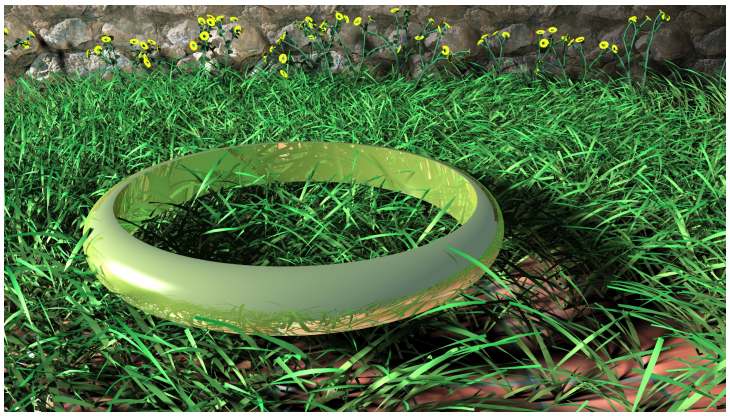
Helsinki University of Technology
Publications in Telecommunications Software and Multimedia
Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian julkaisuja
Espoo 2006 TML-A13

EFFICIENT PHYSICALLY-BASED SHADOW ALGORITHMS

Samuli Laine



TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology
Publications in Telecommunications Software and Multimedia
Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian julkaisuja
Espoo 2006 TML-A13

EFFICIENT PHYSICALLY-BASED SHADOW ALGORITHMS

Samuli Laine

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering, for public examination and debate in Auditorium T2 at Helsinki University of Technology (Espoo, Finland) on the 29th of September, 2006, at 12 noon.

Helsinki University of Technology
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietoliikenneohjelmistojen ja multimedian laboratorio

Distribution:
Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
P.O.Box 5400
FIN-02015 HUT
Finland
Tel. +358-9-451 2870
Fax. +358-9-451 5014
http://www.tml.hut.fi/

Available in PDF format at http://lib.hut.fi/Diss/2006/isbn9512283581/

ISBN 951-22-8357-3 (printed version)
ISSN 1456-7911
ISBN 951-22-8358-1 (electronic version)
ISSN 1455-9722

Otamedia Oy
Espoo 2006

ABSTRACT

Author Samuli Laine
Title Efficient Physically-Based Shadow Algorithms

This research focuses on developing efficient algorithms for computing shadows in computer-generated images. A distinctive feature of the shadow algorithms presented in this thesis is that they produce correct, physically-based results, instead of giving approximations whose quality is often hard to ensure or evaluate.

Light sources that are modeled as points without any spatial extent produce hard shadows with sharp boundaries. Shadow mapping is a traditional method for rendering such shadows. A shadow map is a depth buffer computed from the scene, using a point light source as the viewpoint. The finite resolution of the shadow map requires that its contents are resampled when determining the shadows on visible surfaces. This causes various artifacts such as incorrect self-shadowing and jagged shadow boundaries. A novel method is presented that avoids the resampling step, and provides exact shadows for every point visible in the image.

The shadow volume algorithm is another commonly used algorithm for real-time rendering of hard shadows. This algorithm gives exact results and does not suffer from any resampling problems, but it tends to consume a lot of fillrate, which leads to performance problems. This thesis presents a new technique for locally choosing between two previous shadow volume algorithms with different performance characteristics. A simple criterion for making the local choices is shown to yield better performance than using either of the algorithms alone.

Light sources with nonzero spatial extent give rise to soft shadows with smooth boundaries. A novel method is presented that transposes the classical processing order for soft shadow computation in offline rendering. Instead of casting shadow rays, the algorithm first conceptually collects every ray that would need to be cast, and then processes the shadow-casting primitives one by one, hierarchically finding the rays that are blocked.

Another new soft shadow algorithm takes a different point of view into computing the shadows. Only the silhouettes of the shadow casters are used for determining the shadows, and an unintrusive execution model makes the algorithm practical for production use in offline rendering.

