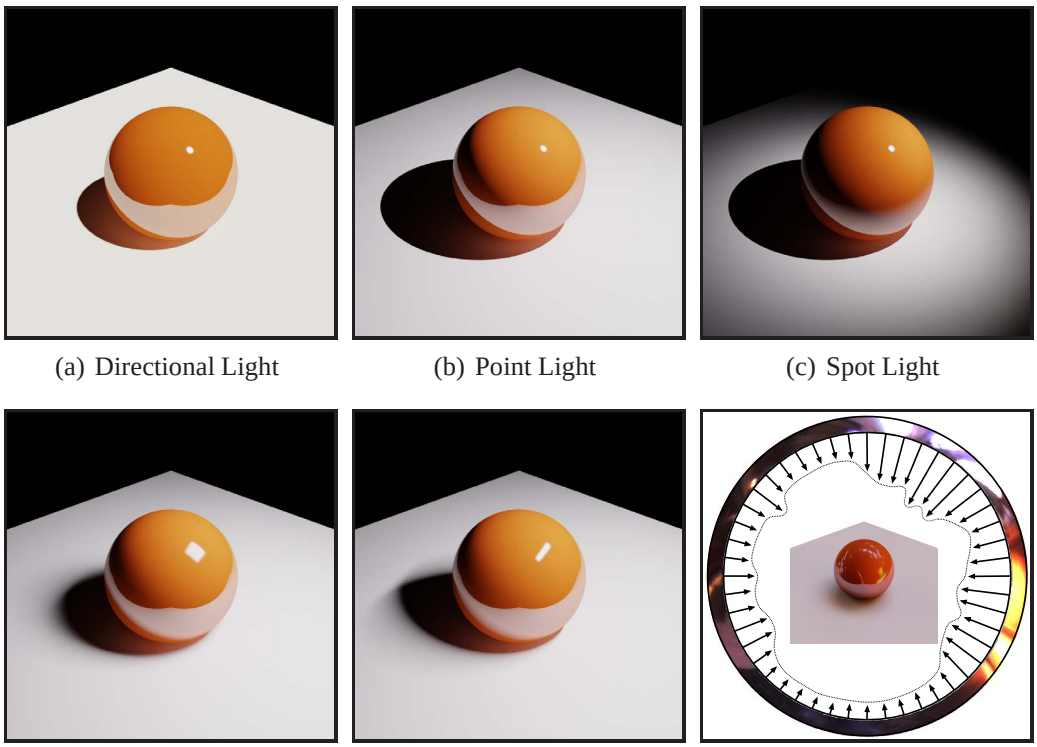


(b) Shadow rendering and occlusion plot for a point light source

Figure 3.2: Occlusion for a given row of pixels. The right part of (a) and (b) shows a close-up of the rendered results R of the focused pixel row and the occlusion plot O as a function of $\mathbf{x}$. Note that we stretched the pixel row in the close-ups in height.

An interesting comparison can be made when rendering the same scene using a point light instead of an area light shown in Figure 3.2 (b). Now we encounter umbra regions exclusively because visibility computation reduces to a simple binary function. One only needs to check if a single ray from $\mathbf{x}$ to the point light is occluded or not. This is why all shadows have hard edges and the occlusion plot shows that $f(\mathbf{x})$ becomes a piecewise constant function for point lights of either $f(\mathbf{x}) = 0$ or $f(\mathbf{x}) = 1$.

In additional to providing information on whether or not a point is located in an umbra or penumbra region, for area light sources the occlusion plot from Figure 3.2 (a) also informs about the spatial relation between objects and the area light. Inspecting the inclination of the occlusion curve tells us about the relative distances between the occluder, receiver, and light source. Given the distance between the light and receiver we can estimate the relative location of an occluder between the light and receiver. For example the red rectangle is relatively close to



(a) Directional Light (b) Point Light (c) Spot Light

(d) Area Light (e) Linear Light (f) Environment Map Light

Figure 3.3: Shadows and illumination as a result of a sphere being lit by all the different light source types we discussed in Section 2.3.1. Note the we extended the point and linear light spatially to appear as reflections on the specular sphere.

the receiver plane which is reflected in the slope of $f(\mathbf{x})$ from pixels $\approx 200-300$. In contrast to this pixels from $\approx 450-650$ have a much smaller inclination which tells us that the blocking object is located further away from the receiver plane.

We now conclude our shadow classification with an overview on the different kind of shadows produced by the light sources we presented in Section 2.3 (see Figure 2.2). Figure 3.3 shows a shadow rendering of a specular sphere for each light type. As aforementioned, simplified light source models whose energy is emitted from an infinitesimal point e.g. Figure 3.3 (a-c) are limited to hard shadow edges whereas Figure 3.3 (d-f) provide an increased visual fidelity as they incorporate more realistic light sources with a finite extent and all-frequency shadows. Further we would like to refer to the Ph.D thesis of Durand [1999] for an exhaustive analysis on visibility.

3.2 Image-Space Methods

In this section we review image-space approaches which constitute important and popular means for visibility computation. In general this class of algorithms has the advantage, compared to methods that operate in the object space, that they are general, efficient, and simple to implement. However, image-based methods also have their shortcomings. Their disadvantages rank among aliasing artifacts and additional memory consumption. These methods are the subject of the following sections.

3.2.1 Z-Buffer Algorithm

The majority of image-space shadow and visibility algorithms are founded on the *Z-Buffer* algorithm introduced in 1974 by Edwin Catmull [Catmull, 1974]. The Z-Buffer technique is a sorting scheme to resolve visibility using an additional image buffer that stores depth information which is utilized during rasterization. The basic idea is trivial and can be summarized as follows:

- Allocate an additional monochrome[1] Z-Buffer Z with a resolution equivalent to the framebuffer size.
- Initialize Z with furthest depth value possible.
- For each rasterized pixel $P_{(i,j)}$ check if $P_{(i,j)}.z < Z_{(i,j)}$. If true replace $Z_{(i,j)}$ by $P_{(i,j)}.z$ and update framebuffer with $P_{(i,j)}.rgba$.

There were two concerns at the time the Z-Buffer was published. One issue relates to the additional memory required to store the z-values and the overhead of writing sample updates into this buffer. The second one emerges from the third item from the above listing. All pixels undergo expensive shading calculations before the visibility test happens. As a result many pixels will be shaded but never written to the framebuffer.

But only a few years later in the early 1980's Silicon Graphics began to provide special hardware support for z-buffering. This had a significant impact on the CG community and influenced hardware architectures up to nowadays latest graphics chips which also support an early-z culling prior to pixel shading to avoid unnecessary computations. Accompanied by cheaper and faster memory the Z-Buffer became the state-of-the-art image-space method for visibility computation.

An important extension of the Z-Buffer algorithm is the *Hierarchical Z-Buffer* by Greene et al. [1993]. Greene et al. exploit two kinds of coherence to speed up the rasterization process significantly. Object-space coherence is achieved by

[1] Colors channels are irrelevant for opaque surface depth sorting.

partitioning the scene geometry into an octree where each node is surrounded by its bounding box. Image-space coherence is utilized by managing an image pyramid of the Z-Buffer. Here each pyramid level above the finest contains the maximum z-value of the 2×2 window above. Rendering a node from the octree works as follows. First, get the hierarchical Z-Buffer entry from the pyramid level that covers covers the node's bounding box in screen-space. If the nearest depth of the bounding box's faces is farther away than the z-value from the Z-Buffer pyramid discard the entire node. If not step inside the node and keep recursively executing this scheme until a node is either rejected or it has reached the finest Z-Buffer level and therefore will be rasterized. After rasterization depth changes are propagated throughout the Z-Buffer pyramid.

3.2.2 Shadow Maps

Only four years after Edwin Catmull's introduction of the Z-Buffer in the mid 1970's Lance Williams introduced *Shadow Mapping* [Williams, 1978] which had great impact on the field of Computer Graphics. Shadow Mapping can be seen as an extension of Catmull's Z-Buffer. Williams' method however generates the depth buffer from the light source's vantage point, thereby recording the distance to the closest surfaces with respect to the light instead of the eye camera. He calls this form of a Z-Buffer a *Shadow Map* and employs it to perform shadow queries. Nowadays, shadow mapping has grown into a de facto standard for rendering shadows in movie productions and video games.

Let us now take a closer look at how Shadow Maps are being used to create shadowed images. For this purpose let us consider the world-space position $\mathbf{x}$ of a given camera pixel $\underline{\mathbf{x}}_\mathbf{c}$ shown in Figure 3.4. In order to decide if $\mathbf{x}$ is shadowed by any occluding geometry (a helix in our example) we need to check if there is any object located in between the light and $\mathbf{x}$. This is where the Shadow Map comes into play. Point $\underline{\mathbf{x}}_\mathbf{l}$ represents the position of a shadow map pixel, which is obtained via a surjective mapping $T : \mathbb{R}^3 \to \mathbb{R}^2$ between world-space and Shadow Map, such that:

$$\underline{\mathbf{x}}_\mathbf{l} = T(\mathbf{x}). \tag{3.1}$$

This mapping basically warps a pixel into Shadow Map space via a perspective projection. The Shadow Map itself encodes a function $z(\underline{\mathbf{x}}_\mathbf{l})$, that represents the depth of the blocker that is closest to the light source for each $\underline{\mathbf{x}}_\mathbf{l}$. A camera pixel $\underline{\mathbf{x}}_\mathbf{c}$ with world-space position $\mathbf{x}$ is considered in shadow when $d(\mathbf{x}) > z(\underline{\mathbf{x}}_\mathbf{l})$, with $d(\mathbf{x})$ being the depth of $\mathbf{x}$ (again, with respect to the light source). See Figure 3.4. We can now formally define a shadow function s:

$$s(\mathbf{x}) := f(d(\mathbf{x}), z(\underline{\mathbf{x}}_\mathbf{l})) = \begin{cases} 1 & \text{if } d(\mathbf{x}) \leq z(\underline{\mathbf{x}}_\mathbf{l}) \\ 0 & \text{if } d(\mathbf{x}) > z(\underline{\mathbf{x}}_\mathbf{l}), \end{cases} \tag{3.2}$$

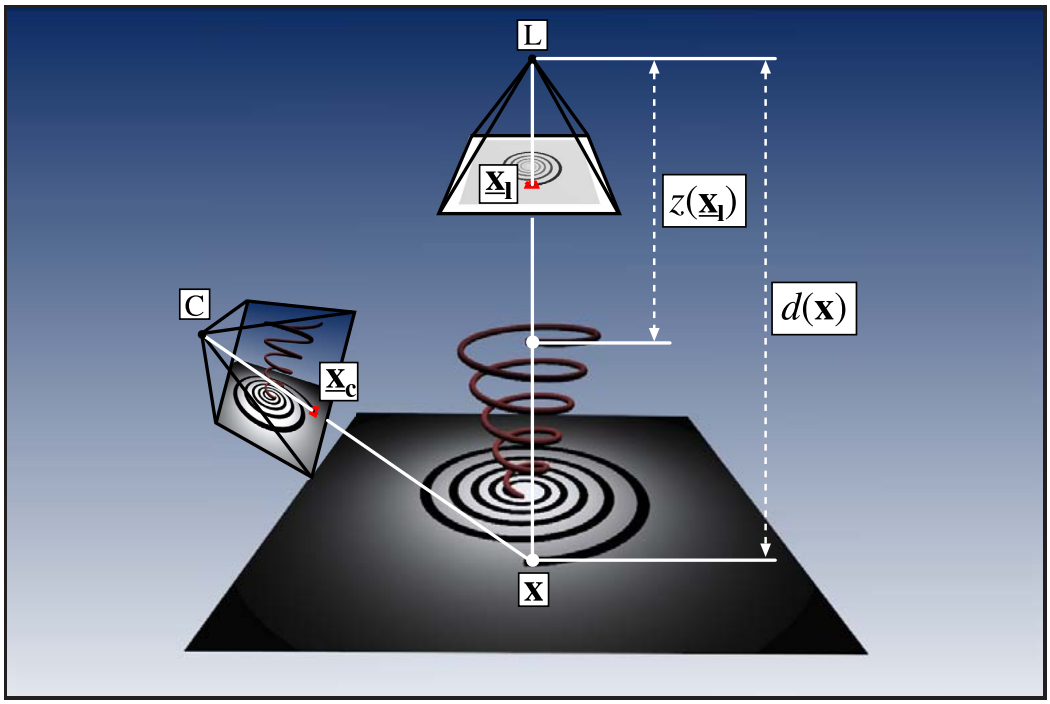
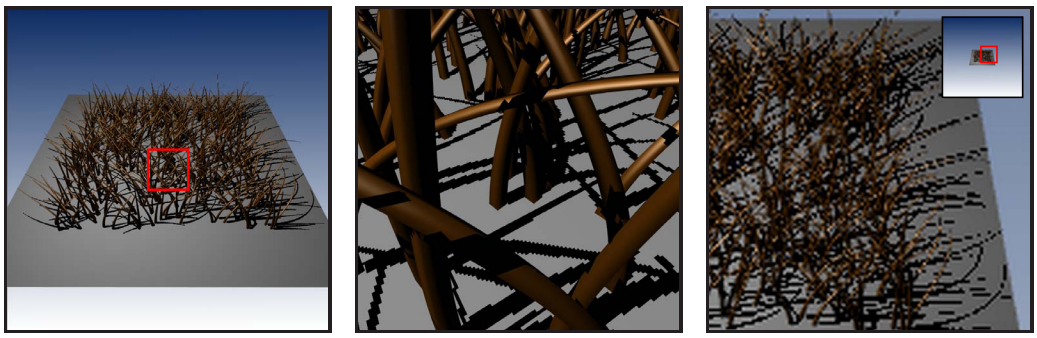


Figure 3.4: Shadow Mapping. Computing the binary visibility function $s(\mathbf{x})$ requires a comparison between $z(\underline{\mathbf{x}}_\mathbf{l})$ and $d(\mathbf{x})$. If $z(\underline{\mathbf{x}}_\mathbf{l}) < d(\mathbf{x})$ then $\mathbf{x}$ is shadowed, otherwise it is lit.

that basically encodes the visibility test. For the remainder of this dissertation we will often use the scalar notation $f(d,z)$ to abbreviate $f(d(\mathbf{x}),z(\underline{\mathbf{x}}_\mathbf{l}))$. Adding shadows to a scene rendering with Shadow Maps then reduces to the evaluation of $f(d,z)$ for each camera pixel.

Due to its purely image-based nature, Shadow Mapping is a versatile shadow algorithm robust against increased scene complexity. It is also very general because it supports any primitive that can be rasterized and its simplicity allows it to translate very well to graphics hardware. For example, Segal et al. [1992] show that the OpenGL texture pipeline can be utilized to implement Shadow Mapping in graphics hardware, and all modern GPUs support a hardware shadow test using special shadow texture lookup functions (Zhang [1998] propose a workaround when the texture pipeline load becomes too high).

Unfortunately, Shadow Mapping also has its problems. Since the blocker geometry is discretized into a finite resolution image, aliasing artifacts can occur which decrease the overall image quality and become disturbing in animated sequences. There are mainly two sources for Shadow Map aliasing which are shown in Figure 3.5.



(a) Rendering of fine details (b) Insufficient Resolution (c) Unfiltered Reconstruction

Figure 3.5: Shadow Mapping issues. Discretization artifacts shown in (a) are due to an insufficient depth map resolution. (b) shows the lack of Shadow Map filtering resulting in severe flickering.

One source for aliasing is an insufficient shadow map resolution which causes discretization artifacts noticeable as jagged edges illustrated in Figure 3.5 (b). Such artifacts are the result of Shadow Map under-sampling where the depth buffer resolution is not capable to encode enough spatial information to provide a shadow sample per camera pixel. It often happens that several camera pixels project into the same depth buffer pixel causing block artifacts.

The second source for aliasing might be less obvious and stems from the fact that Shadow Maps **can not** be filtered in the same manner than ordinary surface textures can be. To confirm this fact we can simply conduct the experiment and apply a blur filter to the depth values of a Shadow Map and inspect the results. For this purpose we rasterize a helix into a Shadow Map as shown in Figure 3.6. We then render with regular Shadow Mapping enabled (left close-ups), and with the filtered version of the Shadow Map (right close-ups).

Though it seems tempting to simply filter the z-values it *does not* produce the desired outcome, as one wants to filter the **results** of the shadow test and not just the depth values. Reeves et al. [1987] were the first to switch the order of filtering and testing. This has two significant consequences:

- Shadow Maps **can not** be pre-filtered e.g. high quality texture map filtering based on mip-mapping [Williams, 1983] is not directly applicable since the result of the filter cannot be pre-computed.

- Consequently, explicit run-time filtering is necessary to reduce screen-space aliasing, which becomes very expensive with growing filter sizes.

While this comes at a foreseeable cost for small up-sampling kernels to alleviate discretization artifacts, down-sampling can potentially require to filter very larger

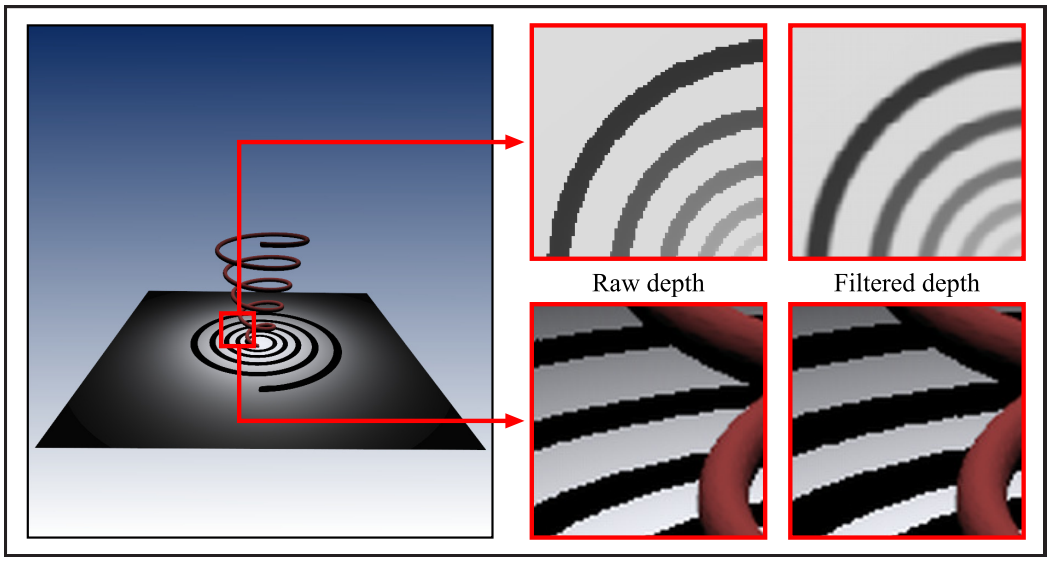
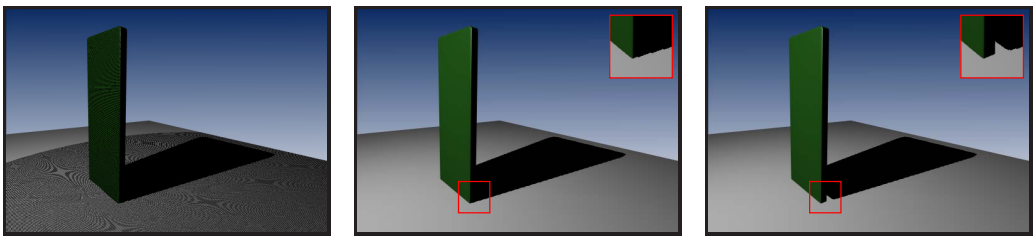


Figure 3.6: Left images show a regular Shadow Map close-ups whereas the right close-ups show a filtered Shadow Map and the resulting shadows. As we can see: filtering depth values does not produce a filtered shadow.



(a) Too little bias (b) Suitable bias (c) Too much bias

Figure 3.7: The bias problem causes incorrect self-shadowing (a) when the bias is to small. (c) shows an example of a bias parameter chosen to high which *pushes* the shadow away, and (b) depicts the result with an appropriate bias.

regions up the full image resolution which is naturally provided by mip-mapping. It is exactly this problem that our framework solves to allow constant time pre-filtering for Shadow Mapping.

Beyond aliasing problems Shadow Maps are also plagued by self-shadowing artifacts. As a result of numerical imprecision during the rasterization and subsequent coordinate frame transformations, $s(\mathbf{x})$ can not always be evaluated accurately and incorrect self-shadowing appears, as demonstrated in Figure 3.7 (a). The solution to this problem is to add a *depth bias* while the Shadow Map is being generated effectively pushing the surfaces a little bit away from the light source.

In most graphics API's such as OpenGL this is achieved with a polygon offset mechanism. The challenge is to find the right bias because the offset depends on the depth slope of polygons and a constant bias can be problematic as it might push the surface too far away to eliminate all self-shadowing, as shown in Figure 3.7 (c) whereas the rendering in Figure 3.7 (b) uses an appropriate bias parameter.

While the latter problem is less difficult to solve, Hourcade & Nicolas [1985]; Wang & Molnar [1994] present methods to avoid incorrect self-shadowing, discretization artifacts and filtered shadow reconstruction are much harder problems. In this chapter we will give an overview of the most recent Shadow Map variants. As mentioned before Shadow Map filtering is described separately in the related work Chapter 4 where we explain in detail why straight forward Shadow Map filtering is not possible.

We will now present the related work that tackles the discretization artifact problem and focus on previous efforts to Shadow Map filtering in the next chapter.

Shadow Map Parameterization. One way to alleviate the discretization problem is to re-parameterize the Shadow Map to extend the effective Shadow Map resolution. The following presents a review on such techniques.

Adaptive Shadow Mapping [Fernando et al., 2001] is based on the fact that a large overall shadow map resolution is not necessary. Only along shadow discontinuities a higher resolution is needed. Adaptive Shadow Maps therefore strives to eliminate the mismatch between camera and Shadow Map pixels by hierarchically refining shadow borders. The original method was implemented in software which was later ported to the GPU by Lefohn et al. [2005].

Perspective Shadow Mapping [Stamminger & Drettakis, 2002] and its descendants [Wimmer et al., 2004; Lloyd, 2007] compute the Shadow Map in normalized post-perspective space which decreases perspective aliasing as it yields a better sampling distribution with respect to the vantage point. The mapping from world-space to the post-perspective coordinate frame requires special care depending on the light source type.

Shadow Silhouette Maps [Sen et al., 2003] embed silhouette information into Shadow Maps for rendering perfectly hard shadows, but cannot deal with every possible configuration of shadow boundaries.

Practical Shadow Mapping [Brabec et al., 2003] first projects a pattern from the eye-camera onto the scene to then analyze from the light source view what region of the shadow map is actually relevant for shadow computation. A minimum enclosing rectangle is fitted to enclose the projected pattern in shadow map space to rotate and up-scale this rectangle to the original shadow map size. This paper also proposes to use linearly distributed depth values instead of the regular hyperbolic distribution. We will used linear z-values in all our implementations.

Alias Free Shadow Maps [Aila & Laine, 2004] and the *Irregular Z-Buffer*

[Johnson et al., 2004] were both presented in 2004 and showed that it is possible to find an optimal sampling distribution for a Shadow Map when using an irregular sampling structure. While the work of [Aila & Laine, 2004] additionally supports transparent objects, both methods use an irregular sampling strategy, but a direct implementation on graphics hardware was not possible. Therefore, Aila & Laine [2004] utilized a software rasterizer. In 2007, Arvo [2007] amended the performance of Alias Free Shadow Maps by implementing a layer-based variant on graphics hardware. A similar performance gain was achieved for the Irregular Z-Buffer by Johnson et al. [2005].

Soft Shadows using Shadow Maps. Accurate real-time display of soft shadows due to extended light sources, is a topic of ongoing research and we will present our contribution to this area in Part III. Instead of physically-based computation Shadow Maps can be used as a sampled scene representation to render inaccurate but visually plausible soft shadows in order to lessen computational effort as a viable alternative [Brotman & Badler, 1984].

Early work on Shadow Mapping extensions to render soft shadows borrow ideas from image-based rendering to efficiently average hard shadows. Chen & Williams [1993] use a view interpolation algorithm to portrait 3D scenes. The same algorithm can be naturally used to generate soft shadows from area light sources. The main idea is to render a few key Shadow Maps first. New Shadow Maps can then be efficiently interpolated from near-by key Shadow Maps.

Similar in spirit, Agrawala et al. [2000] merge Shadow Maps rendered from different positions on an area light source to pre-compute *Layered Attenuation Maps*. Due to fixed sampling locations during pre-computation this algorithm can lead to banding artifacts. They use Layered Attenuation Maps as a quick preview tool and use a coherence based ray tracer on the same data structure to render higher quality results (see Section 3.3).

Chan & Durand [2003], and Wyman & Hansen [2003] create plausible penumbrae in such a way that they extend the occluder object's silhouettes by a virtual geometric hull. The object itself is considered opaque whereas the geometric extension around the object represents a smooth decay from fully opaque to fully lit to mimic a penumbra region. Though these methods render plausible soft shadows, they overestimate umbra size and also require costly silhouette information.

Brabec & Seidel [2002] follow a similar avenue and attenuate light rays near blockers to reproduce the outward decay in visibility with respect to the umbra region, but rely on a costly neighborhood search in the depth map. These heuristics may produce results that deviate significantly from the actual physically-based solution.

In more recent work Atty et al. [2006] and Guennebaud et al. [2006] have transferred ideas from classical discontinuity meshing [Stewart & Ghali, 1994;

Drettakis & Fiume, 1994] to the Shadow Mapping domain. Such techniques compute a shadow value as the fraction of coverage of blocker geometry projected back onto the area light. To maintain high performance, the Shadow Map is used as a piecewise constant approximation of the blocker geometry which may yield either incorrect occluder fusion or light leaking. The work by Guennebaud et al. [2007] and bitmask soft shadows by Schwarz & Stamminger [2007] remove some of these problems, but increase the algorithmic complexity or computation time.

Coherent Shadow Maps [Ritschel et al., 2007] use to pre-compute visibility events from many Shadow Maps placed around the scene and present a loss-less compression scheme to prevent a large memory footprint. A GPU based Monte Carlo ray tracer then uses the Coherent Shadow Map data structure for efficient visibility queries. Their system enables interactive illumination for dynamic rigid objects including plausible all-frequency shadows, spatially varying BRDFs, and environment maps.

Coherent Surface Shadow Maps [Ritschel et al., 2008a] are an extension of Coherent Shadow Maps and render the Shadow Maps from objects surfaces to support indirect light transfer which is required for global illumination.

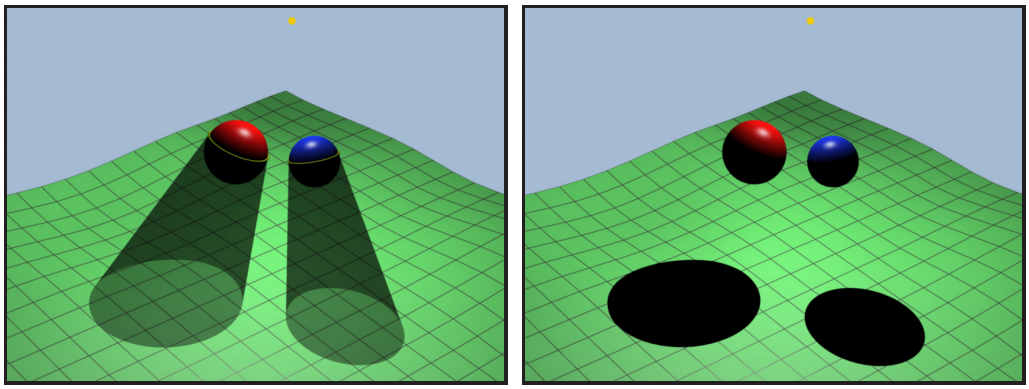
Imperfect Shadow Maps [Ritschel et al., 2008b] are low resolution Shadow Maps which exploit the fact that indirect light transfer is in general smooth. A point based scene approximation is rasterized into the low resolution Shadow Maps and is subsequently used for indirect lighting computations to decrease the rendering cost.

3.3 Object-Space Methods

The next class of shadow algorithms are solutions that operate in object-space rather than on sampled scene representations. These methods are known to produce superior shadow results on one hand but impose a higher computational cost compared to image-space algorithms on the other hand. Often such techniques require well defined input geometry and need to exercise special care or even break when objects deform or change their topology

3.4 Painter's Algorithm

One of the simplest algorithms for hidden surface removal is the *Painter's* method [Newell et al., 1972]. It owes its name to the way a painter might draw the part of the scene at the furthest distance first and then adding objects closer to the observer. The algorithm is an object-space sorting where each polygon is checked again the others and therefore has a run-time complexity of $O(n^2)$. The basic



(a) Shadow Volumes and silhouettes (b) Accurate and crisp shadows

Figure 3.8: Shadow Volumes. (a) silhouettes are constructed on a per-object basis first. Then finite volumes are constructed and subsequently rendered to updated the stencil buffer. The stencil buffer is then used as a shadow mask in screen-space (b) to render crisp and accurate shadows.

method involves the following steps:

- Sort all polygons according to their z-value (e.g. maximum depth).
- Check if polygons overlap. If their z-range overlaps too, split polygons.
- Draw all polygons from back-to-front.

An advantage of this sorting scheme compared to the Z-Buffer is that it can handle transparent surfaces which requires a back-to-front rendering to correctly accumulate color contributions.

3.4.1 Shadow Volumes

In 1977 Crow [1977] published *Shadow Volumes* which harnesses object silhouettes to produce shadows from point lights. The idea is to compute an object's silhouette with respect to the current light position. Once the silhouettes are available, semi-finite volumes are constructed as shown in Figure 3.8.

Each of the resulting finite volumes partitions the scene into regions lit and unlit volumes. In practice Shadow Volumes are mostly used in combination with a Z-Buffer. All volume faces attached to an object are rasterized front-to-back. Each times a front-facing pixel is generated the stencil buffer of that pixel is incremented. Similarly, when a back-facing pixel is rasterized the pixel's stencil value is decremented. Eventually, after all volume faces have been rendered the

stencil buffer acts as a shadow mask where value 0 refers to a lit pixel, and ≥ 1 means that the pixel is in at least one shadow volume and therefore occluded.

Shadow Volumes are known to render accurate shadows up to the underlying geometric fidelity because they operate in object-space. However, this is also a major source for its performance bottlenecks. Shadow Volumes strongly depend on the geometric complexity, are inherently fillrate-limited due to the rasterization overhead, and suffer from robustness problems [Everitt & Kilgard, 2002].

Assarson & Akenine-Möller [2002, 2003] enhance the original algorithm to support soft shadows. They introduce a new primitive called a *penumbra wedge*. Penumbra Wedges from region of penumbra and are used to compute a visibility buffer to enable soft shadows based on the occlusion within a Penumbra Wedge. Though using frequent frame buffer accesses their work has pushed soft shadow rendering for limited complexity scenes toward real-time performance [Assarson et al., 2003].

Aila & Akenine-Möller [2004] developed a tile-based hierarchical Shadow Volume algorithm to reduce the fillrate drastically. Based on the observation that a shadow boundary can only appear inside a screen-space tile if at least one shadow volume triangle intersects that tile. Therefore, tiles with no intersections are masked as non-boundary tiles and can rapidly determine if all pixels within that tile are in shadow or lit.

Please consult Heidmann [1991]; Everitt & Kilgard [2002]; McGuire et al. [2003]; Assarson et al. [2003] for more information on implementations of Shadow Volumes which utilize graphics hardware or Brabec & Seidel [2003] for a solution completely realized on a GPU.

3.4.2 Shadow Rays

Ray Tracing [Whitted, 1979] shoots a ray through each camera pixel into the virtual environment and intersects each ray with a hierarchical spatial acceleration structure containing all scene polygons. Each time a surface is hit, a shader is executed to compute the pixel color. While original Ray Tracing only shoots a single ray to the light, Cook et al. [1984] show that it can easily be extended to distribute many samples, e.g. over the light sources to determine the current occlusion magnitude to render accurate umbra and penumbra. Depending on the amount of sampling and time spent during image generation Ray Tracing based rendering systems produce stunning results and reach photo-realistic quality.

Parker et al. [1998] demonstrate a simple extension to Ray Tracing where they alter the geometry (they use the term *soft-edged object*) similar to Brabec & Seidel [2002] who use a sampled representation to virtually extend the object to render believable soft shadows by just using a single ray sample. Special treatment

becomes necessary when the distance between two objects becomes too small in which case light leaking would appear.

Despite promising advances in recent years (see the work of Wald [2004] and references therein), whether utilizing the GPU [Timothy J. Purcell & Hanrahan, 2002], building special purpose hardware [Schmittler et al., 2002], or distributing the workload [Wald et al., 2003] to PC-Clusters, Ray Tracing unfortunately is still too costly for real-time rendering. Especially dynamic scenes pose a great challenge (see Günther et al. [2006]; Wald et al. [2007]) and necessitate updates of the spatial acceleration structure.

3.5 Hybrid Methods

Researcher frequently combine the advantages of two algorithms to yield improved solutions. In the following we review some interesting combinations of Shadow Maps, Shadow Volumes, and Ray Tracing.

McCool [2000] combines Shadow Maps and Shadow Volumes to leverage the performance of a sampling-based method and the accuracy of an object-space technique. His algorithm starts with rendering a Shadow Map to get a discrete scene description. A special edge-detection and reconstruction process determines silhouette information from the depth map and constructs Shadow Volumes. He further shows that a single bit stencil buffer is sufficient to distinguish lit and shadowed pixels.

Chan & Durand [2004] lessen the fillrate related burden of traditional Shadow Volume rendering in such a way that they use Sen's method [Sen et al., 2003] on a low resolution Shadow Map to detect shadow discontinuities. Shadow Volume pixels are only processed if confirmed by a screen-space mask reducing the fillrate significantly.

Ray Tracing Multi-Layer Shadow Maps [Keating & Max, 1999; Im & Han, 2005] instead of a full polygonal model has proven to be a profitable solution and yields very good results depending on the Shadow Map resolution and the budget spent for ray sampling multiple depth layers to reconstruct the visibility function.

Agrawala et al. [2000] present a *Coherence-Based* Ray Tracing for depth images. This allows for more flexible sampling and increases the quality while keeping the run-time cost cheaper than regular Ray Tracing.

Supporting multiple layers has an important advantage compared to the single layer philosophy. Such methods can handle transparencies and therefore render more complex materials and also represent high frequency details such as hair or fur as coverage. Xie et al. [2007] demonstrate that Multi Layer Depth Maps achieve film production quality and are frequently used on current shows.

As a last hybrid method we would like to mention *Soft Shadow Volumes for Ray-Tracing*, a combination proposed by Laine et al. [2005]. The authors present a highly efficient hierarchical process to compute silhouettes that overlap an area light source from the view of a shading point. In contrast to previous techniques [Nishita & Nakamae, 1983; Takahashi & Tanaka, 1997], Laine et al. [2005] use silhouette edges to integrate the depth complexity. Their results are one to two orders of magnitude faster compared to tracing shadows ray for similar image quality.

3.6 Pre-computation Methods

Other methods based on Precomputed Radiance Transfer Sloan et al. [2002] calculate and store an illumination-invariant light transport solution off-line and use it for real-time relighting. The scene is assumed to be static and storage demands aggressive compression which may introduce artifacts such as blurring. Even though these limitations have been alleviated [Ng et al., 2003; Zhou et al., 2005; Sloan et al., 2005], it remains challenging to support fully dynamic scenes with arbitrary illumination.

Shadow computation for dynamic scenes can be accelerated by simplifying the geometry. Ren et al. [2006] approximate dynamic objects using a sphere hierarchy, whereas Kautz et al. [2004] use a two-level mesh hierarchy. These methods only support model deformation, and assume that object topology remains static.

We would like to conclude this chapter with a new method that is conceptually different from the ones we have mentioned so far; *Implicit Visibility*. Dachsbacher et al. [2007] show how to reformulate the rendering equation in order to transform explicit visibility sampling into local iteration using a new quantity called *Antiradiance*.

Chapter 4

Related Work on Shadow Map Filtering

In the previous chapter, Shadow Mapping was reviewed with regard to its discretization problems. The focus of this chapter lies on methods providing Shadow Map filtering which are closely related to our work. We distinguish two objectives. Efficient anti-aliasing of shadow discontinuities and approximate soft shadow rendering both through spatial convolutions of Shadow Map (or data structures derived from depth maps).

4.1 Anti-aliasing

After its introduction, a lot of effort has been made to tackle the aliasing problem inherent in Shadow Mapping, and efficient filtering techniques similar to texture filtering [Heckbert, 1989] have been investigated.

Reeves et al. [1987] observed that shadows should be anti-aliased by filtering Shadow Map pixels *after* after the depth test. This led to the *Percentage Closer Filtering* (PCF). PCF determines the coverage of a camera pixel in light space and applies the shadow test to a number of samples distributed over this region. The outcome of the shadow test is then averaged into a filtered results. Unfortunately, the shadow test depends on the distance from the light to the point to shade. Therefore, efficient filtering as for regular textures which relies on *pre-filtering* the image a priori is not possible. PCF became available on graphics hardware, albeit with limited quality (bilinear filtering only). One can increase the quality of PCF by taking into account more samples, (e.g., in a hardware shader) but this reduces performance dramatically.

Deep Shadow Mapping by Lokovic & Veach [2000] pre-computes the aggregate result of binary shadow tests within each texel for excessively complex scenes like hair, yielding a continuous visibility function for each texel, which can be queried at render-time. It assumes illumination and geometry to be static. This restriction was later lessened by exploiting graphics hardware [Kim & Neumann, 2001; Mertens et al., 2004]. Our technique bears some similarity to Deep Shadow Maps, since we also store a visibility function. However, we are only interested in using binary visibility functions, and applying spatial convolution instead of intra-pixel averaging.

In a recent effort, Donnelly & Lauritzen [2006] introduced *Variance Shadow Maps* a probabilistic approach for rendering filtered shadows that supports pre-filtering, and additional convolutions. The result can therefore be computed in constant time by using mip-mapping or summed area tables (SAT) [Crow, 1984]. When the Shadow Map is rasterized, the z and z^2-values are stored and used in the Chebyshev Inequality during rendering to estimate the probability whether a point is in shadow or not. Their estimate only gives an upper bound of the result and produces noticeable high-frequency light leaking artifacts for scenes with a high depth complexity.

A variant of VSMs using summed-area tables has been published by Lauritzen [2007a], which reduces light leaking. However, the authors show that it cannot be removed completely. Our method from Chapter 6 requires less stringent assumptions, and even though it is also approximate, it converges to the exact solution instead of an upper bound.

In concurrent work Salvi [2008] derives the same exponential formula we will present in Chapter 7. While his work originates from the Marcov Inequality our approach stems from geometric considerations. In addition to Salvi [2008] we present a failure case analysis and offer a fall-back filtering solution to reduce noticeable artifacts.

4.2 Soft Shadows

Soler & Sillion [1998] propose an image-based shadow algorithm based on convolution. Convolutions can be computed efficiently, even for large penumbrae. Soler and Sillion do not employ a depth buffer and therefore require an explicit notion of blockers and receivers, and cannot directly support self-shadowing. We apply a similar convolution in the context of shadow mapping, which naturally allows for self-shadowing.

Fernando [2005] introduced *Percentage Closer Soft Shadows* to render plausible penumbra from Shadow Maps using two-step filtering approach. First the

virtual area light source is used to compute an appropriate Shadow Map filter size. For instance for a rectangular light, the pyramid formed by the four light source corners and the shading point can be intersected with the Shadow Map to determine a spatial filter size. Fernando [2005] use 32 samples to compute the average depth of pixels blocking the current shading point. The average blocker z-value is used in combination with the triangle equality to estimate a new filter width proportional to the penumbra size to filter the Shadow Map again by distributing 64 samples into the new filter region. Although this technique can be implemented using adaptive sampling according to the filter kernel is still requires many samples to yield good results. We will compare the quality and sampling bandwidth of this approach against our filtering strategy and demonstrate that we can not only significantly reduce the sampling but even provide *constant-time* shadow filtering.

A recent version of Variance Shadow Maps [Lauritzen, 2007b] simulates penumbrae more accurately by varying the kernel size based on the average blocker depth, similar to Fernando Fernando [2005]. Unfortunately, the cost of computing this average defeats the purpose of constant cost convolution, as it requires brute-force sampling of the Shadow Map. An important advantage of our soft shadow approach is that this step can be carried out in constant time as well.

Part I

Linearization

Chapter 5
Shadow Test Linearization

In the previous chapter we have seen that discretization artifacts due to insufficient Shadow Map resolution are a major problem of Shadow Mapping. They degrade the image quality, cause aliasing, and temporal incoherence (Figure 3.5). Compared to the vast amount of papers dedicated to increase the effective Shadow Map resolution relatively few articles deal with Shadow Map filtering to provide effective screen-space anti-aliasing.

It appears tempting to apply the same filtering operations, e.g. mip-mapping, as commonly used for texture filtering to a depth map, and then expect the same outcome. Indeed it would be beneficial if Shadow Maps could be treaded in the exact same way as regular texture images can be. Not only would this speed-up rendering time but it would also allow to take advantage of high-quality texture filtering which is commonplace on todays graphics chips.

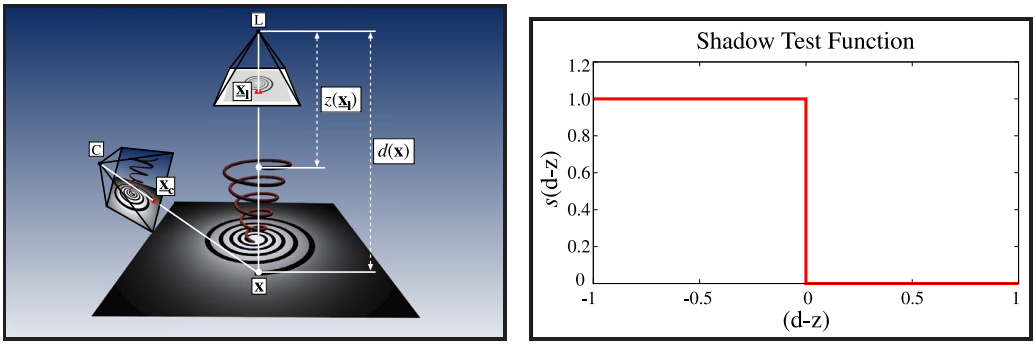
Unfortunately, as we have already seen in Section 3.2.2, simply filtering the depth values does not produce the desired result. What happens in this situation is that geometric details are being filtered and after the spatial convolution is complete, a binary shadow test is applied. This only widens the shadow boundary but does not remove or conceal jagged edges.

In the following we will explain and discuss this problem in more detail and propose a new framework to solve this problem and to enable efficient filtering for Shadow Maps.

5.1 Shadow Test Function

We would like to first review the shadow test function $s(\mathbf{x})$ from Equation 3.2

$$s(\mathbf{x}) := f(d(\mathbf{x}), z(\underline{\mathbf{x}}_\mathbf{l})) = \begin{cases} 1 & \text{if } d(\mathbf{x}) \leq z(\underline{\mathbf{x}}_\mathbf{l}) \\ 0 & \text{if } d(\mathbf{x}) > z(\underline{\mathbf{x}}_\mathbf{l}), \end{cases}$$



(a) Shadow Mapping (b) Heaviside Step Function

Figure 5.1: Shadow Mapping is shown in (a). If we were to slide a second point $\mathbf{x}'$ along the ray from the light to $\mathbf{x}$ we would receive a step function shown in (b).

which we introduced in the Background chapter on Shadow Maps 3.2.2. We saw that $s(\mathbf{x})$ is a piecewise constant function and we would like to illustrate what this function represents geometrically as we will often refer to it and to easy the understanding for the following chapters. For this purpose let us *slide* a second point $\mathbf{x}'$ along the ray from L into $\mathbf{x}$ and plot the result of the shadow test for $\mathbf{x}'$ as shown in Figure 5.1 (a) and (b). $\mathbf{x}'$ remains lit $= 1$, as shown in Figure 5.1 (b) until it reaches a depth larger than $z(\underline{\mathbf{x}}_l)$. In this case it becomes shadowed $= 0$. As a result for $s(\mathbf{x})$ we receive the aforementioned piecewise constant function. As one can see this function resembles the Heaviside Step function:

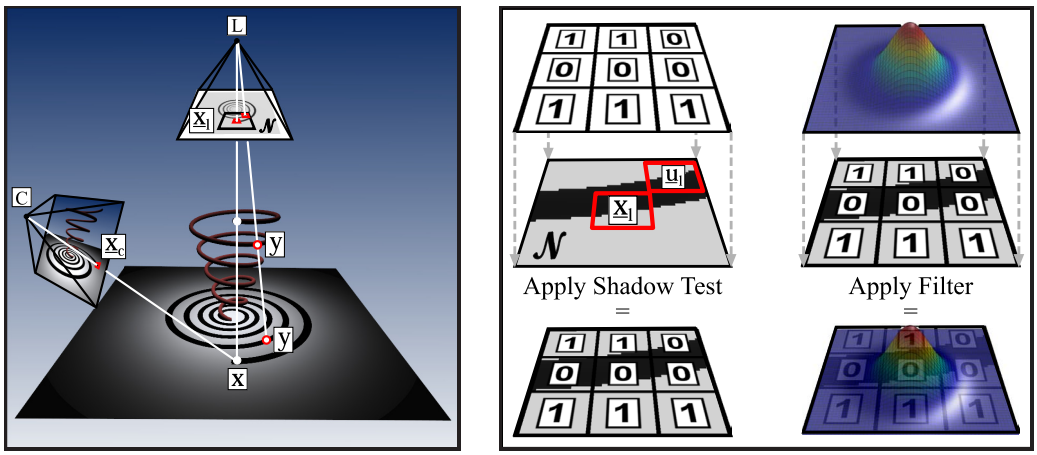
$$H(t) := \begin{cases} 0 & \text{if } t < 0 \\ \frac{1}{2} & \text{if } t = 0 \\ 1 & \text{if } t > 0, \end{cases} \tag{5.1}$$

except that we have $1 - H(t)$. This function plays an integral part in solving the filtering problem as we would like to convolve its results with a filter kernel. Let us now investigate what steps need to be taken to achieve this goal.

5.2 Convolution

In order to have anti-aliased shadows, we need to filter $s(\mathbf{x})$ e.g., using a low pass filter. Generally speaking, a convolution (or linear filtering) operation on a function g with kernel w supported over a neighborhood $\mathcal{N}$, is defined as:

$$\left[w * g\right](\underline{\mathbf{x}}_l) := \sum_{\underline{\mathbf{u}}_l \in \mathcal{N}} w(\underline{\mathbf{u}}_l) g(\underline{\mathbf{x}}_l - \underline{\mathbf{u}}_l). \tag{5.2}$$



(a) Filtering Shadow Maps (b) Order of Shadow Test and Convolution

Figure 5.2: Shadow Map filtering. (a) shows why we need to assume $d(\mathbf{y}) \approx d(\mathbf{x})$. (b) visualizes the correct order of shadow test and subsequent filtering. Note that the convolution happens over variable $\underline{\mathbf{u}}_\mathbf{l}$ in the neighborhood $\mathcal{N}$.

Figure 5.2 (a) shows the filter region $\mathcal{N}$ and (b) illustrates the filtering process we seek to achieve. Let us now try to convolve $s(\mathbf{x})$, and denote the result as $s_f(\mathbf{x})$:

$$s_f(\mathbf{x}) = \sum_{\underline{\mathbf{u}}_\mathbf{l} \in \mathcal{N}} w(\underline{\mathbf{u}}_\mathbf{l})\, f\big(d(\mathbf{y}), z(\underline{\mathbf{x}}_\mathbf{l} - \underline{\mathbf{u}}_\mathbf{l})\big). \tag{5.3}$$

Even though s_f is formulated in terms of $\mathbf{x}$, the actual convolution happens in Shadow Map space, i.e. over variable $\underline{\mathbf{u}}_\mathbf{l}$. Note that Equation 5.3 contains a new variable $\mathbf{y}$, which is informally defined as the point that lies near $\mathbf{x}$, such that $T(\mathbf{y}) = \underline{\mathbf{x}}_\mathbf{l} - \underline{\mathbf{u}}_\mathbf{l}$. Unfortunately, there is no unique $\mathbf{y} = T^{-1}(\underline{\mathbf{x}}_\mathbf{l} - \underline{\mathbf{u}}_\mathbf{l})$, because T is not invertible, see Figure 5.2 (a). In order to arrive at a mathematically sound formulation of Shadow Map convolution, we need to assume that $d(\mathbf{y}) \approx d(\mathbf{x})$, so that we can write:

$$\begin{aligned} s_f(\mathbf{x}) &= \sum_{\underline{\mathbf{u}}_\mathbf{l} \in \mathcal{N}} w(\underline{\mathbf{u}}_\mathbf{l})\, f\big(d(\mathbf{x}), z(\underline{\mathbf{x}}_\mathbf{l} - \underline{\mathbf{u}}_\mathbf{l})\big) \\ &= \big[w * f\big(d(\mathbf{x}), z\big)\big](\underline{\mathbf{x}}_\mathbf{l}) \end{aligned} \tag{5.4}$$

This assumption $d(\mathbf{y}) \approx d(\mathbf{x})$ basically states that $d(\mathbf{x})$ is a representative distance for the neighborhood $\mathcal{N}$, which is only correct for a planar receiver, parallel to the Shadow Map's image plane. Note that a similar approximation is made for PCF [Reeves et al., 1987], and in Soler et al.'s work [Soler & Sillion, 1998].

(a) Regular Shadow Mapping (b) Incorrect Filtering (c) Correct Filtering

Figure 5.3: Non-linearity of $s(\mathbf{x})$. Close-ups from the helix scene (a–c). Shadow Mapping (a), filtering the z-values (b) which gives incorrect and widened shadow boundaries, and correctly filtered shadows (c).