The proposed techniques accelerate the computing of physically-based shadows in real-time and offline rendering. These improvements make it possible to use correct, physically-based shadows in a broad range of scenes that previous methods cannot handle efficiently enough.

UDC 004.925, 004.383.5
Keywords computer graphics, shadow algorithms, soft shadows, physically-based rendering

TIIVISTELMÄ

Tekijä Samuli Laine
Työn nimi Efficient Physically-Based Shadow Algorithms

Tämä tutkimus käsittelee tehokkaita algoritmeja varjojen laskemiseksi tietokonegrafiikassa. Väitöskirjassa esitettyjen algoritmien keskeinen piirre on niiden tuottamien varjojen fysikaalisesti perusteltavissa oleva oikeellisuus. Tämä erottaa ne approksimatiivisista algoritmeista, joiden tuottamien varjojen laatua on usein vaikea arvioida tai taata.

Pistemäiset valonlähteet tuottavat ns. kovia varjoja, joiden reunat ovat teräviä. Varjokartta-algoritmi (shadow mapping) on perinteinen menetelmä tämänkaltaisten varjojen piirtämiseen. Varjokartta on valonlähteestä katsoen laskettu syvyyspuskuri. Koska varjokartan resoluutio on äärellinen, sen uudelleennäytteistys varjokyselyvaiheessa aiheuttaa monenlaisia virheitä kuvaan, kuten pintojen virheellistä itsevarjostusta ja sahalaitaisia varjoreunoja. Tässä työssä esitellään uusi menetelmä, joka välttää uudelleennäytteistysvaiheen ja tuottaa siten tarkat varjot kaikkiin kuvassa näkyviin pisteisiin.

Varjotilavuusalgoritmi (shadow volumes) on toinen usein käytetty menetelmä kovien varjojen reaaliaikaiseen piirtämiseen. Se tuottaa tarkat varjot ja välttää uudelleennäytteistyksestä aiheutuvat ongelmat, mutta vaatii paljon laskentaresursseja, mikä puolestaan johtaa tehokkuusongelmiin. Tässä työssä esitellään uusi tekniikka, joka valitsee kuva-avaruudessa paikallisesti kahden aikaisemman varjotilavuusmenetelmän käytön välillä. Kun valitaan aina paikallisesti tehokkaammaksi arvioitu menetelmä, saavutetaan parempi tehokkuus kuin käyttämällä kumpaakaan menetelmää yksinään.

Ei-pistemäiset valonlähteet tuottavat pehmeäreunaisia varjoja. Työssä esitellään uusi algoritmi, joka kääntää perinteisen pehmeiden varjojen laskentamenetelmän järjestyksen päinvastaiseksi. Sen sijaan, että tarkasteltaisiin yksittäisten varjosäteiden näkyvyyttä, algoritmi kokoaa ensin tiedon kaikista tarvittavista varjosäteistä, minkä jälkeen varjostavat primitiivit käsitellään yksitellen ja haetaan niiden peittämät varjosäteet.

Toinen esiteltävä pehmeiden varjojen laskenta-algoritmi laskee varjot eri tavalla. Vain varjostavien kappaleiden siluetteja käytetään varjojen laskentaan, ja käytännöllinen laskentajärjestys mahdollistaa algoritmin helpon soveltamisen osana käytännön kuvanlaskentajärjestelmiä.

Esitellyt tekniikat nopeuttavat fysikaalisesti oikeiden varjojen laskentaa sekä reaaliaikaisissa että ei-reaaliaikaisissa sovelluksissa. Tämä mahdollistaa tarkkojen, fysikaalisesti oikeiden varjojen käytön useissa tilanteissa, joissa aiempien menetelmien tehokkuus ei ole riittävä.