It is important to see that we cannot directly apply a convolution to $z(\underline{\mathbf{x}}_\mathbf{l})$, because f is *non-linear* with respect to its arguments. In other words:

$$\big[w * f\big(d(\mathbf{x}), z\big)\big](\underline{\mathbf{x}}_\mathbf{l}) \neq f\big(d(\mathbf{x}), [w * z](\underline{\mathbf{x}}_\mathbf{l})\big). \tag{5.5}$$

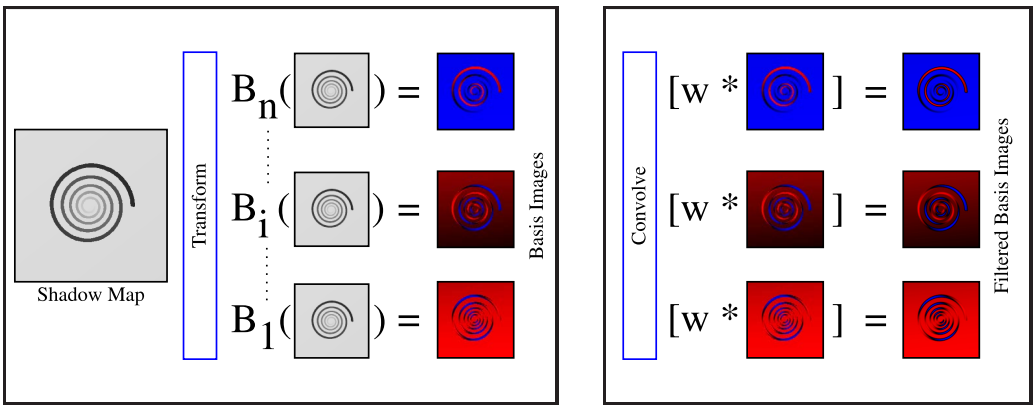
This explains why regular texture filtering cannot be applied to $z(\underline{\mathbf{x}}_\mathbf{l})$ (i.e., the Shadow Map): filtering $z(\underline{\mathbf{x}}_\mathbf{l})$ values is **not** equivalent to filtering the result of the shadow test. We show an example in Figure 5.3 where we illustrate this inequality and show the expected result.

Although it is possible to carry out the summation in Equation 5.5 directly at run-time Reeves et al. [1987], our goal is to apply pre-filtering. In other words, to apply a filter *before* the shadow test is actually used. This would enable efficient separable filtering, and more importantly, to employ mip-mapping.

To achieve this, we transform the z-values such that the shadow test can be written as a sum. This will allow us to *linearize* the depth test. Let us therefore expand $f(d,z)$ as follows:

$$f(d,z) = \sum_{i=1}^{\infty} a_i(d) B_i(z) \tag{5.6}$$

Here, B_i are basis functions in terms of z, which we will concretely define in Section 6.1 and 7.1. Each basis is weighted by corresponding coefficients a_i depending on d. The expansion has to be truncated in practice to some truncation order N. We see that the expansion does not yield a direct linear dependence on z,



(a) Transform Shadow Map (b) Convolve Basis Images

Figure 5.4: Conversion of a traditional Shadow Map into a set of basis images (a). Applying a convolution to the individual basis images is essentially the same as filtering the shadow test function $s(\mathbf{x})$.

but it is linear with respect to the basis set $B_{i=1\ldots N}$. In order to apply this expansion in practice, we convert the Shadow Map to *basis images* by applying each basis function to the Shadow Map: $B_i\big(z(\underline{\mathbf{x}_\mathbf{l}})\big)$. This process is shown in Figure 5.4 (a). Consequently, the shadow function in Equation 3.2 can be translated to linear combination of these basis images:

$$s(\mathbf{x}) \approx \sum_{i=1}^{N} a_i\big(d(\mathbf{x})\big) B_i\big(z(\underline{\mathbf{x}_\mathbf{l}})\big) \tag{5.7}$$

To see why this is useful, we fill in the expansion from Equation 5.7 in the convolution in Equation 5.4:

$$\begin{aligned} s_f(\mathbf{x}) &\approx \big[w * f\big(d(\mathbf{x}), z\big)\big](\underline{\mathbf{x}_\mathbf{l}}) \\ &\approx \Big[w * \sum_{i=1}^{N} a_i\big(d(\mathbf{x})\big) B_i(z)\Big](\underline{\mathbf{x}_\mathbf{l}}) \\ &\approx \sum_{i=1}^{N} a_i\big(d(\mathbf{x})\big)\big[w * B_i(z)\big](\underline{\mathbf{x}_\mathbf{l}}) \end{aligned} \tag{5.8}$$

The last equation is the key observation in this dissertation:

Any convolution operation on the shadow function is equivalent to convolving the individual basis images $B_i\big(z(\underline{\mathbf{x}_\mathbf{l}})\big)$.

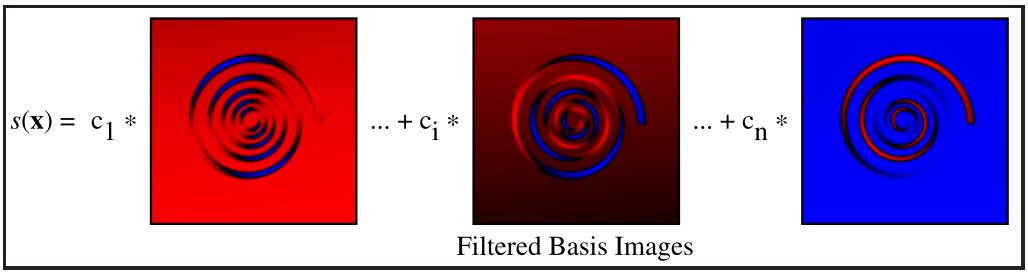


Figure 5.5: Reconstructing a filtered shadow value for $\mathbf{x}$ requires to evaluate the linear combination from Equation 5.8. This is achieved by summing up the weighted and pre-filtered basis images.

(a) Shadow Mapping (b) Basis Image Reconstruction

Figure 5.6: The difference between Shadow Mapping and our new basis image reconstruction theory is compared. Filtering is possible for basis images, as opposed to regular depth maps and therefore allows pre-filtering.

It is important to see that in order to reach Equation 5.8, each term in the expansion (Equation 5.6) had to be separable with respect to variables d and z. Decoupling $d(\mathbf{x})$ from $z(\underline{\mathbf{x}_l})$ is important, because it enables us to convolve the images $B_i\big(z(\underline{\mathbf{x}_l})\big)$ *before* the shadow test.

Once we have transformed a Shadow Map into this new basis image representation (see Figure 5.4 (a)), we can reconstruct the a shadow from the filtered basis images by simply evaluating Equation 5.8 (see Figure 5.4 (b)). Comparing the result to the original Shadow Mapping algorithm in Figure 5.6 illustrates the usefulness and quality improvement as shadow boundaries are nicely anti-aliased. The image from Figure 5.3 (c) has also been rendered with our technique.

The success of our technique will obviously depends on the chosen series expansion. We will now continue to Part II of this dissertation where we present two expansions: a Fourier and an Exponential series.

Part II

Anti-aliasing of Shadows

Chapter 6

Convolution Shadow Maps

In this chapter we present *Convolution Shadow Maps* (CSMs) [Annen et al., 2007], our first solution to the proposed expansion from the previous chapter by approximating the shadow test by a Fourier series expansion. Depending on the truncation order, z-values are converted into several basis textures. In the final rendering, pre-filtered texture samples are fetched to reconstruct a smoother shadow.

We demonstrate the usefulness of this representation, and show that hardware-accelerated anti-aliasing techniques, such as tri-linear and anisotropic filtering, can be applied naturally to Convolution Shadow Maps. Our approach can be implemented very efficiently in current generation graphics hardware, and offers real-time frame rates.

Compared to Variance Shadow Maps [Donnelly & Lauritzen, 2006], our approach is unbiased and can deal with arbitrary depth complexity, and even though it is also approximate, it converges to the exact solution instead of an upper bound.

6.1 Fourier Series Expansion

In this section we show how the Fourier series can be utilized as a solution for to shadow test linearization. For clarity, we note that the Fourier expansion will not be used for applying the convolution theorem to perform spatial filtering; convolution of the basis images $B_i\big(z(\underline{\mathbf{x}_\mathbf{l}})\big)$ will be done explicitly.

We strive to expand the shadowing function f according to Equation 5.6 using a Fourier series. In general, we can decompose any periodic function $g(t)$ as an infinite sum of waves:

$$g(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} \Big[a_n \cos(\frac{2}{T}nt) + b_n \sin(\frac{2}{T}nt)\Big], \tag{6.1}$$

where the coefficients a_n and b_n are obtained by integrating the cosine and sine basis functions against g, respectively. This is the standard Fourier series and will be used to represent the shadowing function.

f is a function in terms of two variables, but it can be expressed as the Heaviside step (or the "unit step") function $H(t)$, which we saw in the last chapter, as follows: $f(d,z) = H(d-z)$. Let us first focus on expanding $H(t)$. We represent it using a square wave function, in order to make it periodic (a requirement to apply a Fourier series approximation). Let $S(t)$ be a square wave function with period 2. For $t \in (-1,1)$ we have $H(t) = \frac{1}{2} + \frac{1}{2}S(t)$. For this particular case of S(t), the (truncated) Fourier series expansion yields:

$$S(t) \approx \frac{4}{} \sum_{k=1}^{M} \frac{1}{2k-1} \sin\left[(2k-1) \quad t\right] \tag{6.2}$$

Now, returning to f we have:

$$f(d,z) \approx \frac{1}{2} + 2 \sum_{k=1}^{M} \frac{1}{c_k} \sin\left[c_k(d-z))\right], \tag{6.3}$$

with $c_k = \quad (2k-1)$. We convert the previous summation into a form similar to Equation 5.6 using the trigonometric identity

$$\sin(a-b) = \sin(a)\cos(b) - \cos(a)\sin(b). \tag{6.4}$$

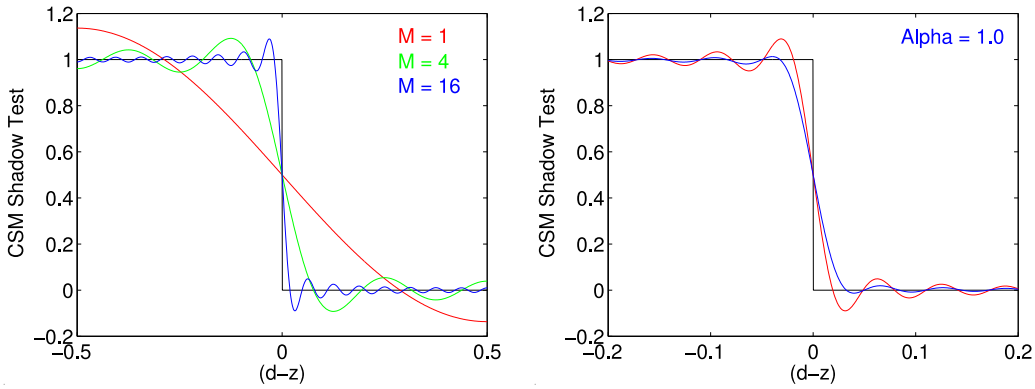
Note the we swap the sine and cosine terms in the above equation to flip f along the x-axis, as $(d-z) <= 0$ has to represent a lit surface, or in other words a value of 1. This is opposite to the original Heaviside function. We then get:

$$\begin{aligned} f(d,z) \approx \frac{1}{2} &+ 2 \sum_{k=1}^{M} \frac{1}{c_k} \cos(c_k d)\sin(c_k z) \\ &- 2 \sum_{k=1}^{M} \frac{1}{c_k} \sin(c_k d)\cos(c_k z) \end{aligned} \tag{6.5}$$

We see that Equation 6.5 complies with Equation 5.6, and we have separable terms with respect to d and z:

$$\begin{aligned} a_{(2k-1)}(d) &= \tfrac{2}{c_k}\cos(c_k d), \quad a_{(2k)}(d) = \tfrac{-2}{c_k}\sin(c_k d) \\ B_{(2k-1)}(z) &= \quad \sin(c_k z), \quad B_{(2k)}(z) = \quad \cos(c_k z) \end{aligned} \tag{6.6}$$

with $k = 1 \ldots M$ (note that $N = 2M$ in Equation 5.6) and we add the constant term $\frac{1}{2}$ separately.



(a) Reconstruction Order M (b) Ringing

Figure 6.1: Reconstruction and ringing. The x-axis encodes the difference $(d-z)$ along a shadow ray (lookup). (a) illustrates the conflict of increasing M to achieve a more reliable shadow test and introducing high frequencies noticeable as ringing artifacts. (b) shows the impact of attenuation to suppress ringing as the red turns into the blue signal.

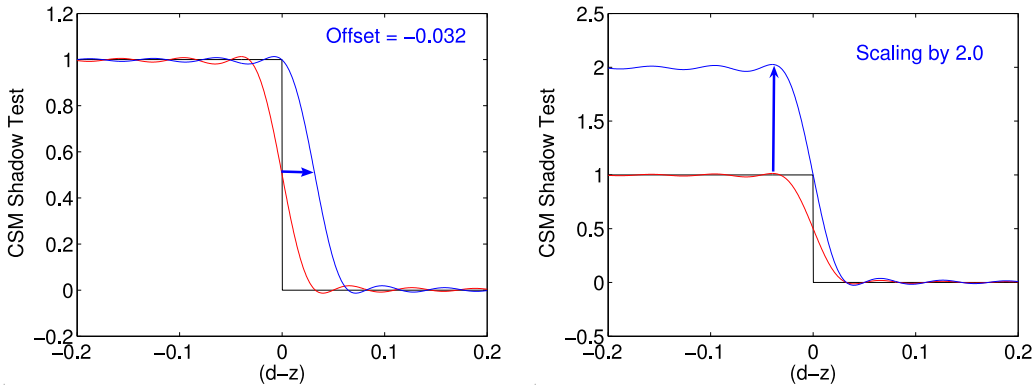
6.1.1 Discussion of Fourier Expansion

We opted for the Fourier expansion for two reasons. First, it is shift-invariant with respect to d and z, which is a general property of the Fourier transform (cf. rotational invariance of Spherical Harmonics Sloan et al. [2002]). Intuitively speaking, this enables us to "move" the Heaviside step around without any loss in precision. In fact, this can be done by independently changing d and z, while keeping the approximation error due to truncation constant. The second reason is that the basis functions (sine and cosine waves) are bounded: they always map to the interval $[-1,1]$. This affords a fixed point representation, which we can even quantize to 8 bits in practice.

The Fourier series does not come without disadvantages. First, as with any Fourier representation, it is prone to ringing (Gibbs phenomenon). Second, the Fourier expansion smoothens the step function, which can result in incorrect shadowing if not handled. We deal with both problems as shown the following subsections.

Ringing

A Fourier expansion potentially suffers from ringing, particularly when the expansion is truncated to a small number of terms M as illustrated in Figure 6.1 (a). We reduce this effect by attenuating each k-th term by $\exp\left(- \quad (\frac{k}{M})^2\right)$. Parameter controls the attenuation strength ($= 0$ leaves the series unchanged). The



(a) Offset (b) Scaling

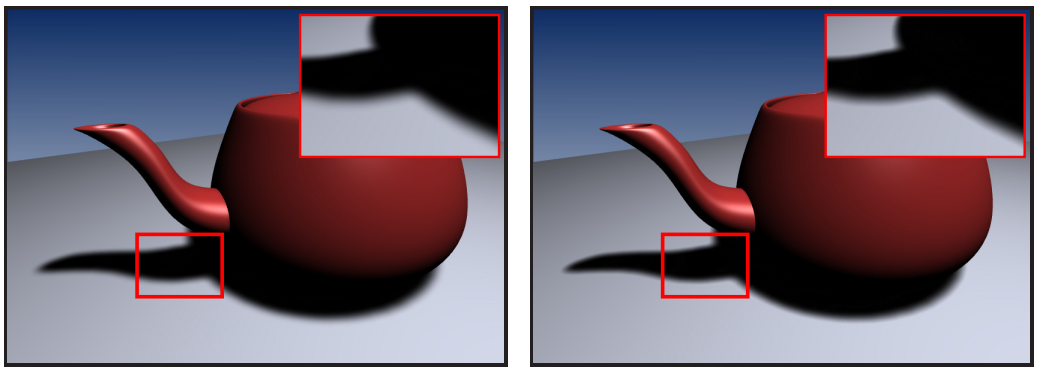
Figure 6.2: Two methods to enhance the shadow test. In (a) an offset is applied to d before reconstruction, which prevents incorrect darkening of lit areas. (b) shows how scaling makes the transition steeper and how it also prevents incorrect darkening. Please compare (a) and (b) with the results in Figure 6.3.

magnitude of the high frequencies is always reduced more, while the low frequencies remain almost the same. This incurs an important tradeoff: reducing ringing also means that the reconstructed Heaviside step becomes less steep as shown in Figure 6.1 (b).

Offsetting and Scaling

The Fourier expansion of the step function introduces a smooth transition, which is obvious with low-order expansions M, see Figure 6.1 (a). This means that for lit surfaces, where $(d-z) \approx 0$, the shadow function $f(d,z)$ evaluates to 0.5. This is undesirable, as all lit surfaces would be 50% shadowed. We can correct this, by offsetting the expansion of the Heaviside step, see Figure 6.2 (a). After offsetting, $f(d,z)$ goes through 1 for $(d-z) \approx 0$, which results in correctly lit surfaces. The shift-invariance property of the Fourier expansion allows us to formulate a constant offset, which only depends on the truncation order and can thus be applied at every pixel. Of course, offsetting makes the transition from unshadowed to shadowed more obvious near contact points.

Scaling the expansion by 2.0 makes the transition steeper and also ensures that all lit surfaces (around $d-z \approx 0$) are actually correctly lit, see Figure 6.2 (b). However, scaling sharpens shadows and can potentially reintroduce aliasing. The shadow value is always clamped to $[0,1]$. Figure 6.3 shows renderings with offsetting and scaling.



(a) Offsetting d (b) Scaling f

Figure 6.3: Difference of subtracting an offset from d or scaling f. (a) shows that subtracting an offset preserves convolution results but may exhibit reconstruction limitations near contact points (depending on M). Here we used a 9×9 Gauss filter and $M = 16$. (b) illustrates that scaling f sharpens the transition but also reduces filtering (shadows are sharper).

6.2 Anti-aliasing Using CSMs

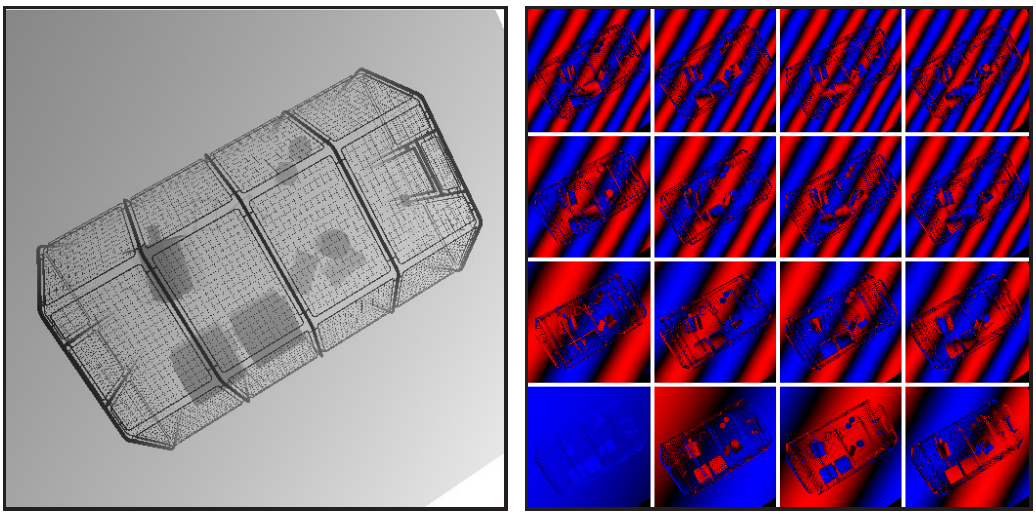
Aliasing from Shadow Map minification (multiple Shadow Map texels falling onto the same image pixel) as well as from Shadow Map discretization (jagged boundaries) are difficult problems, since pre-filtering techniques cannot be easily applied. However, Convolution Shadow Maps enable filtering with arbitrary convolution kernels, and therefore enable the use of pre-filtering techniques for anti-aliasing.

In particular, we perform mip-mapping as well as blurring of the Shadow Map, i.e. of the basis functions to be more precise, in order to remove aliasing artifacts from both minification as well as discretization.

6.2.1 GPU Implementation

Convolution Shadow Maps require only a few modifications to a standard Shadow Mapping pipeline. After rendering the depth values from the light's point of view, we evaluate the basis functions ($\sin(c_k z)$ and $\cos(c_k z)$, see Equation 6.6) using the current z-values at each texel and store the result, which correspond to the basis functions $B_i\big(z(\underline{\mathbf{x}}_l)\big)$ from Equation 5.7, in texture maps. Figure 6.4 shows the evaluated sine basis functions for a given depth map (blue positive, red negative). Note that we use linear depth values to increase the sampling precision Brabec et al. [2003].

Depending on the Fourier expansion order M and hardware capabilities, we

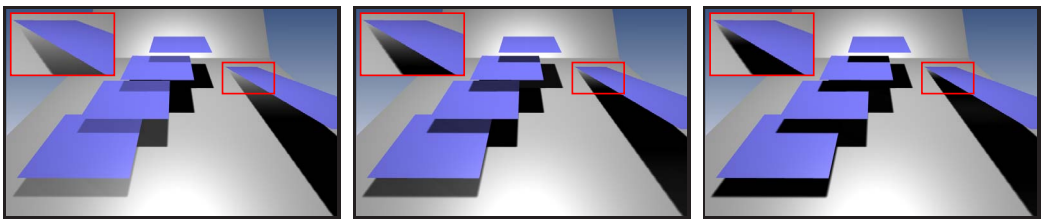


(a) Shadow Map (linear depth) (b) CSM basis textures

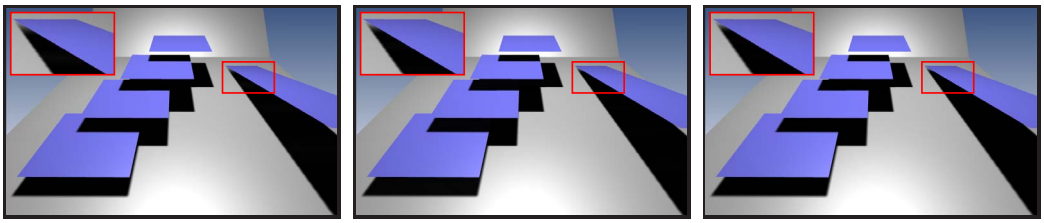
Figure 6.4: Visualization of a shadow map and its corresponding basis textures for $M = 16$ (RGBA channels are split in separate images for visualization purposes).

perform multiple rendering passes to convert a single shadow map into a set of sine and cosine textures. For example, with $M = 16$ we need to generate 16 sine and also 16 cosine terms which we will pack into four sine and four cosine 8-bit RGBA textures. 32-bit floating precision did not produced noticeable differences and we use 8-bits fixed point for all our renderings. With four Multiple Rendering Targets (MRTs) only two additional render passes are necessary. Each pass renders a screen-align quad and computes the sine and cosine terms based on the current shadow map respectively. Results are packed into four RGBA textures simultaneously. Once this set of basis textures has been computed, we can apply filtering to it. First, we apply a separable Gaussian filter kernel on the textures to hide aliasing from discretization. Of course for high-resolution Shadow Maps, this is not necessary. We then build a mip-map of this texture (using the auto-mip-map feature of modern GPUs) to prevent minification aliasing of shadows.

During the final rendering from the camera view we exchange regular shadow mapping (either binary or PCF) with our shadow reconstruction as described by Equation 6.5. I.e., we evaluate a weighted sum at each pixel of the filtered basis functions multiplied by coefficients $a_i(d)$ (defined in Equation 6.6), where d is the distance from the current pixel to the light source. The resulting value s_f (see Equation 5.8) is the filtered shadow value. Simply switching on mip-mapping or even anisotropic filtering removes screen-space aliasing; no shader magic is needed. Due to ringing, the resulting shadow value can be outside outside the $[0,1]$-range and we therefore clamp the result to lie within $[0,1]$.



(a) M=1 (b) M=2 (c) M=4



(d) M=8 (e) M=12 (f) M=16

Figure 6.5: Quality comparison for shadow test reconstruction using different number of Fourier series order M. All signals have been quantized to 8-bits per channel. When using a tightly fitted light frustum a single 8-bit RGBA texture usually faithfully reconstructs the shadow test. Note that ringing causes varying lightness in shadowed areas for small M.

6.3 Results

In this section we present results highlighting the potential of Convolution Shadow Maps. All figures have been rendered using OpenGL on a Dual-Core AMD Opteron with 2.6GHz and 2.75GB RAM equipped with an NVIDIA GeForce 8800 GTX graphics card. All results have been rendered using 8-bit precision per basis function and using offsetting as described earlier.

The amount of memory required by the CSM data structure only depends on the reconstruction order M. As we fix the precision to 8-bits per channel, we require $\frac{M}{2}$ 8-bit RGBA textures to store the basis functions. Compared to VSM, the CSM requires four times more memory for $M = 16$ than a VSM with 32-bit floating point precision. For scenes where $M = 4$ is sufficient, CSMs require the same amount of memory as 32-bit VSMs. In practice, this is a reasonable configuration, as we have seen in Figure 6.5 that this setting yields good results.

Table 6.1 contains performance measurements for various sizes, shadow map sizes, and different reconstruction orders M. Timings are stated in frames per second. All images rendered with PCF use standard NVIDIA hardware filtered shadow test. Note that all processing happens on the GPU and that reconstruction order M determines the number of texture fetches per pixel that is shaded. For

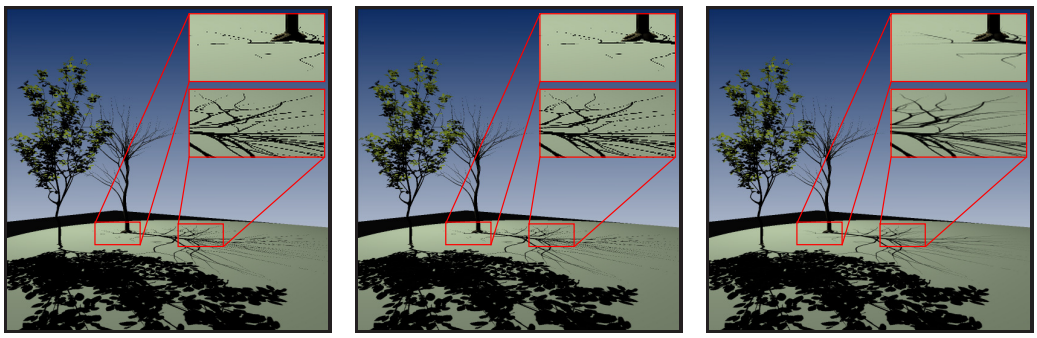
$C = no$	$S : 256^2$	$S : 512^2$	$S : 1024^2$	$S : 2048^2$
PCF	76 fps	74 fps	71 fps	69 fps
$M = 4$	64 fps	62 fps	60 fps	50 fps
$M = 8$	55 fps	53 fps	49 fps	38 fps
$M = 16$	47 fps	45 fps	39 fps	26 fps
$C = 3x3$	$S : 256^2$	$S : 512^2$	$S : 1024^2$	$S : 2048^2$
$M = 4$	63 fps	61 fps	57 fps	43 fps
$M = 8$	53 fps	50 fps	44 fps	30 fps
$M = 16$	42 fps	39 fps	32 fps	19 fps
$C = 7x7$	$S : 256^2$	$S : 512^2$	$S : 1024^2$	$S : 2048^2$
$M = 4$	62 fps	60 fps	53 fps	36 fps
$M = 8$	52 fps	49 fps	41 fps	24 fps
$M = 16$	41 fps	38 fps	28 fps	14 fps

Table 6.1: Frame rates for the complex scene (365k faces) from Figure 6.10 for varying shadow map sizes S and varying reconstruction order M (screen resolution is 1024×768). We compare $16\times$ anisotropic tri-linear hardware filtering without additional convolution, a 3×3 and a 7×7 convolution kernel C.

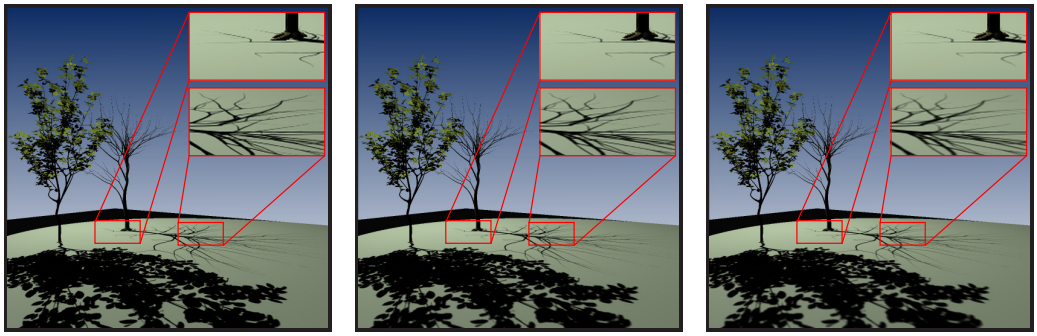
$M = 4$ we need two, for $M = 8$ we need four, and for $M = 16$ we need eight RGBA texture fetches. Timings include convolution (if applied) and mip-map generation for all basis textures. As can be seen, CSMs are generally slower than PCF but enable effective anti-aliasing.

Figure 6.5 shows the relationship between reconstruction order M and shadow intensity. A small M results in wrongfully brightened shadows when occluder and receiver are close to each other (lower square), and near contact points (slanted polygon), since the reconstructed step function is very smooth (see Figure 6.1 (a)). As M grows, the reconstructed step function becomes steeper, which produces correctly shaded shadows. In practice $M = 4$ yields satisfactory shadows without noticeable intensity artifacts. Even the slanted plane in Figure 6.5 (c) which touches the receiver plane achieves good shadowing quality using only 4 terms.

To demonstrate the image quality of CSMs, we chose a scene with high depth complexity where two tree models are lit from the side, in order to project long and thin shadows on a tilted plane. Figure 6.6 (a) was rendered with percentage closer filtering and illustrates the inability of PCF to reconstruct a thin branch close to the tree root due to shadow map aliasing. Figure 6.6 (b) shows that CSM renders the same result when bi-linear filtering is used for both mini- and magnification. In contrast, CSMs (see Figure 6.6 (c)-(f)) drastically increase image quality when using standard tri-linear filtering. Various convolution kernels can



(a) PCF (b) CSM Bi-linear (c) CSM Tri-linear



(d) CSM Tri-linear 3x3 Gauss (e) CSM Tri-linear 5x5 Gauss (f) CSM Tri-linear 7x7 Gauss

Figure 6.6: Different filter techniques applied to a scene with high depth complexity. (a) was rendered using PCF. (b) shows that bi-linear filtering for CSMs gives the same result as PCF. In contrast, CSMs (c)-(f) use better reconstruction filters and improve the image quality significantly.

be used to additionally hide shadow map discretization errors. In this example the shadow map resolution was 2048×2048 to capture fine details. Therefore the 7×7 convolution has limited extend in screen space. $16\times$ anisotropic filtering was enabled for tri-linear filtering.

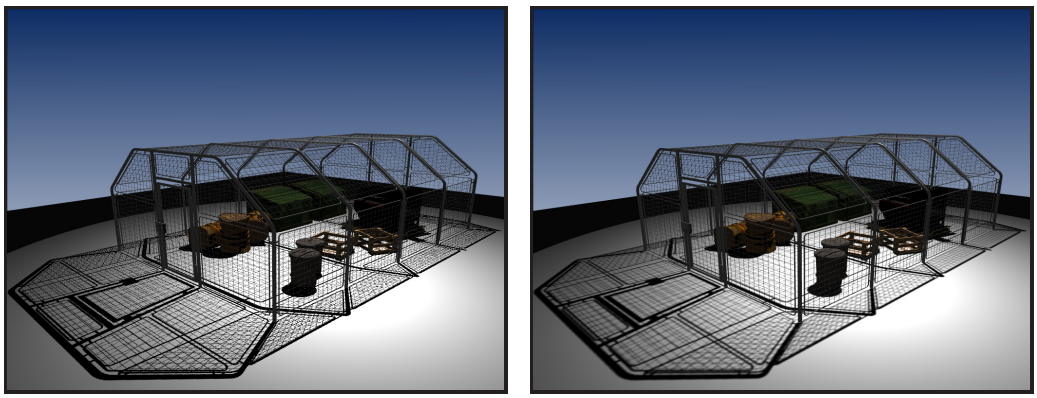
Figures 6.7 (a)-(h) present CSM examples of different shadow map resolutions and filter widths. As can be seen, even small shadow map resolution can produce nice shadows without visible discretization artifacts, if a large enough blur size is chosen. This can also be used as a crude approximation to soft shadows.

Figure 6.9 compares Variance Shadow Maps [Donnelly & Lauritzen, 2006] to our approach. VSMs are based on a statistical method to compute a filtered shadow test. However, when the variance increases within a filter region due to high depth complexity, light leaking artifacts appear, as illustrated in Figure 6.9 (a). Please note, that the fence itself does dot have high depth complexity, thus light leaks only appear where addition objects behind the fence add more depth

(a) SM=128^2 C=3x3 (b) SM=256^2 C=3x3 (c) SM=512^2 C=3x3 (d) SM=1024^2 C=3x3

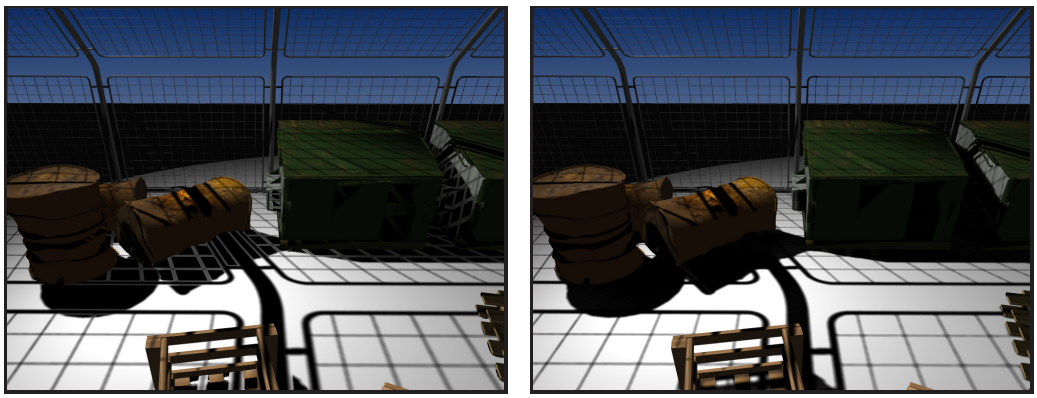
(e) SM 128^2 C=7x7 (f) SM=256^2 C=7x7 (g) SM=512^2 C=7x7 (h) SM=1024^2 C=7x7

Figure 6.7: Our method can reduce discretization artifacts of the shadow map by applying a convolution kernel to the CSM. This can even be used to render a crude approximation to soft shadows.



(a) Complex Geometry (PCF) (b) Complex Geometry (CSM)

Figure 6.8: Standard percentage closer filtering does not support tri-linear filtering and suffers from severe aliasing artifacts during minification. In contrast, Convolution Shadow Maps (CSM) enable tri-linear filtering of shadows and thereby achieve effective screen-space anti-aliasing. Additional convolution can hide shadow map discretization artifacts.



(a) Variance Shadow Maps (VSM) (b) Convolution Shadow Maps (CSM)

Figure 6.9: Comparison of VSMs (a) and CSMs (b). Scenes with high depth complexity such as this fence in front of other objects cause high variance in the convolution kernel. In such cases VSMs suffer from light leaking artifacts. (b) shows that CSMs correctly reconstruct the shadow function and render shadows without artifacts.

complexity and therefore increase the variance within the filter kernel. Convolution Shadow Maps do not suffer from these artifacts and can deal with high depth complexity.

The final example emphasizes that filtering Shadow Maps drastically reduces aliasing due to minification, e.g., when a scene moves further away. The top row in Figure 6.10 shows aliasing (spatial and temporal) in the PCF renderings. The bottom row show the same scene rendered with CSMs. Note how the shadow is anti-aliased (again spatial, as well as temporal).

6.4 Discussion

We have considered two other possible expansions: Taylor expansion and locally supported functions. The Heaviside step function can be approximated by a smooth analytic function (e.g. the sigmoid function), and subsequently expanded around $(d-z)=0$. With some algebraic manipulation, it is possible to group terms in a factorized sum like Equation 5.6. But, the approximation error will not be constant with respect to $(d-z)$. Moreover, it often happens that $|d-z|$ is large, in which case the approximation diverges.

Locally supported functions like the block or hat basis, also produce a variable error because they lack shift-invariance. Furthermore, they are prone to severe temporal artifacts (popping). In general, any basis expansion always incurs an

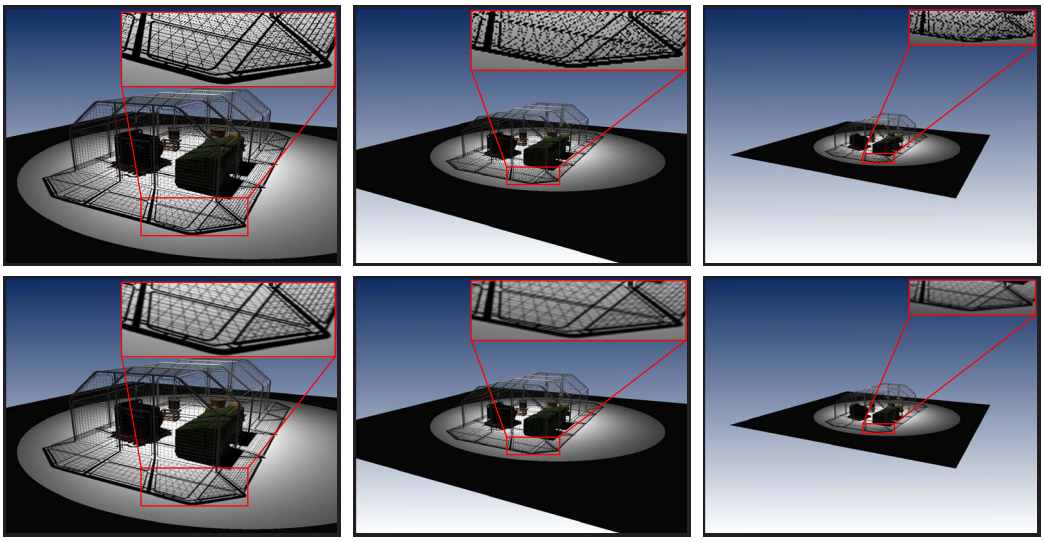
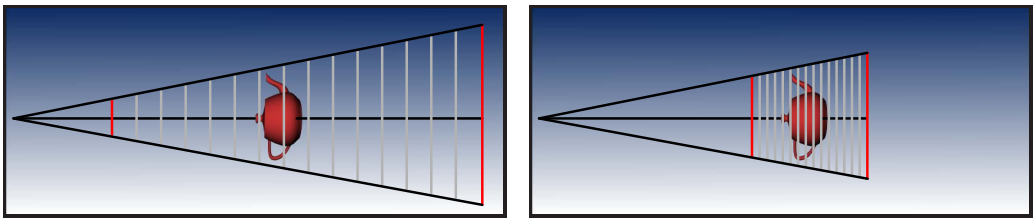


Figure 6.10: Tri-linear filtering is especially important when a scene is moved far away from the camera and minification occurs. Here we compare PCF (top row) to CSMs using regular tri-linear filtering with a 5*x*5 filter kernel (lower row).



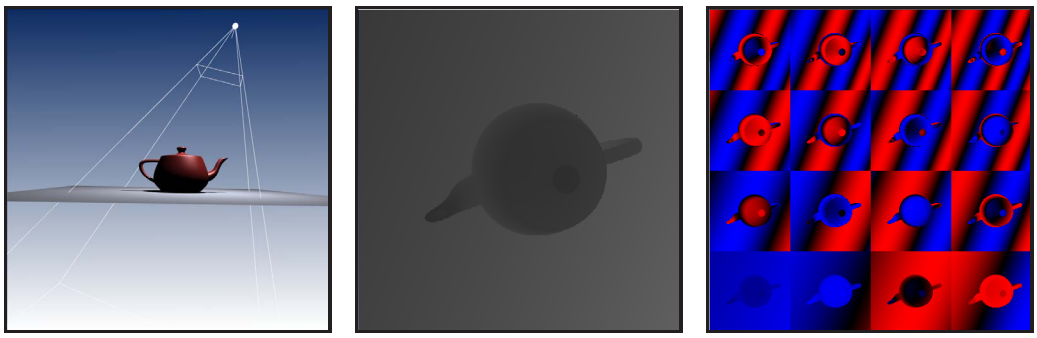
(a) Loosely packed clipping planes (b) Tightly packed clipping planes

Figure 6.11: Influence of the distance between near- and far-clipping planes on the depth sampling rate. A tightly fitted frustum maximizes the accuracy.

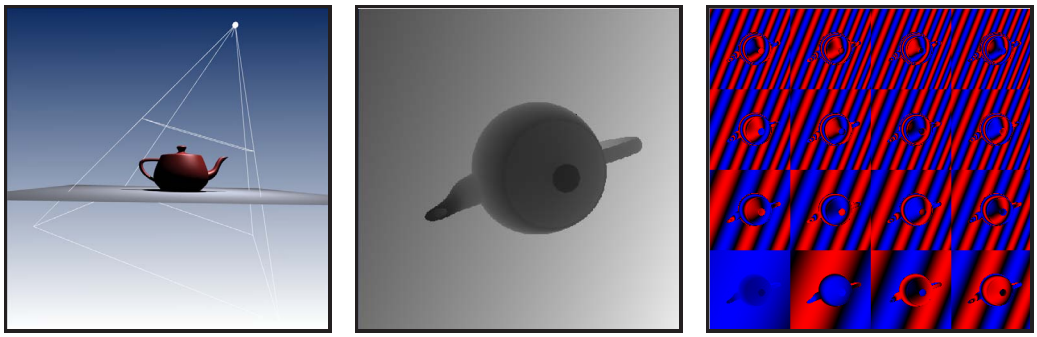
error due to truncation and may need to be accounted for. The Fourier series serves our purpose well, but it is conceivable that other viable solutions exist as well.

We would further like to discuss the impact the near- and far-clipping planes have on the *z*-value resolution in the depth buffer, and the resulting basis image frequency content. As an example were we compare a teapot enclosed by a loosely and tightly fitted light view frustum show in Figure 6.11 (a) and (b) respectively. The depth values are linearly distributed between the near- and far-planes, where the near-plane maps to 0 and the far plane maps to 1.

This affects the precision up to which two depth samples can be discriminated from each other on one hand. As for the example in Figure 6.11 (a), the teapot can

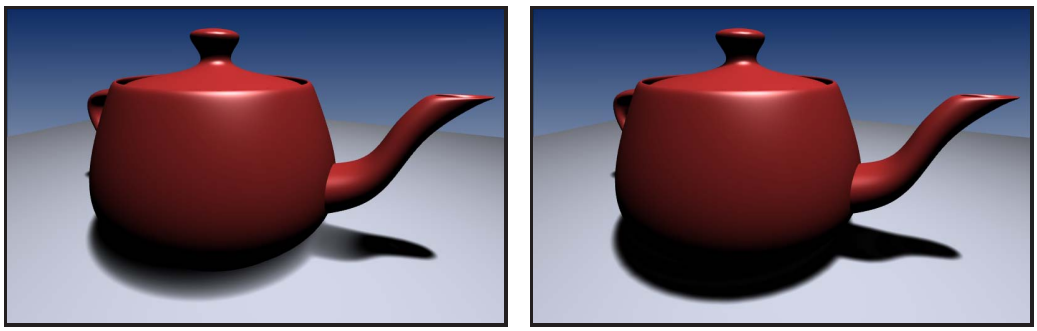


(a) Loose near-/far plane (b) Resulting z-values (c) Depth sampling rate



(d) Tight near-/far plane (e) Resulting z-values (f) Depth sampling rate

Figure 6.12: Two example setups to illustrate the effect of tightly fitted clipping planes. (a) and (d) show the scene and clip plane setup. (b) and (e) show the resulting depth map. (c) and (f) visualize the difference in the basis images.



(a) Shadows for loose clipping planes (b) Shadows for tight clipping planes

Figure 6.13: The quality of the shadow test reconstruction depends on the depth sampling rate. The tighter the frustum, the higher the frequency content of the basis images, which yields a better shadow reconstruction.

only be sampled at three locations, whereas a tight view frustum shown in Figure 6.11 (b) allows for a much higher sampling rate. On the other hand, this directly relates to the frequency which we will receive in the basis images. The larger the depth range for the geometry (equal to the tighter the clipping planes are), the more frequencies are sampled, hence resulting in better quality. A visualization of the scene setup, the Shadow Map, and the resulting basis images are presented in Figure 6.12. The quality difference is shown in Figure 6.13.

Chapter 7

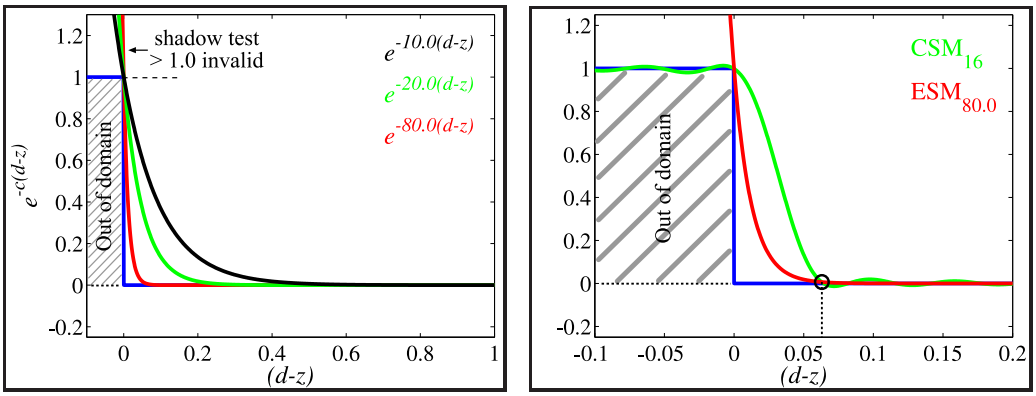
Exponential Shadow Maps

We now present *Exponential Shadow Maps* (ESMs) [Annen et al., 2008b], our second solution to the shadow test linarization we proposed in Chapter 5. We introduce a simple and efficient approach by approximating the shadow test using an exponential function. ESMs are inspired by CSMs, but use a single-term approximation, whereas CSMs use more (typically 16) terms. Compared to Convolution Shadow Maps, this technique is therefore faster, consumes less memory, and shows better behavior for close contact shadows. In order to achieve these goals we treat shadow quality in certain configurations for speed and memory savings. More precisely, our exponential approximation assumes that the support of a filter kernel does not contain surface samples (i.e., z-values) that lie beyond the distance from each screen pixel's world-space position to the light source. This approximation holds for many cases, e.g., for rendering a shadow on a large receiver, like a floor. When the assumption does not hold, we fall back to PCF, which typically only happens for a small fraction of pixels on the screen.

7.1 Exponential Approximation

We outline the theory behind ESMs in this section, and discuss a practical implementation. ESMs are based on a simple observation related to the domain of the shadow test, in other words, the d and z parameters in $f(d,z)$. Consider a point $\mathbf{x}$ seen by the camera, we know that the distance to the light source must be larger than or equal to the corresponding z-value read from the Shadow Map, because a depth map always stores the closest surface to the light source and therefore, $d(\mathbf{x}) - z(\underline{\mathbf{x}_\mathbf{l}}) \geq 0$ holds. However, in practice, this is not always true.

Before discussing when this happens, we will first outline how we can exploit this assumption in order to simplify the shadow test. Finally, we discuss how to deal with cases that violate the assumption.



(a) Shadow test domain (b) CSM_{16} vs ESM_{80}

Figure 7.1: ESMs assume that the domain of the shadow test is always positive $[(d-z) \geq 0]$ (a). As a result the shadow test can be approximated by an exponential decay. A larger factor c yields a better approximation. (b) shows that an ESM_{80} achieves better quality than a CSM_{16} (with an offset of -0.032). (The abscissa in (b) has been scaled to emphasize the difference.)

Let us for now assume that $d \geq z$. In that case we can define the shadow test $f(d,z)$ as:

$$f(d,z) = \lim_{\to\infty} e^{-\ (d-z)}$$

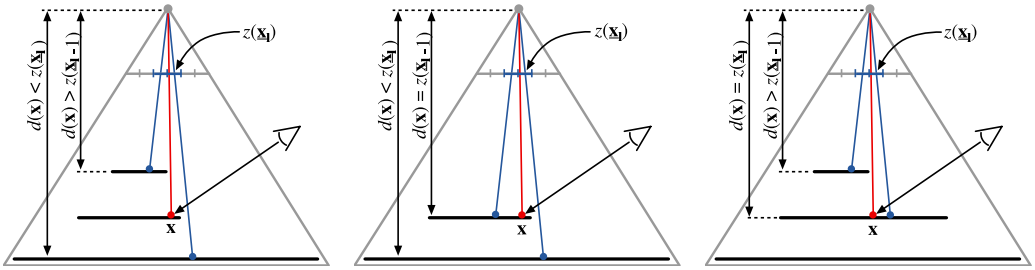
which can be approximated by filling in a large positive constant c for . This exponential function can be separated into factors depending on d and z:

$$\begin{aligned} f(d,z) &= e^{-c(d-z)} \\ &= e^{-cd}e^{cz}. \end{aligned} \tag{7.1}$$

We now continue by filtering the shadow function s, to yield the filtered shadow value value s_f. We represent the filtering operation in the same way we did in the previous chapter, and we fill in the exponential approximation:

$$\begin{aligned} s_f(\mathbf{x}) &= \left[w * f\big(d(\mathbf{x}),z\big)\right](\underline{\mathbf{x}_\mathbf{l}}) \\ &= \left[w * \left(e^{-cd(\mathbf{x})}e^{cz}\right)\right](\underline{\mathbf{x}_\mathbf{l}}) \\ &= e^{-cd(\mathbf{x})}\left[w * e^{cz}\right](\underline{\mathbf{x}_\mathbf{l}}) \end{aligned} \tag{7.2}$$

We see that shadow filtering now has become equivalent to applying a filter directly to the exponent-transformed depth values, which can be done beforehand.