UDC 004.925, 004.383.5
Avainsanat tietokonegrafiikka, varjoalgoritmit, pehmeät varjot, fysiikkapohjainen kuvanlaskenta

Tiinalle,
Santerille ja
vauvalle

PREFACE

This research was carried out at the Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, Espoo, during 2004–2006.

I want to express my gratitude to Prof. Lauri Savioja, the supervisor of this thesis, for support and encouragement, and for giving me considerable freedom in carrying out my research activities.

I also want to thank Dr. Timo Aila, the instructor of this thesis. Being able to follow his dynamic and to-the-point way of doing science has been a great asset to me, and collaborating with him has always been a pleasant experience.

In addition, I would like to thank my other co-authors Ulf Assarsson, Jaakko Lehtinen, and Tomas Akenine-Möller for fruitful collaboration, and my colleagues Janne Kontkanen and Jussi Räsänen for insightful conversations. Many thanks to the good people at Hybrid Graphics for selflessly sharing their expertise back in the days when I entered the field of computer graphics. Of those, I want to especially thank Ville Miettinen who showed the importance of vigor, in addition to the usual rigor, when tackling tricky problems.

Furthermore, I want to thank the pre-examiners of this thesis, Dr. Jukka Arvo and Dr. Xavier Décoret, for positive feedback and constructive comments.

I am grateful to the organizations that have supported this work financially: Tekes (The National Technology Agency), Anima Vitae, Bitboys, Hybrid Graphics, Nokia, and Remedy Entertainment.

Finally, I am most deeply indebted to my family. Tiina, this work would not have been possible without your love, support, and apparently unlimited patience throughout my studies. Santeri, your endless curiosity and cheerful attitude towards life always brighten up my day. Thank you for reminding me of things that really matter.

Otaniemi, Espoo, 29th August 2006

Samuli Laine

TABLE OF CONTENTS

LIST OF PUBLICATIONS

This thesis summarizes the following articles and publications, referred to as [P1]–[P5]:

[P1] T. Aila and S. Laine. Alias-Free Shadow Maps. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)*, pages 161–166. Eurographics Association, 2004.

[P2] S. Laine and T. Aila. Hierarchical Penumbra Casting. *Computer Graphics Forum*, 24(3):313–322, 2005.

[P3] S. Laine. Split-Plane Shadow Volumes. In *Graphics Hardware 2005 (Eurographics Symposium Proceedings)*, pages 23–32. Eurographics Association, 2005.

[P4] S. Laine, T. Aila, U. Assarsson, J. Lehtinen and T. Akenine-Möller. Soft Shadow Volumes for Ray Tracing. *ACM Transactions on Graphics*, 24(3):1156–1165, 2005.

[P5] J. Lehtinen, S. Laine and T. Aila. An Improved Physically-Based Soft Shadow Volume Algorithm. *Computer Graphics Forum*, 25(3): 303–312, 2006.

LIST OF ABBREVIATIONS

2D	Two-dimensional
3D	Three-dimensional
AFSM	Alias-free shadow maps [P1]
CPU	Central processing unit
GPU	Graphics processing unit
HPC	Hierarchical penumbra casting [P2]
SPSV	Split-plane shadow volumes [P3]
SSV	Soft shadow volumes [P4, P5]

1 INTRODUCTION

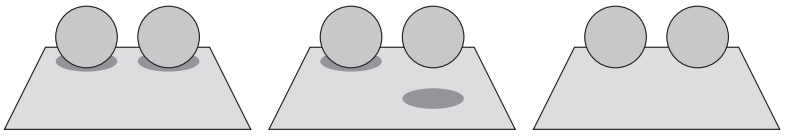
The topic of this thesis is computing realistic shadows in computer graphics. It is a well-known fact that shadows are an important feature in computer-generated images, as they increase the perceived level of realism and also convey information about spatial relationships between objects. An example of the latter feature is illustrated in Figure 1.1.