(a) Failure case 1 (b) Failure case 2 (c) Correct filtering

Figure 7.2: ESM filtering. Red dots denote a camera sample and blue a shadow map sample. (a) shows a failure case where $\mathbf{x}$ should be darkened by 50% but its intensity is incorrectly increased because $z(\underline{\mathbf{x}}_\mathbf{l}) > d(\mathbf{x})$. This violates our assumption. (b) illustrates a similar case but here $\mathbf{x}$ is lit anyway and therefore does not cause artifacts. In both cases however a failure is detected and we enable PCF. (c) depicts a setup where our assumption holds and correct filtering is applied.

7.1.1 Choice of Exponent

A higher value c results in a steeper fall-off, and thus a better approximation of the shadow test; see Figure 7.1 (a). If c is not high enough, we will observe light leaking artifacts, similar to those we observed for Convolution Shadow Maps. However, there is an upper bound for c, depending on the precision of the floating point representation. We empirically determined an optimal value of $c = 80$ for 32-bit floating point numbers, which is unaffected by precision issues. It even gives a better approximation than CSMs with 16 basis functions (Figure 7.1 (b)). We abbreviate the reconstruction order M of CSMs and parameter c of ESMs as lower script values (e.g. CSM_{16} and ESM_{80}).

7.2 Violation of Assumption

Let $\Delta_\mathbf{x} = d(\mathbf{x}) - z(\underline{\mathbf{x}}_\mathbf{l})$. In the previous section, we assumed that $\Delta_\mathbf{x} \geq 0$. If not, the shadow test returns an arbitrarily large number as the new expansion does not converge to 1.0 but grows exponentially. We will discuss how this affects the results in the following two cases.

Without Filtering. We first analyze the case when the Shadow Map is not filtered (nearest neighbor sampling). In unshadowed areas, $d(\mathbf{x})$ should ideally be equal to $z(\underline{\mathbf{x}}_\mathbf{l})$. However, the precision of the Shadow Map is finite due to the limited spatial and numerical resolution. Consequently, $d(\mathbf{x})$ will only be approximately equal to $z(\underline{\mathbf{x}}_\mathbf{l})$, especially for slanted surfaces. In standard Shadow Mapping, this

leads to the well-known "shadow acne" problem. In our case, the reduced precision may incur negative $\Delta_{\mathbf{x}}$ values, yielding an overflow of the shadow function (i.e., a value larger than one). To overcome this problem, we can simply clamp the exponential to one.

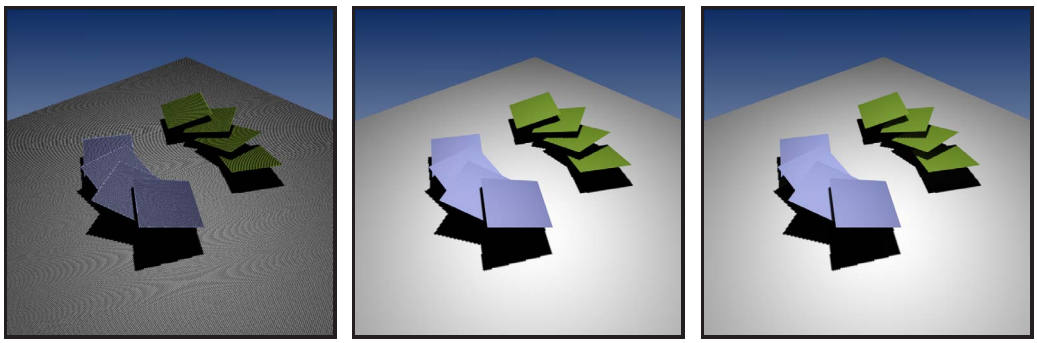
With Filtering. Similar to CSMs and VSMs, ESMs can be filtered prior to using it for rendering the actual shadows. However, z-values under the support of the filter will not necessarily be smaller than a given $d(\mathbf{x})$. For instance, this happens at slanted surfaces, or possibly at depth discontinuities. Consequently, we will get an erroneous filter response due to an overflow of the exponential. Clamping the values for each individual sample, would require us to resort to a PCF-style method, which defeats the purpose of pre-filtering. Figure 7.2 (a) and (b) illustrate two common failure cases when $\Delta_{\mathbf{x}}$ becomes negative.

7.2.1 Frequency of Violation

In most cases, the filter support contains z-values that are smaller than $d(\mathbf{x})$; see Figure 7.2 (c). When sampling points that are far away from shadow borders, the z-values are either all blockers (fully in shadow), or represent the sampled surface itself (fully illuminated) and our assumption holds. This occurs quite often, as most of the pixels are either fully in shadow or fully illuminated. Furthermore, the assumption holds for large receivers (e.g., a floor), in which case all blockers lie in front of the receiver, with respect to the light source.

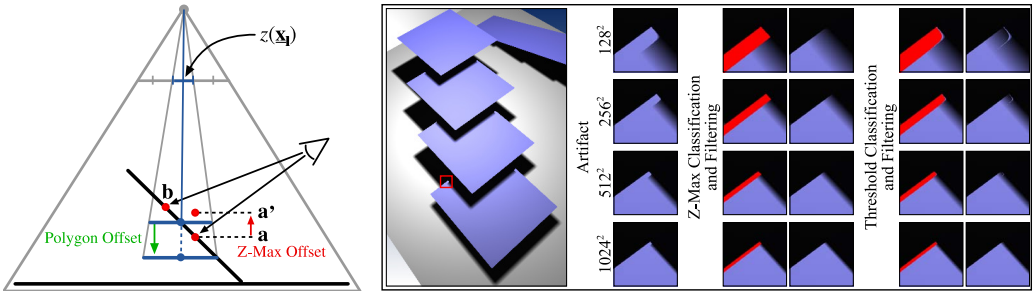
Even if the assumption is violated, the effect may not be visible. For example, unoccluded slanted surfaces (see Figure 7.4 (a)), may be sampled above the stored z-value (denoted by b in the figure), and therefore overflow. However, this overflow can be easily clamped to one (i.e., fully visible), not introducing artifacts. Since this surface is actually supposed to be fully visible, the violation goes unnoticed. In Section 7.3, we introduce two methods to classify pixels where the assumption is violated. For these pixels, we (can) fall back to a custom filtering solution in order to avoid artifacts.

Polygon Offset Regular shadow mapping suffers from the so-called "shadow acne" artifact which we have seen in the Chapter on shadow algorithms, and produces erroneous self-shadowing due to precision issues and is illustrated in Figure 7.3 (a). Note that we describe the polygon offset in OpenGL terms where an offset o is computed by $o = m \cdot factor + r \cdot units$, where m is the maximum polygon depth slope and r is the smallest value that ensures a resolvable offset. This can be solved by slightly offsetting the z-values away from the light source (see Figure 7.2 (d) and 7.3 (b)). In practice, polygon offsetting is not required for ESMs, because the exponential does not decay fast enough over such small distances.



(a) PCF (F=0, U=0) (b) PCF (F=2, U=1) (c) ESM (F=0, U=0)

Figure 7.3: Shadow acne and polygon offset (F=factor, U=units in OpenGL format). (a) without polygon offset numerical imprecision generates incorrect self-shadowing. (b) ESMs are less prone to numerical inaccuracies because the exponential decay is not steep enough over such small distances.



(a) Polygon and Z-Max Offset (b) Classification

Figure 7.4: (a) describes the difference between polygon offset and Z-Max offset which is important during failure classification. (b) shows ESM failure classification and fall back to PCF. We illustrate the artifacts and the difference in Z-Max and Threshold classification.

However, we still employ an additional offsetting but for another reason, namely for failure classification, which we will be detailed in the next section.

7.3 Classification and Fall Back Solution

The previous section explained in which situations our initial assumption of $d(\mathbf{x}) - z(\underline{\mathbf{x}}_l) \geq 0$ will be violated. This section presents two methods to check for such failure cases, and how to fix them. If a given pixel is classified as invalid, we fall back to a customized filtering which we refer to as *custom filtering* or *custom*

(a) Offset: 0.001 (b) Threshold: 0.020

Figure 7.5: (a) The influence of the offset, which is added to $d(\mathbf{x})$ for Z-Max classification. (b) The threshold on intensities shown in (b). (a) shows that the offset is underestimated and needs to be larger.

PCF. For performance reasons we opt to only use a 2×2 filter kernel similar to the bilinear version of PCF implemented in hardware Reeves et al. [1987] which we cannot use as we don't want to use the Shadow Map in addition to an 32-bit ESM (this would increase the memory consumption by 24- or even 32-bit times the shadow map resolution). Fortunately, we can simulate a filtered shadow test simply by evaluating the ESM at the four nearest neighbors followed by a bilinear interpolation on the clamped results.

7.3.1 Z-Max Classification

This approach relies on an additional texture in which we maintain the maximum z-values in a given neighborhood for the current shadow map. When we convert the z-values into the exponential basis, we simultaneously copy the z-values into the base level of the Z-Max texture. A max-filter is then used to build a mip-map structure, effectively storing maximum z-values for mip-mapped neighborhoods.

Classification of the pixel $\mathbf{x}$ works as follows. First $d(\mathbf{x})$ is computed and the z_{max} for the current filter kernel is fetched from the mip-mapped Z-Max texture (an appropriate LOD is selected to match the filter kernel). Checking if $d(\mathbf{x}) < z_{max}$ gives a conservative answer whether the assumption is violated for $d(\mathbf{x})$ or not.

To avoid misclassification of fully lit surfaces we have to add a small offset to $d(\mathbf{x})$. Note that this is problem is similar to the original polygon offsetting but

it works in the exact opposite direction (see Figure 7.4 (a). We want z_{max} to be slightly smaller so that a lit surface is not incorrectly flagged. The effect of the offset can be seen in Figure 7.5 (a).

7.3.2 Threshold Classification

A second option to check if our assumption holds for a given pixel is to first evaluate the ESM result and then check if it exceeds $1+$ where is a given threshold. This essentially checks if a large ESM value contributed to the result indicating that the assumption is violated (then large values occur to exponential growth depicted in Figure 7.1). The effect of Thresholding compared to the Z-Max method is depicted in Figure 7.5 (b).

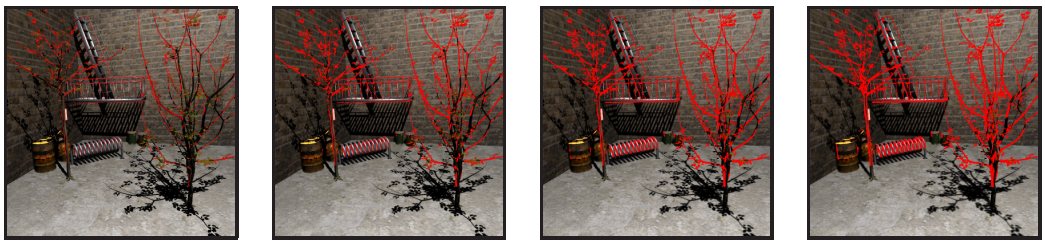
7.4 Implementation

Integrating ESMs into an existing rendering pipeline is straightforward. To generate exponential basis images we use a 32-bit floating point depth texture available through the *NV_depth_buffer_float* OpenGL extension by writing $exp(cz)$ instead of regular z values. In our implementation we also use a linear depth buffer. Any additional convolution is applied to the exponential basis image in the same manner as for Convolution Shadow Map. Rendering shadows with ESMs is now trivial. Instead of performing an explicit shadow test against d and z we simply evaluate Equation 7.1. Failure cases are detected by either one of the methods described in Section 7.3 and should incorporate the current polygon offset for shadow map generation for faithful detection.

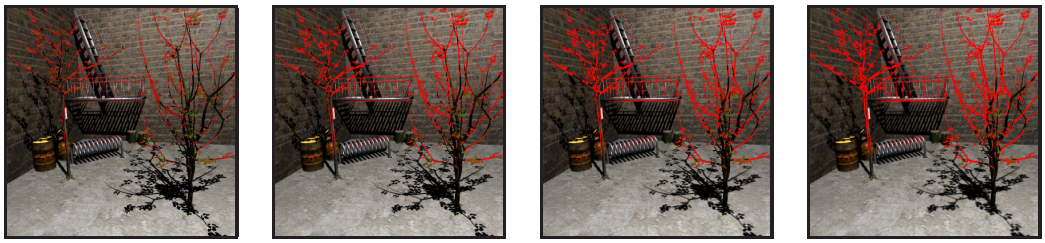
7.5 Results

This section demonstrates the quality and efficiency of Exponential Shadow Maps. All examples have been implemented in OpenGL 2.0 and rendered on a Dual-Core AMD Opteron PC with 2.6GHz and 2.75GB RAM equipped with an NVIDIA GeForce 8800 GTX graphics card. We have used the Thresholding approach as failure classifier. Rendering performance for various shadow algorithms are compared in Table 7.1. All memory statistics already contain the mip-map overhead (a factor of 1.3).

Figure 7.6 visualizes the impact additional convolutions have on the failure classification. The larger the filter kernel the more often our assumption fails and we have to perform custom filtering for all pixels indicated in red. Table 7.2 lists the exact numbers (for this measurement anti-aliasing was turned off).



(a) No conv. (3.0%) (b) 3x3 Gauss (6.1%) (c) 5x5 Gauss (7.9%) (d) 7x7 Gauss (9.3%)



(e) No conv. (2.8%) (f) 3x3 Gauss (4.9%) (g) 5x5 Gauss (6.1%) (h) 7x7 Gauss (7.1%)

Figure 7.6: Failure case classification. (a)–(d) uses Z-Max and (e)–(h) Thresholding. An increasing filter kernel size also increases the number of pixels for which ESMs cannot reconstruct a valid shadow test. For all red pixel we perform custom PCF filtering. The ratio of the total number of screen-space pixels (800×800) and failure cases is given in brackets.

Table 7.3 gives the performance timings (for Figure 7.4 (b)) regarding the failure detection and offers information for choices when one or the other classification approach is more applicable depending on the shadow map size.

A crucial situation for ESMs is minification, where the filtering size can be very large and thus the probability increases that the z-values within the kernel are larger than the current $d(\mathbf{x})$. Figure 7.7 shows how custom PCF avoids artifacts. We compare ESMs with regular tri-linear filtering without custom PCF, ESMs with custom PCF, and ESMs with anisotropic filtering and again no custom PCF.

It is interesting to note, that the fixed 2×2 PCF filter is sufficient to remove visible artifacts, which is most likely due to the fact that only very few pixels are filtered with PCF. Furthermore, the figure shows that when anisotropic filtering is turned on, the more expensive custom filtering is not really necessary. However, in situations where the filter kernel becomes very large, our custom PCF as well as anisotropic filtering may yield slightly less temporal coherence than the original CSM algorithm, which is due to our limiting the number of samples of our filter to 2×2 samples.

Figure 7.8 demonstrates that the filtering quality between ESMs and CSMs is virtually not distinguishable especially for scenes with high depth complexity

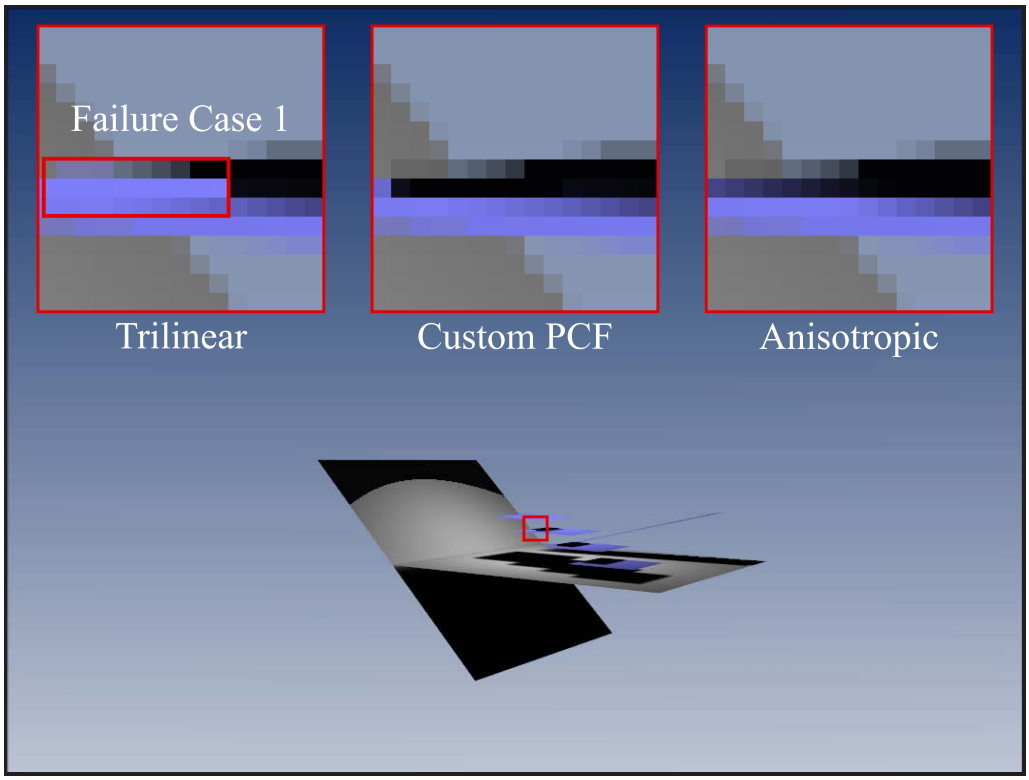


Figure 7.7: Anisotropic filtering. (a) without custom filtering camera pixels are incorrectly lit (red rectangle), (b) trilinear with custom PCF can prevent artifacts, and 10× anisotropic filtering without custom filtering often handles such failure cases properly.

owing to the surrounding fence. This examples illustrates the quality that ESMs achieve with 8× less memory and a significantly better performance. We also compare ESMs against VSMs showing less light leaking and better performance. The latest variant of VSMs, Summed-Area VSMs Lauritzen [2007a] also reduce light leaking but cannot completely avoid it and still have higher memory cost. The memory consumption for both sets of images was ESMs 21 MB, VSMs 42 MB, CSMs 170 MB.

In Figure 7.9 we evaluate how many samples an adaptive PCF filter would have to use to achieve anti-aliasing of similar quality as ESMs provide. To reach regular tri-linear ESM filtering quality, PCF has to use at least 16 or up to 36 samples which reduces the framerate significantly compared to ESMs. To match ESMs with an additional 5*x*5 Gauss convolution PCF needs at least 64 samples.

$C = no$	$ESM-T.$	$ESM-Z.$	VSM	CSM
$S: 512^2$	145 fps	132 fps	152 fps	83 fps
$S: 1024^2$	140 fps	123 fps	140 fps	68 fps
$S: 2048^2$	119 fps	101 fps	106 fps	40 fps
$C = 3x3$	$ESM-T.$	$ESM-Z.$	VSM	CSM
$S: 512^2$	141 fps	127 fps	149 fps	75 fps
$S: 1024^2$	132 fps	113 fps	131 fps	56 fps
$S: 2048^2$	102 fps	78 fps	87 fps	27 fps
$C = 7x7$	$ESM-T.$	$ESM-Z.$	VSM	CSM
$S: 512^2$	138 fps	119 fps	146 fps	71 fps
$S: 1024^2$	124 fps	100 fps	125 fps	46 fps
$S: 2048^2$	86 fps	61 fps	78 fps	19 fps

Table 7.1: Frame rates for the backyard scene from Figure 6.8. We compare ESMs (Thresholding and Z-Max) against VSMs and CSMs. Measurements include varying Shadow Map resolution and additional convolution (Gauss) kernel sizes. S and C denote the shadow map and convolution size.

Failure	*No Conv.*	3x3	5x5	7x7
Z – Max	3.0%	6.1%	7.9%	9.3%
Threshold	2.8%	4.9%	6.1%	7.1%

Table 7.2: Failure classification for the backyard scene from Figure 6.8. Even for an additional 7×7 convolution only 7.1% (or 9.3% for Z-Max test) of the screen-space pixels require special treatment.

7.5.1 Discussion

We have shown that the performance gain and memory reduction achieved by Exponential Shadow Maps is the result of a restriction on the parameter space of the shadow test function, which is motivated by the fact that the Fourier basis is hard to beat without any further assumptions. Usually, this assumption holds for the vast majority of screen-space pixel. Pixels for which the assumption does not hold are easy to detect and we have presented two alternative solutions to such failure scenarios. In the remainder of this discussion we would like to point out the difference between our classification schemes and addresses issues regarding the overall temporal coherence of ESMs.

Z-Max Classification is a conservative method and achieves more accurate results, but requires an additional texture map, which needs to be down-sampled using a max-filter. In case where additional convolutions are applied, the Z-Max texture also requires a max-filter of the same size reducing the overall frame rate.

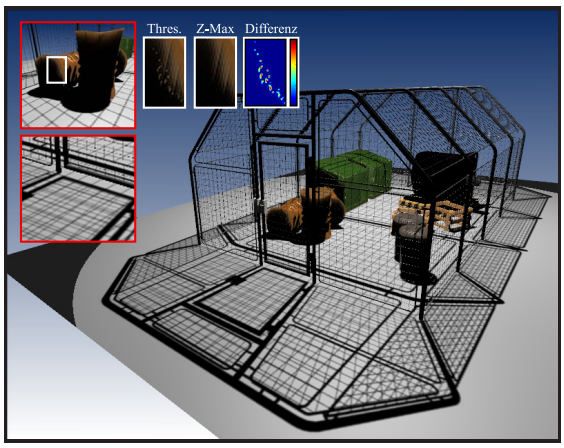
$C = no$	$S : 128^2$	$S : 256^2$	$S : 512^2$	$S : 1024^2$
$Z - Max$	410 fps	393 fps	374 fps	347 fps
$Threshold$	569 fps	556 fps	527 fps	438 fps
$C = 3x3$	$S : 128^2$	$S : 256^2$	$S : 512^2$	$S : 1024^2$
$Z - Max$	369 fps	357 fps	345 fps	273 fps
$Threshold$	547 fps	536 fps	495 fps	374 fps
$C = 5x5$	$S : 128^2$	$S : 256^2$	$S : 512^2$	$S : 1024^2$
$Z - Max$	358 fps	351 fps	338 fps	237 fps
$Threshold$	544 fps	527 fps	474 fps	342 fps

Table 7.3: Failure detection performance for Z-Max and Thresholding (800×800 viewport) for the scene from Figure 7.4 (b).

Threshold Classification does not require any additional resources and renders efficiently, but may suffer from small artifacts due to *false negative* classification errors. This can occur because thresholding is not a safe method to determine if the initial assumption is valid for all pixels within the filter kernel, as only the resulting filtered ESM value is checked. The visual quality and classification result is shown in Figure 7.4 (b).

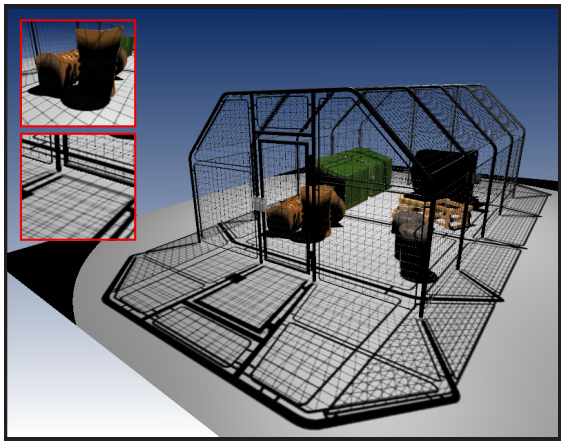
Temporal Coherence for ESMs is, independent of the classification, superior to regular shadow mapping methods. However, due to the assumption we make and the resulting need for custom filtering, ESMs exhibit slightly less temporal coherence in a small neighborhood of pixels as CSMs can achieve. As this usually only happens for a very small amount of screen space pixels we did not recognize noticeable differences between ESMs and CSMs.

Both quality and performance comparison show that the benefit of Z-Max decreases with texture size whereas its performance penalty increases at the same time. Figure 7.4 (b). According to this observation we opted to use thresholding for all results. It should be noted that our current custom PCF cannot remove artifacts that occur when additional convolutions are used, as the custom filter kernel size would be too large to be applicable in real-time applications.