When rendering an image, we need to compute the shadows at some number of 3D points, which we will call *receiver points* throughout this thesis. Figure 1.2 illustrates a low-resolution image of a simple scene, where one receiver point is deposited at surface points that are visible through the centers of pixels. The 2D points on the image are distributed into a uniform lattice, since that is the shape of the pixel grid, but their respective 3D positions are not uniformly distributed.

To limit the scope of this thesis, we will consider only shadows of *direct illumination*, i.e. light that arrives at receiver points directly from the light sources. In particular, we shall not consider so-called *global illumination* effects where multiple scattering events are allowed for a single light path. These two illumination models are illustrated in Figure 1.3. It is often beneficial to solve global and direct illumination separately, since efficient solution methods are considerably different in each case.

A receiver point is in shadow when the flow of light from a light source to the receiver point is completely or partially obstructed. We differentiate between two types of shadows that result from using different types of light sources. *Hard shadows* appear when the light source is modeled as a point without any spatial extent. For this kind of light source, any given receiver point is either in full shadow or not in shadow at all, which yields a sharp shadow boundary (Figure 1.4a). Point light sources do not exist in physical world, but in computer graphics they can be used as approximations for small light sources.

If the light source has some kind of spatial extent, being e.g. a line segment, a polygon, or a volumetric entity, the receiver points may be not only completely shadowed or completely lit, but also partially shadowed. Partial shadowing occurs when light from some portion of the light source flows to a receiver point without being occluded, while light coming from the rest



(a) (b) (c)

Figure 1.1: The canonical example that illustrates the importance of shadows as spatial cues. (a) and (b) show obviously distinct geometrical situations. Note that only the location of shadows differ in these figures. (c) In absence of shadows, these two configurations would look identical.

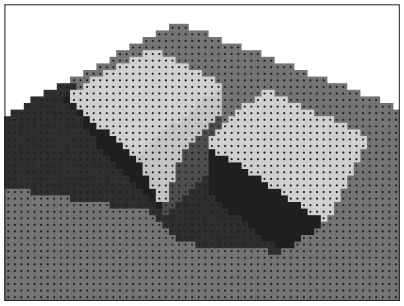
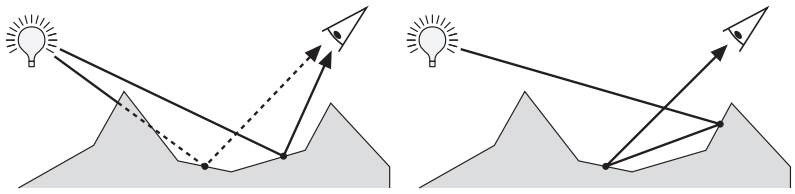
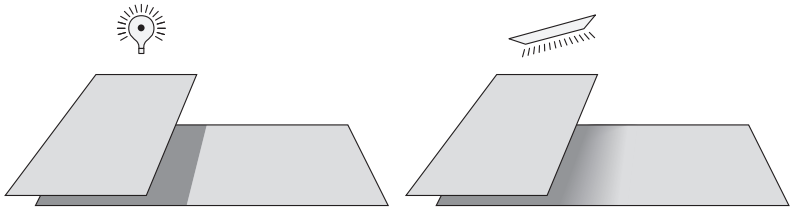


Figure 1.2: Receiver points in a simple low-resolution image where no antialiasing has been used. Every pixel specifies a 3D receiver point, except when the center of the pixel does not map to any surface in the scene. Note that even though the receiver points are uniformly distributed in the image plane, their distribution in 3D may be highly irregular. In this image, hard shadows have been computed from a point light source.



(a) direct illumination (b) global illumination

Figure 1.3: Direct versus global illumination. (a) The shadow algorithms presented in this thesis consider only direct illumination paths, two of which are shown in the figure. Each path scatters exactly once off a visible surface. (b) Global illumination paths may scatter multiple times in the scene before reaching the viewer. A single global illumination path with two scattering events is shown.