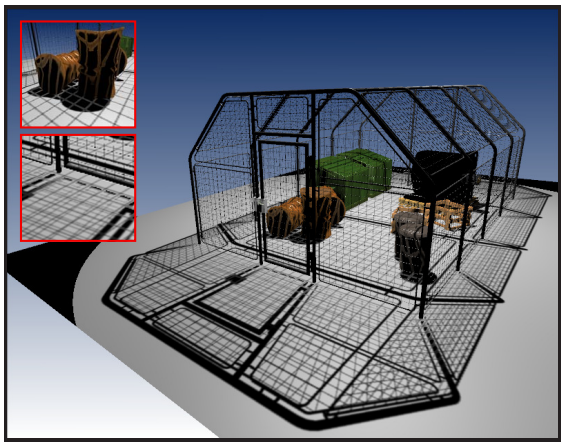
(a) ESM 66 FPS)

(b) ESM (94 FPS)

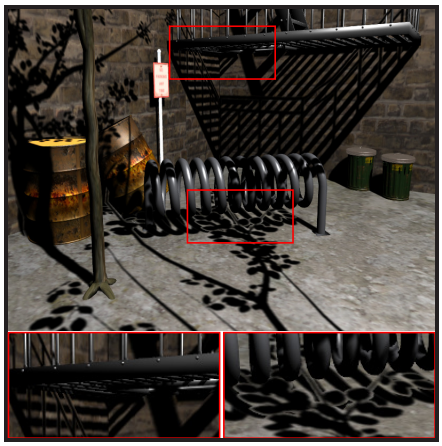


(c) CSM (21 FPS)

(d) CSM (22 FPS)

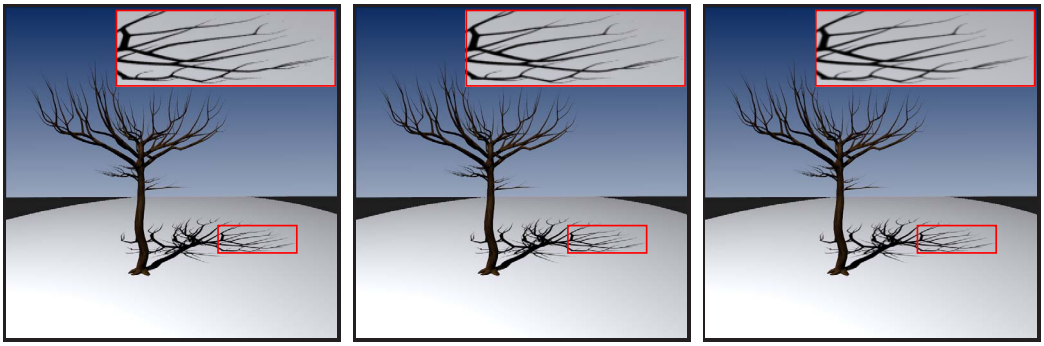


(e) VSM (60 FPS)

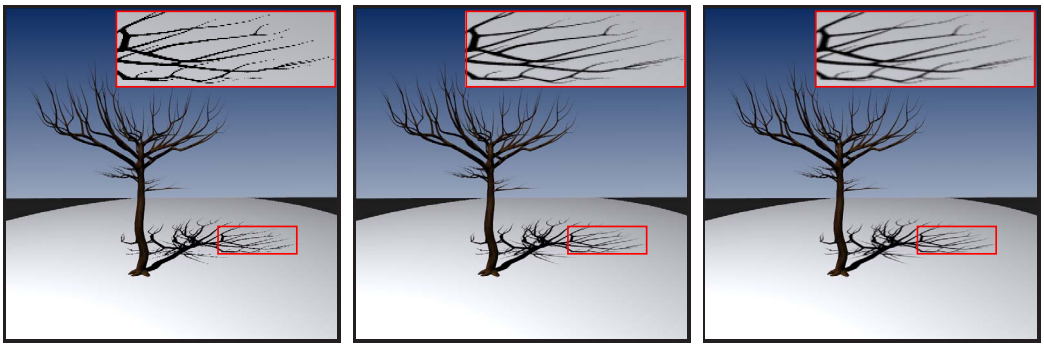


(f) VSM (84 FP)

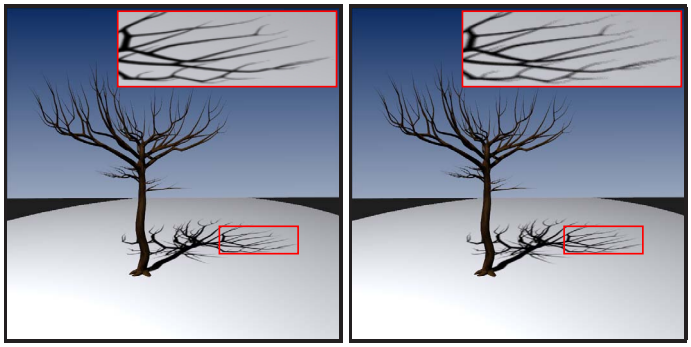
Figure 7.8: A complex fence scene (left), and a backyard scene (right), both rendered with a $2k \times 2k$ Shadow Map and 5×5 Gauss filtering. Like CSMs, ESMs also avoid the high frequency light leaking artifacts seen with VSMs.



(a) ESM trilinear (248 FPS) (b) ESM 3x3 (182 FPS) (c) ESM 5x5 (155 FPS)



(d) PCF (409 FPS) (e) PCF 4x4 (224 FPS) (f) PCF 6x6 (142 FPS)



(g) ESM 7x7 (136 FPS) (h) PCF 8x8 (95 FPS)

Figure 7.9: Quality and performance comparison between regular PCF with multiple samples and ESMs with additional convolutions. We compare bilinear, 4×4, 6×6, and 8×8 PCF versus trilinear ESMs, and trilinear ESMs with additional convolutions applied. At least 36 samples are necessary for PCF to match regular trilinear ESMs and 64 samples are required to achieve similar quality to ESMs with an additional 5*x*5 Gauss filter.

Part III

Pre-filtered Soft Shadows

Chapter 8

Convolution Soft Shadow Maps

So far we have only been concerned with shadow anti-aliasing. In this chapter we introduce *Convolution Soft Shadow Maps* (CSSMs) [Annen et al., 2008a], an extension to our mathematical framework to take our pre-filtering capabilities one step farther to generate plausible soft shadows efficiently. Our new method does not require any pre-computation, naturally handles dynamic objects regardless of their topology, and renders all-frequency shadows in real-time.

The Background chapter has shown that rendering soft shadows for area light sources is challenging. Our goal is to render several area light sources in real-time without having to sacrifice visual quality. We argue that computing penumbrae at full physical accuracy is intractable in this case. Instead, reducing shadow accuracy slightly enables us to achieve very high frame rates while keeping the visual error at a minimum.

8.1 Plausible Soft Shadows Using Convolution

We build on convolution-based methods which simulate penumbrae by filtering shadows depending on the configuration of blocker, receiver, and light source [Soler & Sillion, 1998; Fernando, 2005]. These methods are approximate in general, but produce an exact solution if the light source, blocker, and receiver are planar and parallel [Soler & Sillion, 1998]. Fortunately, deviating from this geometric configuration still produces plausible results.

The advantage of computing shadows using convolution is two-fold: it is compatible with image-based representations, in particular Shadow Mapping, and thus scales well to scenes with a high polygon count. Second, convolutions can be computed efficiently using a Fourier transform [Soler & Sillion, 1998], or even in constant time if the shadows have been pre-filtered using mipmaps or summed area tables [Lauritzen, 2007a].

However, applying convolution to Shadow Maps in order to produce soft shadowing is not trivial. The size of the convolution kernel needs to be estimated based on the blocker distance as described by Soler & Sillion [1998], but when multiple blockers at different depths are involved there is no single correct blocker distance. To get a reasonable approximation of blocker depth we compute the average depth of the blockers over the support of the filter. This approach was taken by Fernando [2005], as well as Lauritzen [2007a].

Unfortunately, estimating this average is expensive since it seemingly requires to average depths from the shadow map in a brute force fashion. The strength of our technique is that it allows for both efficient filtering of the shadows as well as efficient computation of the average blocker depth. Both of these operations can be expressed with the same mathematical framework, and will be described in Section 8.2.

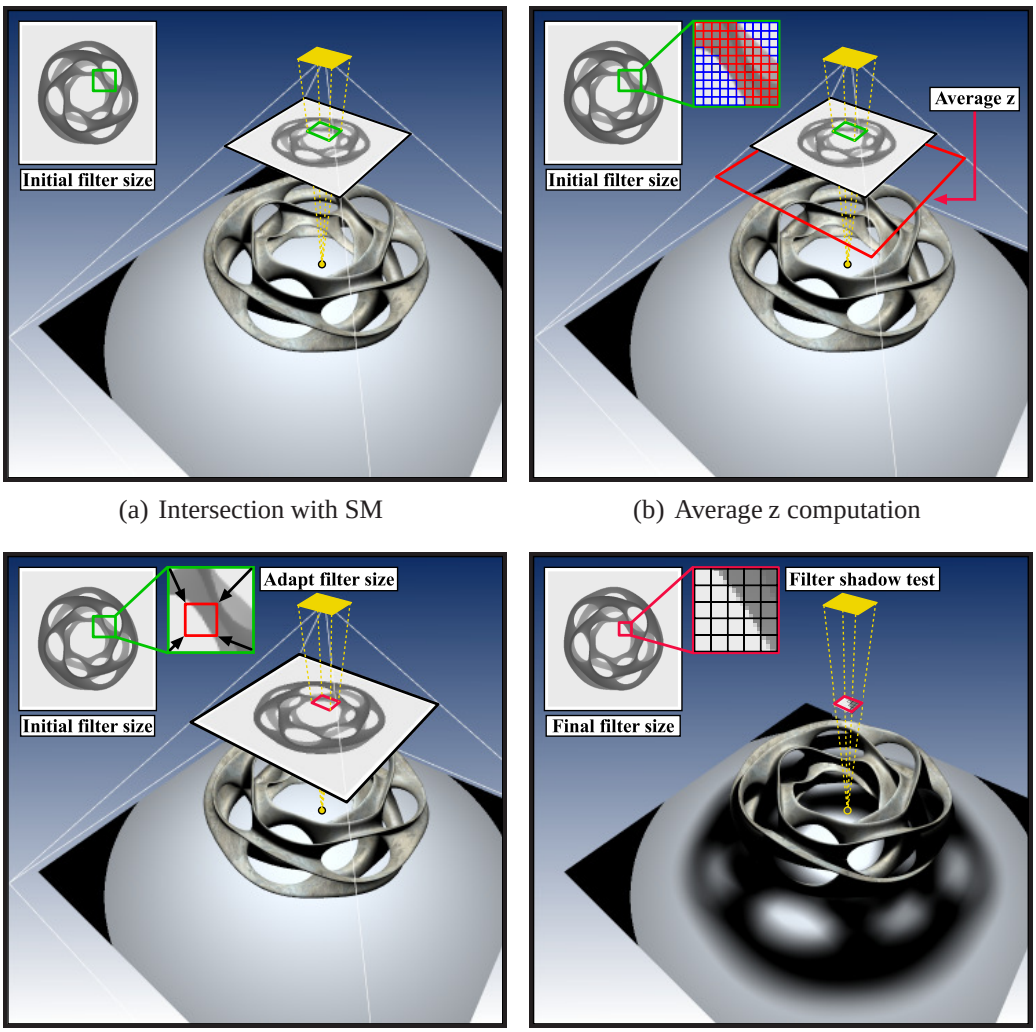
The main visual consequence of the average blocker depth approximation is that the penumbra width may not be estimated exactly (it is correct for the parallel-planar configuration described above though). We show that this approximation does not produce offensive artifacts, and even closely approximates the ground truth solution.

Figure 8.1 presents an overview of our soft shadow method and will be detailed in the following section. First, we determine an initial filter size according to the cone defined by the intersection of the area light source, the Shadow Map plane, and the current receiver point (a). This filter size shown in green is used to fetch the z_{avg} value from the pre-filtered average z-textures. We then virtually place the shadow map plane at the z_{avg} and compute the final filter width marked in red for soft shadow computation as shown in (c). The last part of our algorithm then reconstructs the visibility value for this point by a constant number of CSM texture lookups (d).

8.2 Convolution for Soft Shadows

As indicated above, soft shadows can be rendered efficiently through shadow map filtering and we therefore build on our previous approach Convolution Shadow Maps. As will be shown, CSMs can be extended to also compute the average blocker depth, which is required to estimate penumbra widths. We also introduce extensions that allow us to safely reduce the approximation order to further push rendering performance.

Review. In order to keep the discussion self-contained, we briefly review the theory from Chapter 5 again. We defined the shadow function $s(\mathbf{x})$, which encodes



(a) Intersection with SM (b) Average z computation

(c) Move SM to average z (d) Filter shadow test

Figure 8.1: Overview. (a) determine initial filter size (green) to fetch the z_{avg}. Then virtually place Shadow Map z_{avg} (b) and compute final filter width (red) (c). In the last step the incoming visibility is looked up from the CSM texture (d).

the shadow test, as:

$$s(\mathbf{x}) := f(d(\mathbf{x}), z(\underline{\mathbf{x}}_\mathbf{l})) = \begin{cases} 1 & \text{if } d(\mathbf{x}) \leq z(\underline{\mathbf{x}}_\mathbf{l}) \\ 0 & \text{if } d(\mathbf{x}) > z(\underline{\mathbf{x}}_\mathbf{l}), \end{cases}$$

and we saw when $f()$ is expanded into a separable series:

$$f(d(\mathbf{x}), z(\underline{\mathbf{x}}_\mathbf{l})) = \sum_{i=1}^{\infty} a_i(d(\mathbf{x}))B_i(z(\underline{\mathbf{x}}_\mathbf{l})),$$

we were able to spatially convolve the result of the shadow test through pre-filtering:

$$\begin{aligned} s_f(\mathbf{x}) &\approx \big[w * f\big(d(\mathbf{x}), z\big)\big](\underline{\mathbf{x}_\mathbf{l}}) \\ &\approx \sum_{i=1}^{N} a_i\big(d(\mathbf{x})\big)\big[w * B_i(z)\big](\underline{\mathbf{x}_\mathbf{l}}) \end{aligned} \tag{8.1}$$

where the basis images B_i are pre-filtered with the kernel w, which in practice is achieved through mipmapping each B_i or computing summed area tables Crow [1984]. At run-time, one only needed to weight the pre-filtered basis images by $a_i(d(\mathbf{x}))$ and sum them up. The next section derives an extension that allows to also employ this theory for the average blocker z computation.

8.2.1 Estimating Average Blocker Depth

The above pre-filtering of the shadow test results allows us to apply convolutions to soften shadow boundaries. However, for real soft shadows the size of the convolution kernel needs to vary based on the geometric relation of blockers and receivers [Soler & Sillion, 1998]. We follow Fernando [2005] and use the average depth value z_{avg} of all blockers that are *above* the current point $\mathbf{x}$ to adjust the size of the kernel.

Estimating the average blocker depth appears to be a very expensive operation. The obvious solution of sampling a large number of shadow map texels in order to compute the average depth value z_{avg} is very costly, and achieving good frame rates for large convolution kernels is not only difficult [Fernando, 2005] but also counterproductive for constant time filtering methods [Donnelly & Lauritzen, 2006; Annen et al., 2007; Lauritzen, 2007a].

The key insight into making this step efficient is that this selective averaging can be expressed as convolution and can therefore be rendered efficiently. To see this, let us first compute a simple local average of the z-values in the Shadow Map:

$$z_{avg}(\mathbf{x}) = \big[w_{avg} * z\big](\underline{\mathbf{x}_\mathbf{l}}) \tag{8.2}$$

Here, w_{avg} is a (normalized) averaging kernel. However, we only want to average values that are smaller than $d(\mathbf{x})$. Let us therefore define a "complementary" shadow test $\bar{f}$:

$$\bar{f}(d(\mathbf{x}), z(\underline{\mathbf{x}_\mathbf{l}})) = \begin{cases} 1 & \text{if } d(\mathbf{x}) > z(\underline{\mathbf{x}_\mathbf{l}}) \\ 0 & \text{if } d(\mathbf{x}) \leq z(\underline{\mathbf{x}_\mathbf{l}}). \end{cases} \tag{8.3}$$

which returns 1 if the shadow map z-value $z(\underline{\mathbf{x}_\mathbf{l}})$ is smaller than the current depth $d(\mathbf{x})$, and 0 otherwise. We can now use this function to "select" the appropriate z

samples by weighting them:

$$z_{avg}(\mathbf{x}) = \frac{\left[w_{avg} * \left[\bar{f}\big(d(\mathbf{x}),z\big) \times z\right]\right](\underline{\mathbf{x}_\mathbf{l}})}{\left[w_{avg} * \bar{f}\big(d(\mathbf{x}),z\big)\right](\underline{\mathbf{x}_\mathbf{l}})} \tag{8.4}$$

The denominator normalizes the sum such that it remains an average and is simply equal to the complementary filtered shadow lookup: $1-s_f(\mathbf{x})$. For the numerator we can approximate the product of the complementary shadow test and z using the same expansion as used in regular CSM:

$$\bar{f}\big(d(\mathbf{x}),z\big)z \approx \sum_{i=1}^{N} \bar{a_i}\big(d(\mathbf{x})\big)\bar{B_i}\big(z(\underline{\mathbf{x}_\mathbf{l}})\big) \times z(\underline{\mathbf{x}_\mathbf{l}}) \tag{8.5}$$

Here, coefficients $\bar{a_i}$ are coefficients and $\bar{B_i}$ basis images for $\bar{f}$. We can now approximate the average as:

$$z_{avg}(\mathbf{x}) \approx \frac{1}{1-s_f(\mathbf{x})} \sum_{i=1}^{N} \bar{a_i}\big(d(\mathbf{x})\big)\left[w_{avg} * \left[\bar{B_i}(z) \times z\right]\right](\underline{\mathbf{x}_\mathbf{l}}). \tag{8.6}$$

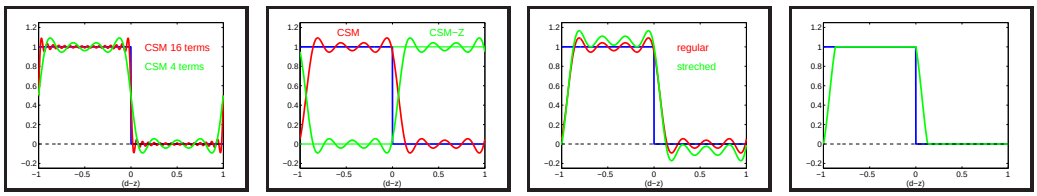
We will therefore compute new basis images $\left[\bar{B_i}(z) \times z\right]$ alongside the regular CSM basis images. We refer to this new approach for computing the average blocker depth as CSM-Z. As our z_{avg} computation also uses the Fourier series to approximate $\bar{f}()$ we need to insert the Fourier series into Equation 8.6 to fully resolve all data sets which we have to compute. Therefore, we approximate $\bar{f}()$ as:

$$\bar{f}(d(\mathbf{x}),z(\underline{\mathbf{x}_\mathbf{l}})) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \frac{1}{c_k}\sin\left[c_k(d(\mathbf{x}) - z(\underline{\mathbf{x}_\mathbf{l}}))\right], \tag{8.7}$$

with $c_k = \quad (2k-1)$. Inserting this term into the convolution from Equation 8.6 yields the following:

$$\begin{aligned} z_{avg}(\mathbf{x}) &\approx \frac{1}{1-s_f(\mathbf{x})}\left[w_{avg} * \left(\frac{1}{2} + \sum_{k=1}^{M} \frac{2}{c_k}\sin\left[c_k(d(\mathbf{x}) - z)\right]\right) z\right](\underline{\mathbf{x}_\mathbf{l}}) \\ &\approx \frac{1}{1-s_f(\mathbf{x})}\left[w_{avg} * \frac{z}{2} + \frac{2}{c_k}\sum_{k=1}^{M} \sin\left(c_k d(\mathbf{x})\right)\left(w_{avg} * z\ \cos(c_k z)\right) - \right. \\ &\qquad \left. \frac{2}{c_k}\sum_{k=1}^{M} \cos\left(c_k d(\mathbf{x})\right)\left(w_{avg} * z\ \sin(c_k z)\right)\right](\underline{\mathbf{x}_\mathbf{l}}). \end{aligned} \tag{8.8}$$

This reveals an important fact. It means there is an additional basis image containing $z/2$ values basically corresponding to a Shadow Map, see Figure 8.4, which needs to be filtered too. We now turn to the average-z computation and discuss some approaches to reduce the reconstruction order M.



(a) M=4 and 16 terms (b) CSM vs CSM-Z (c) Shift and Scale (d) Final Clamp

Figure 8.2: Fourier series expansion. (a) depicts the difference between a 16- and 4-term reconstruction. (b) CSM and CSM-Z are exactly opposite to each other. Ringing suppression is possible with appropriate scaling and shifting (c), followed by clamping the function to $[0,1]$ (d).

8.2.2 Initializing Average Depth Computation

When we want to estimate or approximate the penumbra size for a given camera sample we have to do this by finding the area over the Shadow Map over which we will perform the z_{avg} computation. A first idea is to intersect the frustum formed by the camera sample $\mathbf{x}$ in 3D and the virtual area light source geometry with the Shadow Map plane (as depicted in Figure 8.1(a)). Unfortunately, there is no clear definition of such a plane as the Shadow Map itself only represents a height field and does not have a certain plane location. We have found the near plane to work well for all our results. However, an iterative procedure is possible where one re-adjusts the location after an initial z_{avg} has been found. An other alternative is to initially take the nearest z-value from the Shadow Map as it represents the first possible occluder sample. However, this would require a z-min computation which we would like to avoid in favor for a better performance behavior.

8.2.3 CSM Order Reduction

In the previous chapter we proposed to expand f using a Fourier series. Unfortunately, this series is prone to ringing artifacts and the shadows at contact points may appear too bright unless a high order approximation is used as shown in Figure 8.2(a). We propose two changes that allow us to reduce the order significantly. First, we notice that with appropriate scaling, shifting, and subsequent clamping, ringing can be avoided completely. Figure 8.2 illustrates this. Scaling and shifting $f(d,z)$ such that ringing only occurs above 1 and below 0 is shown in (c). Whenever the function $f(d,z)$ is reconstructed we clamp its result to $[0,1]$ avoiding any visible artifacts (d).

A second problem with a low order series is that the slope of the reconstructed shadow test is not very steep when $(d-z) \approx 0$, as can be seen in Figure 8.2(d), and yields shadows that are too bright near contact points. A simple solution

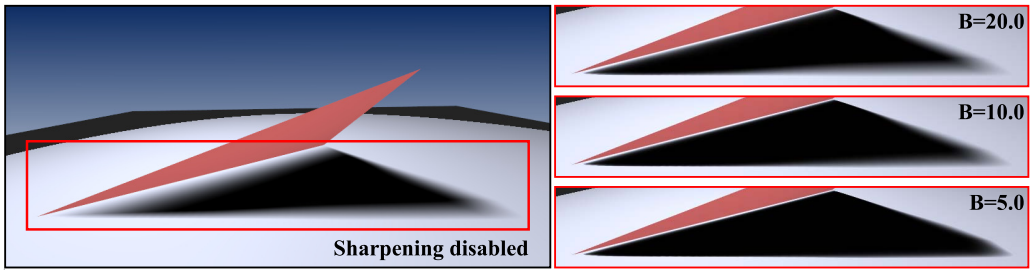


Figure 8.3: An illustration of the impact of sharpening parameters A and B. A is fixed to 30.0 whereas B is set to 5.0, 10.0, and 20.0 showing how B changes the spatial extend of the sharpening.

is to apply a non-linear transformation $G(v) = v^p$ to the filtered shadow value $s_f(\mathbf{x})$ with $p \geq 1$. This tends to darken the shadows and thus hides light leaking. If $p = 1$, nothing changes. On the downside, darkening also removes smooth transitions from penumbra regions, so we want to only apply it where necessary. When $d(\mathbf{x}) - z_{avg}(\mathbf{x})$ is small, we know that $\mathbf{x}$ is near a contact point where leaking will likely occur. Fortunately, this is also where penumbra should be hard anyway. We therefore compute an adaptive exponent p based on this difference:

$$p = 1 + A\, exp\big(-B\,\big(d(\mathbf{x}) - z_{avg}(\mathbf{x})\big)\big). \tag{8.9}$$

A controls the strength of the darkening, and B determines the maximal distance of z_{avg} from the receiver point for which darkening is applied to. Figure 8.3 shows this effect for a varying parameter B.

8.3 Illumination with Soft Shadows

8.3.1 Rendering Prefiltered Soft Shadows

Generating soft shadows with our new algorithm is similar to rendering anti-aliased shadows with Convolution Shadow Maps. First, the scene is rasterized from the center of the area light source and the z-values are written to the shadow map. Based on the current depth map two sets of images are produced: the Fourier series basis and its complementary basis images multiplied by the the shadow map z-values.

After we have generated both data structures we can run the pre-filter process. Note that when the convolution formula from Equation 8.6 is evaluated using a Fourier series, it also requires pre-filtering the Shadow Map due to the constant factor when multiplying $\bar{f}()$ by $z(\underline{\mathbf{x}}_\mathbf{l})$ (see Equation 8.8). In our implementation we support image pyramids (mip-maps) and summed-area-tables. Other linear

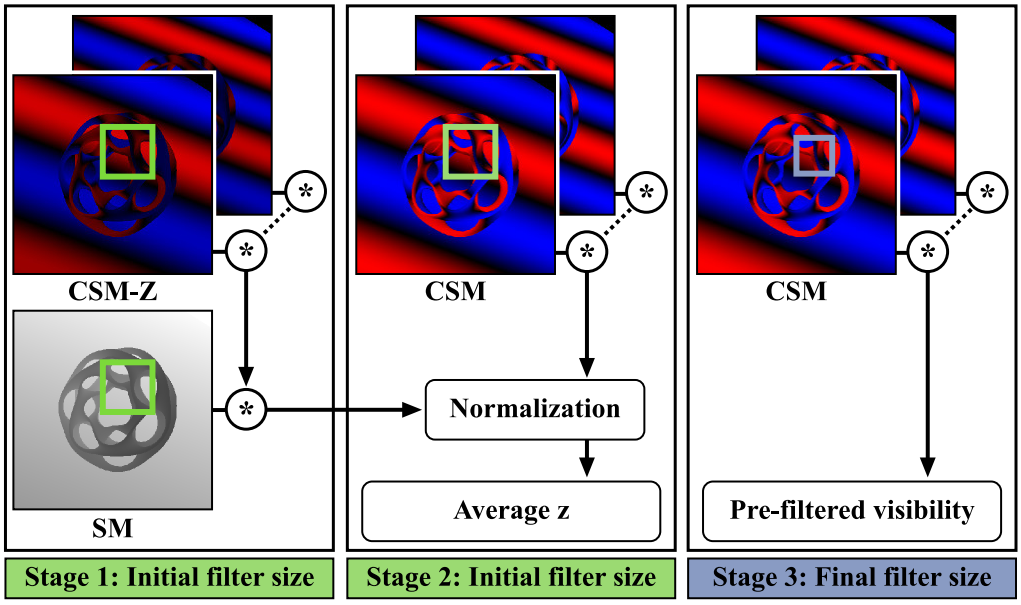


Figure 8.4: Convolution soft shadows pipeline. Stage 1 reconstructs a pre-filtered z_{avg}. The z_{avg} is passed to the 2nd stage for normalization. Thereafter, the final filter size is computed as described in 8.1(c), and the visibility is evaluated by a regular CSM reconstruction.

filtering operations are applicable as well. When filtering is complete, we start shading the scene from the camera view and employ convolution soft shadows for high-performance visibility queries. An overview of the different steps is given in Figure 8.4.