(a) hard shadow (b) soft shadow

Figure 1.4: Hard shadows versus soft shadows. (a) The light source is modeled as a point with no spatial extent. Consequently, every point on the bottom surface is either completely shadowed or completely lit, yielding sharp shadow boundaries. (b) The light source is modeled as a polygon. Now a point on the bottom surface may also be partially shadowed, giving rise to a soft shadow boundary.

	Hard shadows	Soft shadows
Real-time rendering	SPSV	
Offline rendering	AFSM	HPC, SSV

Figure 1.5: Classification of the algorithms presented in this thesis.

of the light source is blocked. This yields smooth shadow boundaries, i.e. *soft shadows* (Figure 1.4b). It has been shown [58] that the smoothness of shadows is an essential ingredient in realistic images, and hard shadows are perceived as artificial and unrealistic. Since one-dimensional light sources are scarcely used in practice, we will from now on assume that any non-point light source has nonzero surface area. These kind of light sources are commonly called *area light sources*, and we shall adopt this term in this thesis as well.

1.1 Scope of This Thesis

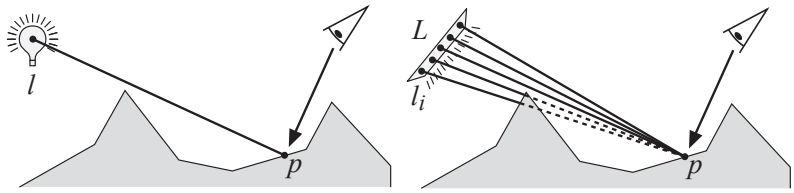
This thesis concentrates on the computation of hard and soft shadows of direct illumination. The novel algorithms presented have been designed to improve the performance of shadow computation over previous methods. The improvements have been achieved by formulating the shadow computation process in various different ways, and then constructing algorithms that exploit hierarchical methods for performing the computation efficiently.

This thesis is based on five publications [P1–P5] that propose new, efficient algorithms for shadow computation. Table 1.1 summarizes the names of the algorithms that are introduced in the publications, as well as the abbreviations that will be used for referring to them.

Figure 1.5 shows the classification of the algorithms. Three of the algorithms (AFSM, HPC and SSV) are designed to be used in offline rendering, and they are based entirely on software solutions. On the other hand, the SPSV algorithm is based on a proposed GPU extension which would allow the algorithm to accelerate the rendering of hard shadows. Both SPSV and AFSM algorithms are designed for computing hard shadows only, while HPC and SSV are soft shadow algorithms.

Algorithm	**Abbreviation**	**Publications**
Alias-free shadow maps	AFSM	[P1]
Hierarchical penumbra casting	HPC	[P2]
Split-plane shadow volumes	SPSV	[P3]
Soft shadow volumes	SSV	[P4, P5]

Table 1.1: List of the algorithms presented in this thesis, the abbreviations used, and the publications in which they have appeared.



(a) point light source (b) area light source

Figure 1.6: Visibility relations. (a) For a point light source, we need to compute the status of one visibility relation per receiver point and light source. In this case, the status of the relation shown is VISIBLE, as there is free line of sight between receiver point p and light source l. (b) An area light source L is represented by a number of point-like samples l_i, where i is the index of the light sample. If there are M such light samples, we need to solve the status of M visibility relations for each receiver point. In this figure, five visibility relations are shown, two that are blocked and three that are visible. In practice, dozens or hundreds of light samples are required for obtaining noise-free shadows.

1.2 Exact Hard Shadows

A desirable property for a shadow algorithm, as for any other algorithm, is that the computed results are correct. With hard shadows, this means that whether a receiver point is in shadow or not is determined exactly. With standard shadow mapping [75], this is unfortunately impossible, since the discretization of the shadow map prevents us from getting exact answers for arbitrary receiver points. Shadow volumes [26, 40], on the other hand, produce exact results, but tend to consume a lot of GPU resources. Shadow maps and shadow volumes will be discussed in further detail in Chapter 3.