For each camera pixel we first determine an initial filter kernel width as previously shown in Figure 8.1 (a) to estimate the level of filtering necessary for the pixel's 3D position and feed this to stage one and two. Stage one reconstructs the average blocker depth based on the pre-filtered CSM-Z textures and the pre-filtered Shadow Map, which is then passed to the second stage for normalization. After normalization, the final filter kernel width f_w is adjusted according to the spatial relationship between the area light source and the current receiver. In particular, the triangle equality tells us the filter width: $f_w = \frac{\Delta}{d} \cdot \frac{(d - z_{avg})}{z_{avg}} \cdot z_n$, where Δ is the area light source width, d is the distance from $\mathbf{x}$ to the light source, and z_n is the light's near plane. The filter width f_w is then mapped to the shadow map space by dividing it by $2 \cdot z_n \cdot \tan(\frac{fovy}{2})$. A final lookup into the CSM textures yields the approximate visibility we wish to compute for the current pixel. All three stages together require only six RGBA and one depth texture access (for a reconstruction order $M = 4$). W

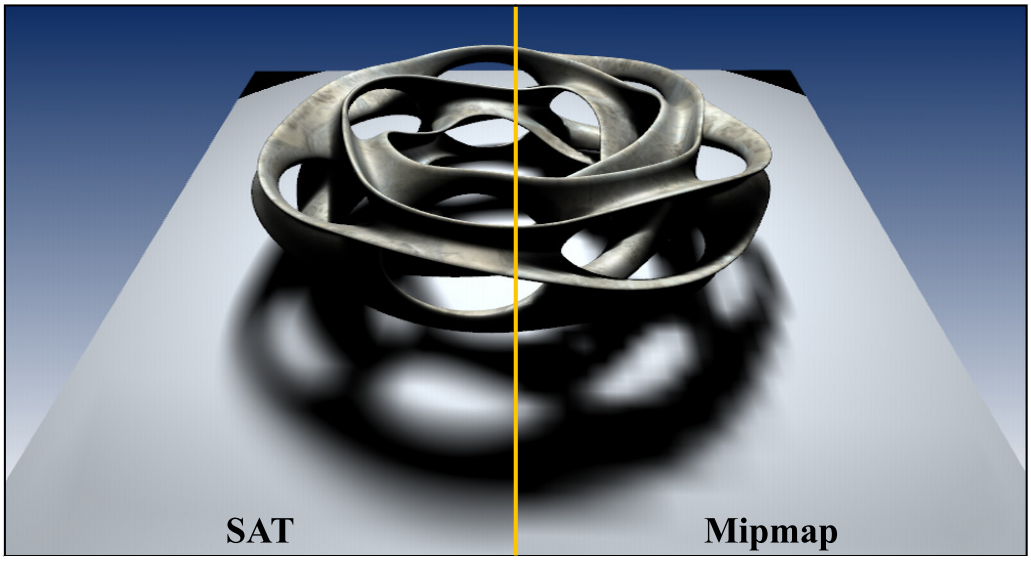


Figure 8.5: The difference in filter quality when using a summed-area-table (left) and a mipmap (right). Successive down sampling with a 2×2 box-filter introduces aliasing at higher mipmap levels.

8.4 Applications and Results

In this section we report on the quality and performance of our method. Our technique was implemented in DirectX 10 and all results were rendered on a Dual-Core AMD Opteron with 2.2GHz using an NVIDIA GeForce 8800 GTX graphics card. Our performance timings are listed in Table 8.1.

The overall performance of our technique and its image quality depend on the choice of pre-filtering, the number of area lights, and the individual light's shadow map size. The next results illustrate the impact of these individual factors.

We begin with a side-by-side comparison between mip-map and SAT-based soft shadows in Figure 8.5. Mip-maps produce less accurate results compared to summed-area-tables for rendering single lights, due to aliasing artifacts. For complex lighting environments, however, shadows from many light sources are averaged, which makes mip-mapping artifacts less noticeable (Figure 8.8).

Figure 8.6 shows that our method can easily deal with complex geometry while delivering high quality renderings. The closeups show how shadows soften as the area light size is increased. To capture fine geometric details we used a $1K \times 1K$ shadow map.

Figure 8.7 compares the shadow quality of several different algorithms to a reference rendering. We analyze two situations in particular, large penumbrae and close-contact shadows (see close-ups). Shadows rendered with our new technique

Figure 8.6: A very complex model illuminated by 1 AL with varying light source sizes from left (small) to right (large).

are very close to the reference, bitmask soft shadows perform slightly better at contact shadows and backprojection methods tend to overdarken shadows when the depth complexity increases. Percentage closer soft shadows produce banding artifacts in larger penumbra regions due to an insufficient number of samples.

In Figure 8.8 we show an example where an environment map is decomposed into a number of area light sources [Annen et al., 2008a]. Below the renderings we show the fitted area light sources and a difference plot. Rendering with 30 lights (Figure 8.8 (d)) already looks quite similar to the reference but some differences are noticeable. With 45 area lights, the differences to the reference are significantly reduced and the result is visually almost indistinguishable. This example illustrates that mip-mapping produces adequate results, while offering a more than threefold speedup compared to summed-area tables. The reference images in Figure 8.8 has been generated with 1000 environment map samples Ostromoukhov et al. [2004] using ray tracing.

Figure 8.9 shows the influence of the reconstruction order and sharpening. We render a foot bone model of high depth complexity and demonstrate the effect of the sharpening function $G()$. While contact shadows (toe close-up) are darkened and slightly sharper than the results rendered with $M = 16$, their larger penumbra areas are not influenced, which maintains the overall soft shadow quality.

	# Area Lights			
SM Type	1	10	20	40
MM: 128^2	258 fps	48 fps	28 fps	18 fps
MM: 256^2	228 fps	44 fps	25 fps	15 fps
MM: 512^2	189 fps	38 fps	20 fps	13 fps
MM: $1K^2$	110 fps	24 fps	5 fps	-
SAT: 128^2	128 fps	15 fps	8.8 fps	-
SAT: 256^2	110 fps	13 fps	7.5 fps	-
SAT: 512^2	89 fps	11 fps	6.0 fps	-
SAT: $1K^2$	52 fps	3 fps	1.5 fps	-

Table 8.1: Frame rates for the Buddha scene with 70k faces from Figure 8.8, rendered using reconstruction order $M = 4$. For many lights and high resolution shadow maps, our method may require more than the available texture memory (reported as missing entries).

Concerning memory consumption, mip-maps (SATs) with $M = 4$ require two 8bit (32bit) RGBA textures for storing the CSM and two 16bit (32bit) RGBA textures for storing the CSM-Z basis values.

8.5 Discussion

Failure Cases Our technique shares the same failure cases as PCF-based soft shadowing Fernando [2005]. We assume that all blockers have the same depth within the convolution kernel (essentially flattening blockers), similar to Soler and Sillion's method Soler & Sillion [1998]. This assumption is more likely to be violated for larger area lights. Nevertheless, shadows look qualitatively similar to the reference rendering, as shown in see Figure 8.7. The use of a single shadow map results in incorrect shadows for certain geometries. This problem is commonly referred to as "single silhouette artifacts", which we share with many other techniques Assarson & Akenine-Möller [2003]; Guennebaud et al. [2006].

Average Z Computation Computing the average z-value as described is prone to inaccuracies due to the approximations introduced by CSM-Z and CSM. These possible inaccuracies may lead to visible artifacts due to the division by $1 - s_f(\mathbf{x})$. Care must be taken to use the very same expansion for CSM-Z and CSM in order to avoid such artifacts.

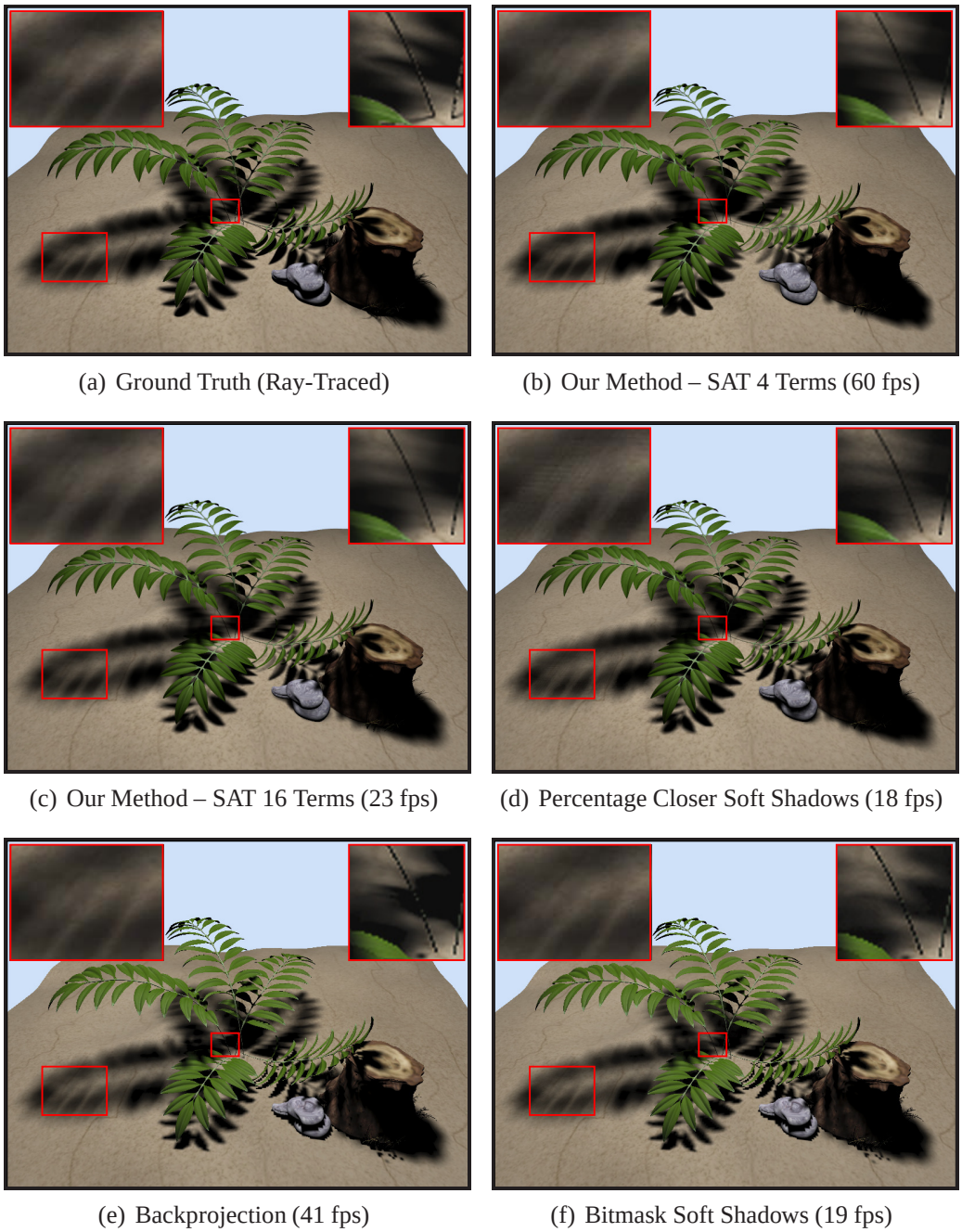
Ringing Suppression Our proposed ringing suppression using scaling and shifting followed by clamping does indeed reduce ringing and improves shadow darkness near contact points, but also sharpens shadows slightly as can be seen in Figure 8.9. However, this process is necessary to keep frame rates high as it allows the use of fewer terms in the expansion and the differences are barely noticeable. See the comparisons in the results section, all of which are rendered using ringing suppression.

Mipmaps vs. Summed Area Tables The quality that our method can achieve depends on the pre-filtering process. Mipmaps are computationally inexpensive, but their quality is inferior compared to SATs as they re-introduce aliasing again at higher mipmap levels. However, SATs require more storage due to the need to use floating point textures Hensley et al. [2005] especially when using many area lights. In the case of multiple area lights, as used for environment mapping, artifacts are masked and mipmapping is a viable option. Figure 8.5 compares both solutions.

Textured Light Sources Our method cannot handle textured light sources as the pre-filtering step cannot be extended to include textures. Nevertheless, we show how to decompose complex luminaires such as environment maps into uniform area lights.

Rectangular Area Lights Rectangular lights are supported, which is especially easy when using SATs. They can also be used in conjunction with mipmapping if the GPU supports anisotropic filtering. The aspect ratio of the area lights is limited by the maximum anisotropy the GPU allows. The increased cost of anisotropic filtering might warrant the use of several square area lights instead. The fitting process described in the last section can also be modified to fit square area lights instead of rectangular ones. In fact, this is what we have used for our results.

BRDFs We do not support integrating the BRDF across the light source domain, similar to most other fast soft shadowing techniques. However, for environment map rendering we do evaluate the BRDF in the direction of the center of each area light and weight the contribution accordingly.



(a) Ground Truth (Ray-Traced)

(b) Our Method – SAT 4 Terms (60 fps)

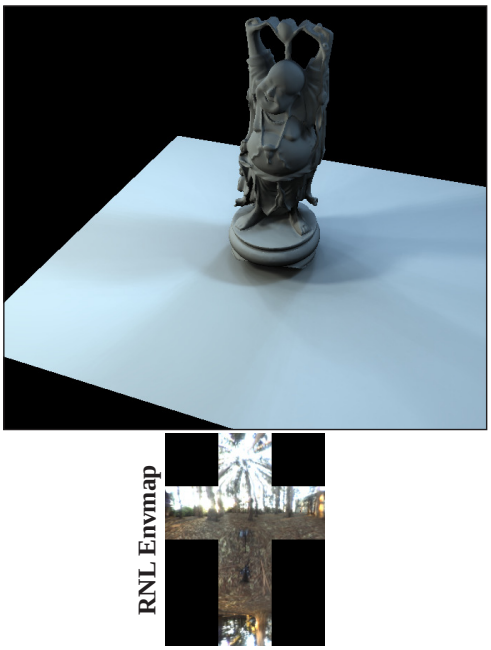
(c) Our Method – SAT 16 Terms (23 fps)

(d) Percentage Closer Soft Shadows (18 fps)

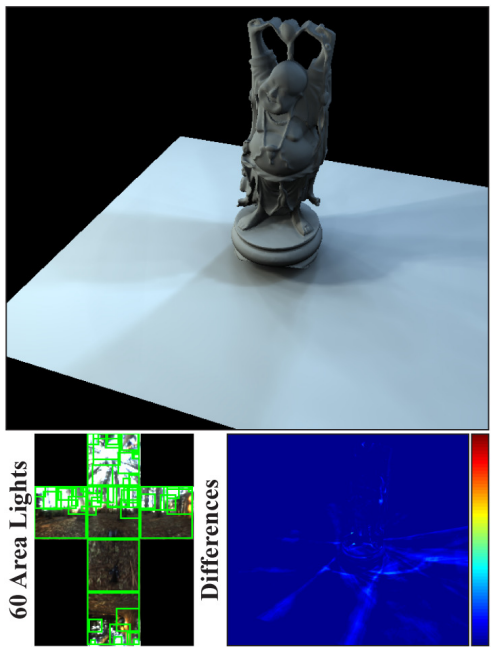
(e) Backprojection (41 fps)

(f) Bitmask Soft Shadows (19 fps)

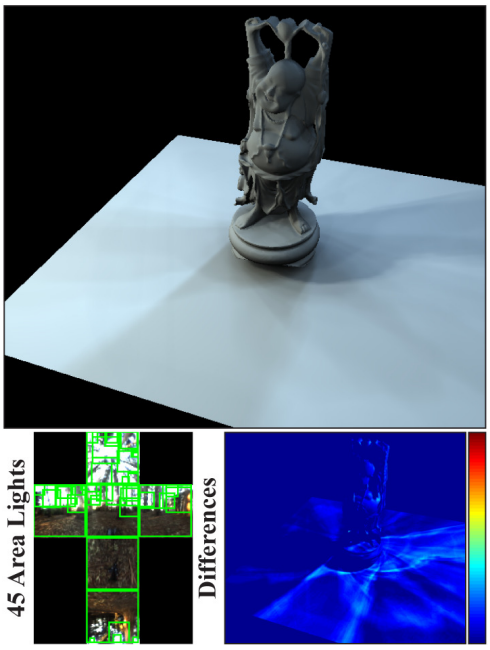
Figure 8.7: Shadow quality comparison of several methods (SM was set to 512 $\times$ 512, scene consists of 212K faces): ray-tracing (a), our method using SATs – 4 terms (b) and 16 terms (c), percentage closer soft shadows (d), backprojection Guennebaud [Guennebaud et al., 2006] (e), and bitmask soft shadows [Schwarz & Stamminger, 2007] (f).



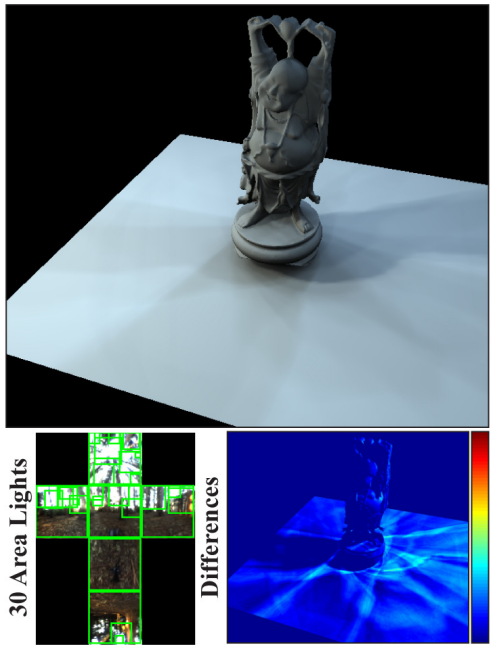
(a) RT SMs



(b) 60 ALs, $t = 0.035$ (9.8 fps)



(c) 45 ALs, $t = 0.025$ (14.1 fps)



(d) 30 ALs, $t = 0.015$ (18.4 fps)

Figure 8.8: Comparison between ray-tracing 1000 point lights (a), our technique with mip-maps using 60 (b), 45 (c), and 30 (d) area light sources. Each image shows the environment map with the the fitted light sources in green. SM resolution was set to 256×256.

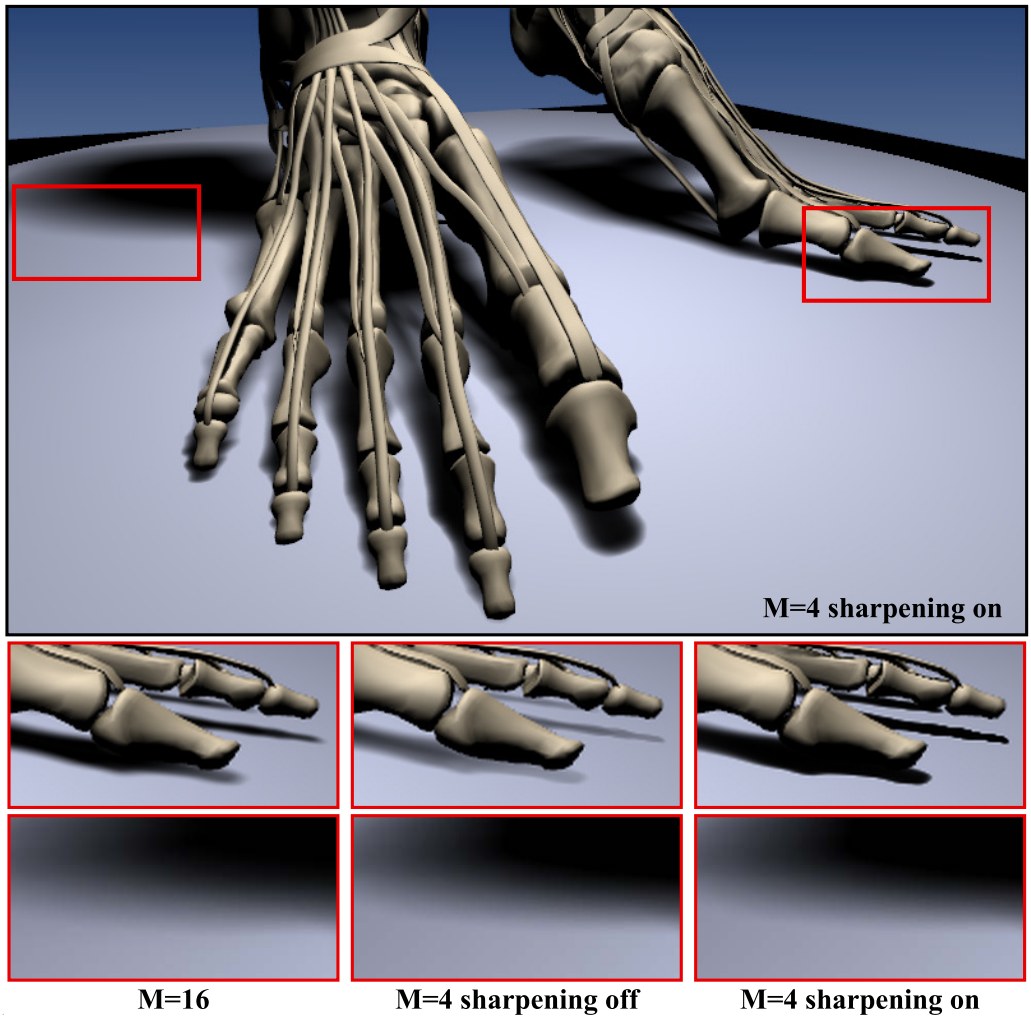


Figure 8.9: Influence of reconstruction order M and sharpening. The close-ups show that shadow darkening is restricted to contact points whereas larger penumbra areas remain unaffected and smooth.

Chapter 9

Summary and Conclusions

Main topic of this dissertation are new shadow algorithms for real-time and interactive applications such as games, virtual reality software, and animation studio tools. As the feedback-loop is a crucial factor in these environments, compromises regarding the image quality are inevitable to achieve real-time image updates. This dissertation contributes new algorithms to narrow this gap to afford high-quality shadow renderings in real-time. We choose to build on top of today's most popular real-time shadow method, Williams' Shadow Mapping approach, because it delivers high flexibility, scales well with the geometric scene complexity, and is simple to implement. To overcome the major limitation of Shadow Maps we develop the following new methods: *Convolution Shadow Maps*, *Exponential Shadow Maps*, and *Convolution Soft Shadow Maps*.

9.1 Summary

Linearization

We introduced a new mathematical framework which proposes to express the shadow test function as a sum instead of a piecewise function. This new theory allows us to translates ordinary depth maps into a new representation called basis images. We have shown that filtering the shadow test result is equivalent to filtering these basis images. Thus, pre-filtering basis images becomes possible and thereby we circumvent the limitation on regular Shadow Maps that they can only be filtered at run-time.

Anti-aliasing of Shadows

We presented Convolution Shadow Maps, a first solution to the linearization theory, which enables linear filtering of shadows by using a Fourier series expansion. In contrast to previous methods like Variance Shadow Maps, we do not have problems with high frequency light leaking, yet, our technique is very efficient and generally applicable.

In order to reduce memory consumption and improve the performance of Convolution Shadow Maps, we presented Exponential Shadow Maps, which incorporate an assumption to simplify the problem of Shadow Map filtering. Enforcing this assumption enables a single term exponential formulation. Compared to Convolution Shadow Maps, the quality trade-offs are very low compared to the increase in performance and memory savings. Due to these characteristics we believe that ESMs are beneficial for real-time applications such as games where resources are limited.

Pre-filtered Soft Shadows

We presented a highly efficient soft shadow algorithm that enables rendering of all-frequency shadows at very high framerates. It is based on convolution, which does not require explicit multiple samples and can therefore be carried out in constant time. It is fast enough to render many area light sources simultaneously. We have shown examples where environment map lighting for dynamic objects can be incorporated by decomposing the lighting into a collection of area lights, which are then rendered using our fast soft shadowing technique. The efficiency of our algorithm is in part due to some sacrifices in terms of accuracy. However, our new soft shadow method achieves plausible results, even though they are not entirely physically correct.

9.2 Conclusions

Visibility and shadow evaluation is a long standing problem in the Computer Graphics community. As one of the most expensive, but also most important parts of rendering, fast visibility queries are key to efficient image synthesis. Image-based approaches such as Shadow Mapping are popular for performance and simplicity reasons, however they suffer from aliasing artifacts as do all sampling based approaches. In this dissertation we focused on a particular problem related to Shadow Map filtering and believe that our new linearization process bears many desirable advantages.

The Shadow Map filtering support of our theoretical framework is two-fold. It

provides an efficient constant-time lookup for shadow discontinuities anti-aliasing. Convolution Shadow Maps and Exponential Shadow Maps presented two possible expansion to reconstructing a pre-filtered shadow test, but our framework is general enough to allow for other solutions. We have looked into many possible directions such as the Gompertz growth function, and analytic functions such as the Sigmoid function. These functions would allows single term expansion, however they are unfortunately not separable with respect to the shadow function parameters. Therefore, they can not be used to render pre-filtered shadows.

The second advantage of out theory is that it is general enough to be extended to render plausible high-quality all-frequency shadows. This is a difficult problem especially in dynamic environments with fine details such as trees or even geometries subject to topology changes. We can save valuable computation resources with on our constant-time filtering to improve the quality and speed of percentage-closer based soft rendering.

Due to the linearity of the new filtering process, both categories naturally exploit graphics hardware filtering facilities including mip-mapping and anisotropic filtering.

Bibliography

Agrawala, M., Ramamoorthi, R., Heirich, A., & Moll, L. (2000). Efficient image-based methods for rendering soft shadows. In *Proceedings of SIGGRAPH '00, Computer Graphics* Proceedings, Annual Conference Series, (pp. 375–384). New York, NY, USA: ACM SIGGRAPH. 32, 43, 47

Aila, T., & Akenine-Möller, T. (2004). A hierarchical shadow volume algorithm. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. ACM Press. 46

Aila, T., & Laine, S. (2004). Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004*, (pp. 161–166). Eurographics Association. 42, 43

Annen, T., Dong, Z., Mertens, T., Bekaert, P., Seidel, H.-P., & Kautz, J. (2008a). Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, *27*(3). 8, 93, 102

Annen, T., Mertens, T., Bekaert, P., Seidel, H.-P., & Kautz, J. (2007). Convolution shadow maps. In J. Kautz, & S. Pattanaik (Eds.) *Rendering Techniques 2007*, vol. 18 of *Eurographics / ACM SIGGRAPH Symposium Proceedings*, (pp. 51–60). Eurographics. 7, 63, 96

Annen, T., Mertens, T., Seidel, H.-P., Flerackers, E., & Kautz, J. (2008b). Exponential shadow maps. In C. Shaw, & L. Bartram (Eds.) *Graphics Interface*, ACM International Conference Proceeding Series, (pp. 155–161). ACM. 7, 77

Appel, A. (1968). Some techniques for shading machine renderings of solids. In *Proceedings of the Spring Joint Computer Conference*, (pp. 37–45). 26

Arvo, J. (2007). Alias-free shadow maps using graphics hardware. *Journal of Graphics Tools*, *12*(1), 47–59. 43

Ashdown, I. (1995). Near-Field Photometry: Measuring and Modeling Complex 3-D Light Sources. In *SIGGRAPH '95 Course Notes - Realistic Input for Realistic Images*, (pp. 1–15). ACM. 20

Assarson, U., & Akenine-Möller, T. (2002). Approximate soft shadows on arbitrary surfaces using penumbra wedges. In *Rendering Techniques '02 (Proceedings of the 13th EG Workshop on Rendering*, Springer Computer Science, (pp. 297–306). Eurographics, Eurographics Association. 46

Assarson, U., & Akenine-Möller, T. (2003). A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*. 46, 103

Assarson, U., Dougherty, M., Mounier, M., & Akenine-Möller, T. (2003). An optimized soft shadow volume algorithm with real-time performance. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. ACM Press. 46

Atty, L., Holzschuch, N., Lapierre, M., Hasenfratz, J.-M., Hansen, C., & Sillion, F. (2006). Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum*, *25*(4). 43

Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, (pp. 192–198). New York, NY, USA: ACM. 22

Brabec, S., Annen, T., & Seidel, H.-P. (2003). Practical shadow mapping. *Journal of Graphics Tools*, *7*(4), 9–18. 42, 67

Brabec, S., & Seidel, H.-P. (2002). Single sample soft shadows using depth maps. In *Proceedings of Graphics Interface*. 43, 46

Brabec, S., & Seidel, H.-P. (2003). Shadow volumes on programmable graphics hardware. *Computer Graphics Forum (Proceedings of Eurographics '03)*, *25*(3). 46

Brotman, L. S., & Badler, N. I. (1984). Generating soft shadows with a depth buffer algorithm. In *IEEE Computer Graphics & Applications*, (pp. 71–81). 43

Burt, P. (1981). Fast filter transforms for image processing. *CGIP*, *16*(1), 20–51. 30

Catmull, E. E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.D. thesis, Dept. of CS, U. of Utah. 27, 37

Chan, E., & Durand, F. (2003). Rendering fake soft shadows with smoothies. In *Proceedings of the Eurographics Symposium on Rendering*, (pp. 208–218). Eurographics Association. 43

Chan, E., & Durand, F. (2004). An efficient hybrid shadow rendering algorithm. In *Proceedings of the Eurographics Symposium on Rendering*, (pp. 185–195). Eurographics Association. 47

Chen, S. E., & Williams, L. (1993). View interpolation for image synthesis. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, (pp. 279–288). New York, NY, USA: ACM. 43

Cook, R. L., Porter, T., & Carpenter, L. (1984). Distributed ray tracing. In *Proc. of ACM SIGGRAPH*, (pp. 137–145). Minneapolis, Minnesota. 46

Crow, F. C. (1977). Shadow algorithms for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH '77)*, (pp. 242–248). 5, 45

Crow, F. C. (1984). Summed-area tables for texture mapping. *Computer Graphics (Proc. of SIGGRAPH '84)*, (pp. 207–212). 50, 96

da Vinci, L. (1970). *The Notebooks of Leonardo da Vinci*. Dover. 3, 33

Dachsbacher, C., Stamminger, M., Drettakis, G., & Durand, F. (2007). Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, *26*(3). 48

Debevec, P. E., & Malik, J. (1997). Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, (pp. 369–378). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. 18

Donnelly, W., & Lauritzen, A. (2006). Variance shadow maps. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, (pp. 161–165). New York, NY, USA: ACM Press. 50, 63, 71, 96

Drettakis, G., & Fiume, E. (1994). A fast shadow algorithm for area light sources using backprojection. In *Proceedings of SIGGRAPH '94*, *Computer Graphics* Proceedings, Annual Conference Series, (pp. 223–230). ACM SIGGRAPH. 44

Durand, F. (1999). *3D Visibility: analytical study and applications*. Ph.D. thesis, Université Joseph Fourier, Grenoble I. Http://www-imagis.imag.fr. 36

Everitt, C., & Kilgard, M. J. (2002). Practical and robust stenciled shadow volumes for hardware-accelerated rendering. Tech. rep., NVIDIA Corporation. 46

Fernando, R. (2005). Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, (p. 35). 50, 51, 93, 94, 96, 103

Fernando, R., Fernandez, S., Bala, K., & Greenberg, D. P. (2001). Adaptive shadow maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, Computer Graphics Proceedings, Annual Conference Series, (pp. 387–390). New York, NY, USA: ACM. 42

Glassner, A. S. (1994). *Principles of Digital Image Synthesis*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 22

Gösele, M. (2004). *New Acquisition Techniques for Real Objects and Light Sources in Computer Graphics*. Ph.d. dissertation, Max-Planck-Institut Informatik. 20

Gösele, M., Lensch, H. P. A., Lang, J., Fuchs, C., & Seidel, H.-P. (2004). Disco: acquisition of translucent objects. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, (pp. 835–844). New York, NY, USA: ACM. 23

Greene, N., Kass, M., & Miller, G. (1993). Hierarchical z-buffer visibility. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, (pp. 231–238). New York, NY, USA: ACM. 37

Guennebaud, G., Barthe, L., & Paulin, M. (2006). Real-time Soft Shadow Mapping by Backprojection. In *Rendering Techinques 2006 (Proc. of EGSR)*, (pp. 227–234). 43, 103, 105

Guennebaud, G., Barthe, L., & Paulin, M. (2007). High-quality adaptive soft shadow mapping. *Computer Graphics Forum (Proc. of Eurographics 2007)*, *26*(3). 44

Günther, J., Friedrich, H., Wald, I., Seidel, H.-P., & Slusallek, P. (2006). Ray tracing animated scenes using motion decomposition. *Computer Graphics Forum*, *25*(3), 517–525. Proceedings of Eurographics. 47

Hasenfratz, J.-M., Lapierre, M., Holzschuh, N., & Sillion, F. (2003). A survey of real-time soft shadows algorithms. *Computer Graphics Forum (Proceedings of Eurographics '03)*, *22*(3). 33

Heckbert, P. S. (1989). *Fundamentals of Texture Mapping and Image Warping*. Master's thesis. 49

Heidmann, T. (1991). Real shadows, real time. *Iris Universe*, *18*, 23–31. 46

Hensley, J., Scheuermann, T., Singh, M., & Lastra, A. (2005). Interactive summed-area table generation for glossy environmental reflections. In *ACM SIGGRAPH 2005 Sketches*, (p. 34). 104

Hourcade, J.-C., & Nicolas, A. (1985). Algorithms for antialiased cast shadows. *Computer & Graphics*, (pp. 259–265). 42

Hullin, M. B., Fuchs, M., Ihrke, I., Seidel, H.-P., & Lensch, H. P. A. (2008). Fluorescent immersion range scanning. *ACM Trans. on Graphics (Proc. of SIGGRAPH 2008)*. 23

Im, Y.-H., & Han, C.-Y. (2005). A method to generate soft shadows using a layered depth image and warping. *IEEE Transactions on Visualization and Computer Graphics*, *11*(3), 265–272. 47

Immel, D. S., Cohen, M. F., & Greenberg, D. P. (1986). A radiosity method for non-diffuse environments. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, (pp. 133–142). New York, NY, USA: ACM. 25

Jensen, H. W., & Christensen, P. H. (1998). Efficient simulation of light transport in scences with participating media using photon maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, (pp. 311–320). New York, NY, USA: ACM. 23

Johnson, G. S., Lee, J., Burns, C. A., & Mark, W. R. (2005). The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Trans. Graph.*, *24*(4), 1462–1482. 43

Johnson, G. S., Mark, W. R., & Burns, C. A. (2004). The irregular z-buffer and its application to shadow mapping. Tech. rep. Technical Report TR-04-09. 43

Kajiya, J. T. (1986). The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, (pp. 143–150). New York, NY, USA: ACM. 25

Kaufman, J. E. (1987). *IES Lighting handbook: application volume*. New York: Illuminating Engineering Society of North America. 18

Kautz, J. (2002). *Realistic, Real-Time Shading and Rendering of Objects with Complex Materials*. Ph.d. dissertation, Max-Planck-Institut Informatik. 22

Kautz, J. (2003). *Realistic, Real-Time Shading and Rendering of Objects with Complex Materials*. Göttingen, Germany: Cuvillier. 23

Kautz, J., Lehtinen, J., & Aila, T. (2004). Hemispherical rasterization for self-shadowing of dynamic objects. In *Rendering Techniques 2004 (Proc. of EGSR)*, (pp. 179–184). 48

Keating, B., & Max, N. (1999). Shadow penumbras for complex objects by depth-dependent filtering of multi-layer depth images. In *Rendering Techniques '99 (Proceedings of the 10th EG Workshop on Rendering*, Springer Computer Science, (pp. 205–220). Eurographics, Eurographics Association. 47

Kersten, D., Mamassian, P., & Knill, D. C. (1994). Moving cast chadows and the perception of relative depth. Tech. rep. 30

Kim, T., & Neumann, U. (2001). Opacity shadow maps. In *Proceedings of the 12th Eurographics Workshop on Rendering*, (pp. 177–182). 50

Koenderink, J. J., van Doorn, A. J., & Stavridi, M. (1996). Bidirectional reflection distribution function expressed in terms of surface scattering modes. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, (pp. 28–39). London, UK: Springer-Verlag. 22

Kozlowski, O., & Kautz, J. (2007). Is accurate occlusion of glossy reflections necessary? In *APGV '07: Proceedings of the 4th symposium on Applied perception in graphics and visualization*, (pp. 91–98). New York, NY, USA: ACM. 31

Laine, S., Aila, T., Assarson, U., Lehtinen, J., & Akenine-Möller, T. (2005). Soft shadow volumes for ray tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, (pp. 1156–1165). 48

Lauritzen, A. (2007a). Summed-Area Variance Shadow Maps. In H. Nguyen (Ed.) *GPU Gems 3*. 50, 85, 93, 94, 96

Lauritzen, A. (2007b). Summed-area variance shadow maps. In H. Nguyen (Ed.) *GPU Gems 3*. 51

Lefohn, A., Sengupta, S., Kniss, J., Strzodka, R., & Owens, J. D. (2005). Dynamic adaptive shadow maps on graphics hardware. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, (p. 13). New York, NY, USA: ACM. 42

Lensch, H. P. A. (2003). *Efficient, Image-Based Appearance Acquisition of Real-World Objects*. Ph.d. dissertation, Max-Planck-Institut Informatik. 22

Lensch, H. P. A., Gösele, M., Chuang, Y.-Y., Hawkins, T., Marschner, S., Matusik, W., & Müller, G. (2005). Siggraph course: Realistic materials in computer graphics. 22

Lloyd, D. B. (2007). *Logarithmic Perspective Phadow Maps*. Ph.D. thesis, Chapel Hill, NC, USA. Adviser-Dinesh Manocha. 42

Lokovic, T., & Veach, E. (2000). Deep shadow maps. In *Proceedings of SIGGRAPH '00, Computer Graphics* Proceedings, Annual Conference Series, (pp. 385–392). ACM SIGGRAPH. 50

McCool, M. D. (2000). Shadow rolume reconstruction from depth maps. *ACM Trans. Graph.*, *19*(1), 1–26. 47

McGuire, M., Hughes, J. F., Egan, K., Kilgard, M., & Everitt, C. (2003). Fast, practical and robust shadows. Tech. rep., NVIDIA Corporation, Austin, TX. 46

Mertens, T., Kautz, J., Bekaert, P., & Reeth, F. V. (2004). A self-shadow algorithm for dynamic hair using density clustering. In *Rendering Techniques*, (pp. 173–178). 50

Mertens, T., Kautz, J., Bekaert, P., Reeth, F. V., & Seidel, H.-P. (2003a). Efficient rendering of local subsurface scattering. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, (p. 51). Washington, DC, USA: IEEE Computer Society. 23

Mertens, T., Kautz, J., Bekaert, P., Seidel, H.-P., & Van Reeth, F. (2003b). Interactive rendering of translucent deformable objects. *Eurographics Symposium on Rendering 2003*, *27*(3). 23

Murdoch, J. B. (1981). Inverse square law approximation of illuminance. *Journal of the Illuminating Engineering Society*, *11*(2), 96–106. 20

Nave, C. R. (2006). The hyper physics website. Georgia State University, Department of Physics and Astronomy, Website URL: http://hyperphysics.phy-astr.gsu.edu/hbase/hph.html. 15

Newell, M. E., Newell, R. G., & Sancha, T. L. (1972). A solution to the hidden surface problem. In *ACM'72: Proceedings of the ACM annual conference*, (pp. 443–450). New York, NY, USA: ACM. 44

Ng, R., Ramamoorthi, R., & Hanrahan, P. (2003). All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph.*, *22*(3), 376–381. 48

Nicodemus, F. E. (1965). Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, *4*(7), 767–773. 21, 22

Nishita, T., & Nakamae, E. (1983). Half-tone representation of 3-d objects illuminated by area sources or polyhedron sources. *Proceedings of The IEEE Computer Society' s International Computer Software and Applications Conference (COMPSAC)*, (pp. 237–241). 18, 48

Nishita, T., Okamura, I., & Nakamae, E. (1985). Shading models for point and linear sources. *ACM Transactions on Graphics*, *4*(2), 124–146. 18

Ostromoukhov, V., Donohue, C., & Jodoin, P.-M. (2004). Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.*, *23*(3), 488–495. 102

Parker, S., Shirley, P., & Smits, B. (1998). Single sample soft shadows. Tech. Rep. UUCS-98-019, University of Utah. 46

Plato (1968). *The Republic of Plato*. New York: Basic Books. Translated with notes and an interpretive essay by Allan Bloom. 3

Porter, T., & Duff, T. (1984). Compositing digital images. *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, *18*(3), 253–259. 27

Poulin, P. (1993). *Shading and Inverse Shading from Direct Illumination*. Ph.d. dissertation, University of British Columbia. 20

Reeves, W. T., Salesin, D., & Cook, R. L. (1987). Rendering antialiased shadows with depth maps. *Computer Graphics (Proceedings of SIGGRAPH '87)*, (pp. 283–291). 40, 49, 57, 58, 82

Ren, Z., Wang, R., Snyder, J., Zhou, K., Liu, X., Sun, B., Sloan, P.-P., Bao, H., Peng, Q., & Guo, B. (2006). Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Trans. Graph.*, *25*(3), 977–986. 48

Ritschel, T., Grosch, T., Kautz, J., & Mueller, S. (2007). Interactive illumination with coherent shadow maps. In J. Kautz, & S. Pattanaik (Eds.) *Rendering Techniques 2007*, vol. 18 of *Eurographics / ACM SIGGRAPH Symposium Proceedings*. Eurographics. 44

Ritschel, T., Grosch, T., Kautz, J., & Seidel, H.-P. (2008a). Interactive global illumination based on coherent surface shadow maps. In *GI '08: Proceedings of graphics interface 2008*, (pp. 185–192). Toronto, Ont., Canada, Canada: Canadian Information Processing Society. 44

Ritschel, T., Grosch, T., Kim, M. H., Seidel, H.-P., Dachsbacher, C., & Kautz, J. (2008b). Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA 2008)*, *27*(5), to–appear. 44

Roth, S. D. (1982). Ray Casting for Modeling Solids. *Computer Graphics and Image Processing*, *18*(2), 109–144. 26

Salvi, M. (2008). *Rendering filtered shadows with exponential shadow maps*. In ShaderX 6.0 - Advanced Rendering Techniques. Charles River Media. 50

Schmittler, J., Wald, I., & Slusallek, P. (2002). Saarcor: A hardware architecture for ray tracing. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, (pp. 27–36). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. 47

Schwarz, M., & Stamminger, M. (2007). Bitmask soft shadows. *Computer Graphics Forum (Proc. of Eurographics 2007)*, *26*(3), 515–524. 44, 105

Segal, M., Korobkin, C., van Widenfelt, R., Foran, J., & Haeberli, P. (1992). Fast shadows and lighting effects using texture mapping. *SIGGRAPH Comput. Graph.*, *26*(2), 249–252. 32, 39

Sen, P., Cammarano, M., & Hanrahan, P. (2003). Shadow silhouette maps. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, (pp. 521–526). 42, 47

Sloan, P.-P., Kautz, J., & Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, *21*(3), 527–536. 48, 65

Sloan, P.-P., Luna, B., & Snyder, J. (2005). Local, Deformable Precomputed Radiance Transfer. *ACM Trans. Graph.*, *24*(3), 1216–1224. 48

Soler, C., & Sillion, F. (1998). Fast calculation of soft shadow textures using convolution. In *Proceedings of SIGGRAPH '98*, *Computer Graphics* Proceedings, Annual Conference Series, (pp. 321–332). ACM SIGGRAPH. 50, 57, 93, 94, 96, 103

Stamminger, M., & Drettakis, G. (2002). Perspective shadow maps. *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, (pp. 557–562). 42

Stewart, A. J., & Ghali, S. (1994). Fast computation of shadow boundaries using spatial coherence and backprojections. In *Proc. of SIGGRAPH '94*, (pp. 231–238). 43

Stoichita, V. I. (1997). *A Short History of the Shadow*. Reaktion Books Ltd. 3

Takahashi, T., & Tanaka, T. (1997). Fast analytic shading and shadowing for area light sources. *Computer Graphics Forum (Proceedings of Eurographics '97)*, *16*(3). 48

Timothy J. Purcell, W. R. M., Ian Buck, & Hanrahan, P. (2002). Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, *21*(3), 703–712. 47

Verbeck, C., & Greenberg, D. (1984). A comprehensive light-source description for computer graphics. *IEEE Computer Graphics and Applications*, *4*(7), 66–75. 18

Wald, I. (2004). *Realtime Ray Tracing and Interactive Global Illumination*. Ph.D. thesis, Saarland University, Germany. 47

Wald, I., Benthin, C., & Slusallek, P. (2003). Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proc. of EGSR*, (pp. 74–81). 47

Wald, I., Boulos, S., & Shirley, P. (2007). Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. *ACM TOG*, *26*(1). 47

Wang, Y., & Molnar, S. (1994). Second-depth shadow mapping. Tech. rep., Chapel Hill, NC, USA. 42

Wanger, L. (1992). The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, (pp. 39–42). New York, NY, USA: ACM. 30

Warn, D. R. (1983). Lighting controls for synthetic images. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, (pp. 13–21). New York, NY, USA: ACM. 17

Whitted, T. (1979). An improved illumination model for shaded display. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, (p. 14). New York, NY, USA: ACM. 26, 46

Williams, L. (1978). Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, (pp. 270 – 274). New York, NY, USA: ACM. v, vii, 5, 38

Williams, L. (1983). Pyramidal parametrics. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, (pp. 1–11). New York, NY, USA: ACM Press. 30, 40

Wimmer, M., Scherzer, D., & Purgathofer, W. (2004). Light space perspective shadow maps. In A. Keller, & H. W. Jensen (Eds.) *Rendering Techniques 2004 (Proceedings Eurographics Symposium on Rendering)*, (pp. 143–151). Eurographics Association. 42

Wolff, L. B., Shafer, S. A., & Healey, G. (Eds.) (1992). *Radiometry*. USA: Jones and Bartlett Publishers, Inc. 21, 22

Woo, A., Poulin, P., & Fournier, A. (1990). A survey of shadow algorithms. *IEEE Computer Graphics & Applications*, (pp. 13–32). 33

Wyckoff, C. W., & Feigenbaum, S. A. (1962). An experimental extended exposure response film. *SPIE Newsletter*, (pp. 117–125). 18

Wyman, C., & Hansen, C. (2003). Penumbra maps: Approximate soft shadows in real-time. In *Proceedings of the EG Symposium on Rendering*, Springer Computer Science, (pp. 202–207). Eurographics, Eurographics Association. 43

Xie, F., Tabellion, E., & Pearce, A. (2007). Soft shadows by ray tracing multilayer transparent shadow maps. In *Eurographics Symposium on Rendering*, (pp. 265–276). Grenoble, France: Eurographics Association. 47

Zhang, H. (1998). Forward shadow mapping. In *Rendering Techniques '98 (Proceedings of the 9th EG Workshop on Rendering*, Springer Computer Science, (pp. 131–138). Eurographics, Eurographics Association. 39

Zhou, K., Hu, Y., Lin, S., Guo, B., & Shum, H.-Y. (2005). Precomputed Shadow Fields for Dynamic Scenes. *ACM Trans. Graph.*, *24*(3), 1196–1201. 48

Curriculum Vitae – Lebenslauf

Curriculum Vitae

1976	born in Merzig (Saar), Germany
1983 – 1987	Grundschule am Kreuzberg, Merzig (Saar)
1987 – 1994	Christian-Kretzschmar-RealSchule Merzig (Saar)
1994 – 1996	TGBBZ for Electrical Engineering, Merzig (Saar)
1996 – 1997	Civil Service, German Red Cross, Merzig (Saar)
1997 – 1998	Internship in construction and electrical engineering
1998 – 2002	Study of Computer Science, Saarland University for Applied Science, Saarbrücken, Germany
2002	Bachelor of Science (Diplom-Informatiker (FH), Dipl.-Inf (FH))
2002 – 2004	Study of Computer Science, Saarland University, Saarbrücken, Germany
2004	Master of Science (MSc.)
2004 – 2008	Ph.D. Student at the Max-Planck-Institute Informatik, Saarbrücken, Germany

Lebenslauf

1976	geboren in Merzig (Saar)
1983 – 1987	Grundschule am Kreuzberg, Merzig (Saar)
1987 – 1994	Christian-Kretzschmar-RealSchule Merzig (Saar)
1994 – 1996	TGBBZ für Elektrotechnik, Merzig (Saar)
1996 – 1997	Zivildienst, Deutsches Rotes Kreuz, Merzig (Saar)
1997 – 1998	Praktikum im Bautenschutz und Elektrotechnikbereich
1998 – 2002	Informatikstudium, Hochschule für Technik und Wirtschaft des Saarlandes, Saarbrücken
2002	Abschluss als Diplom-Informatiker (FH) (Dipl.-Inf(FH))
2002 – 2004	Informatikstudium, Universität des Saarlandes, Saarbrücken
2004	Abschluss als Master of Science (MSc.)
2004 – 2008	Promotion am Max-Planck-Institut Informatik, Saarbrücken, Germany