To consider the process of computing the shadows for an entire image, we now introduce the concept of *visibility relation*. Each visibility relation is defined by two 3D points, one that is a receiver point, and one that lies on the light source. The status of a visibility relation may be either BLOCKED or VISIBLE, depending on whether some surface in the scene blocks the visibility between the endpoints or not, respectively. Here, we shall not consider the possibility of a single visibility relation being partially blocked due to e.g. semi-transparent surfaces, but require that every visibility relation has only two possible statuses. The possibility of supporting partially blocking surfaces is discussed separately in the publications.

Let us consider a single point light source and a single receiver point. To calculate whether the receiver point is illuminated by the light source, we need to evaluate the status of a visibility relation between the two points (Figure 1.6a). If the status is BLOCKED, there is a surface that blocks the visibility from the light source to the receiver point, and consequently also blocks the flow of light, resulting in a shadow. On the other hand, if the status of the visibility relation is VISIBLE, there is free line of sight between the points, and therefore light from the light source can reach the receiver point, yielding no shadow. Thus, for N receiver points and a single point

light source, there are exactly N visibility relations that need to be solved for computing all shadows in the scene.

Restrictions posed by current GPU architectures

For real-time applications, the current structure of the available GPUs mostly dictates the nature of the algorithms that can be run efficiently. The most limiting factor in hardware-based shadow computation is that usually no global knowledge of all geometry in the scene can be made available efficiently. This renders many algorithms such as casting shadow rays [6, 74] impossible to be implemented efficiently on hardware.

The AFSM algorithm [P1] in this thesis modifies the shadow mapping algorithm [75] so that it is able to produce exact shadows, but with the cost that it cannot run entirely on current hardware. Since shadow maps are also heavily used in offline production rendering, the CPU-based algorithm is still widely applicable in practice.

Developments in GPU architectures may enable running AFSM entirely on hardware in the future, which would make it a possible real-time algorithm for computing hard shadows. Recently, a multi-pass hardware implementation of the AFSM algorithm has been proposed by Arvo [10]. Unfortunately, the implementation requires reading a single value from the GPU to CPU, which currently limits its framerate below real-time applications.

To reduce the inefficiency in shadow volume rendering, the SPSV algorithm [P3] employs a hierarchical approach that avoids much of unnecessary pixel processing in the GPU. It also cannot be run efficiently on current GPUs, but only relatively minor extensions to the hardware would be required.

1.3 Physically-Based Soft Shadows

With soft shadows, it is generally not possible to require correct results, since the appearance of a receiver point, as seen from the camera, depends on a complicated integral that does not necessarily have a closed-form solution. This issue will be discussed further in Chapter 2. Instead of attempting to evaluate the integrals analytically, Monte Carlo integration techniques are needed for obtaining an estimate of the value of the integral.

We define a *physically-based* soft shadow algorithm as one that can be used for obtaining an unbiased estimate of the appearance of a receiver point. In other words, the estimated appearance must converge towards the correct result as the accuracy of the computation is increased. In contrast to physically-based algorithms, there are numerous *approximative* soft shadow algorithms that do not give unbiased results but instead approximate the shadows by e.g. blurring the shadow boundaries.

To perform Monte Carlo integration, area light sources must be represented using a number of point-like *light samples* in order to compute the potential partial shadowing, as illustrated in Figure 1.6b. If the light source is sampled using M light samples and there are N receiver points in the image, a total of $N \times M$ visibility relations need to be solved. Obviously, this requires much more computation than with hard shadows, where N

visibility relations were sufficient.

Efficient solving of visibility relations
The central question of physically-based shadow algorithms is how to solve the status of the visibility relations in an efficient manner. The simplest solution is to consider every visibility relation separately, but this easily leads to prohibitively long execution times. For a moderate-resolution image with, say, one million pixels, and a single area light source represented using 200 light samples, there are already two hundred million visibility relations to be solved.

The crucial idea in modern soft shadow algorithms is to exploit the coherence in the status of nearby visibility relations. Light samples that are packed closely together tend to contribute to the resulting image mostly in a similar fashion, and consequently, nearby receiver points are usually illuminated by approximately the same set of light samples. Of the algorithms included in this thesis, the SSV algorithm [P4, P5] exploits the coherence between nearby light samples and neighboring triangles, and the HPC algorithm [P2] takes advantage of the coherence in both light samples and receiver points, while processing each triangle separately.

1.4 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 formulates a mathematical framework for computing physically-based shadows, and Chapter 3 gives an overview of related research on shadow algorithms. Chapter 4 presents the new contributions to physically-based shadow computation. Finally, Chapter 5 summarizes the main results of the thesis and the author's contributions.

2 MATHEMATICAL BACKGROUND FOR PHYSICALLY-BASED SOFT SHADOWS

In this chapter, we will present the mathematical foundations behind physically-based soft shadow computation. As was stated in Chapter 1, we define a physically-based soft shadow algorithm as one whose results converge towards the correct solution as the accuracy of the computation is increased. While this might appear to be a natural requirement for any algorithm, it is not satisfied by most of the previously presented soft shadow algorithms.

We first introduce a number of key terms and their definitions, and then analyze how physically-based soft shadows are computed for direct illumination from area light sources.

2.1 Terminology

Receiver points

When computing shadows, we have a set of points where we are interested in knowing the shadows. These are the *receiver points*, denoted r_i, where i is an index. The receiver points are typically determined by tracing rays from the camera through the pixels of the image. Rasterization can also be used. Multiple receiver points may be needed for each pixel if e.g. antialiasing is performed.

Light samples

Since we use Monte Carlo methods for estimating the appearance of the receiver points, which includes evaluating the visibility between the receiver points and the light sources, we need to represent the light sources as a set of *light samples*, denoted l_j, where j is an index. The light samples are distributed on the surfaces of the light sources using some suitable distribution. The number of light samples to be used is a quality parameter: approximating a light source with too few samples results in visible noise, but as the number of samples increases, the amplitude of the noise decreases, and the appearance of the shadows converges to the correct solution.

Visibility function

Given the assumption that objects always block rays of light either completely or not at all, visibility between two points is a binary function. We may formalize it as follows: given two 3D points p_a and p_b, we define function $\mathcal{V}(p_a, p_b)$ so that $\mathcal{V}(p_a, p_b) = 1$ if no surfaces intersect the line segment $p_a \rightarrow p_b$, excluding the endpoints, and $\mathcal{V}(p_a, p_b) = 0$ if at least one surface intersects the line segment. We also define $\mathcal{V}(p_a, p_b) = 1$ when $p_a = p_b$. For treating partially transparent surfaces, the visibility function would need to assume values other than 0 and 1, and such cases are discussed separately in the publications.

Visibility relations

Given receiver points r_i and light samples l_j, the shadow computation task can be stated as determining $\mathcal{V}(r_i, l_j)$ for every i, j pair. A single i, j pair defines one *visibility relation*, and if there are N receiver points and M light samples, there are $N \times M$ visibility relations in total. The status of a visibility relation may be either VISIBLE or BLOCKED, corresponding to whether $\mathcal{V}(r_i, l_j) = 1$ or $\mathcal{V}(r_i, l_j) = 0$, respectively. For some algorithms, the light samples must remain stationary throughout the computation, whereas some algorithms allow positioning the light samples separately for each receiver point.

2.2 Physically-Based Soft Shadows

Soft shadows appear when the light source is not considered to be a point, but as an object or surface with nonzero area. Here, we omit the rare (but existing [57]) case of light sources that are shaped as line segments or other one-dimensional curves, as they are not used in practice, and extending the analysis to such cases is trivial. In their most general form, area light sources are objects that emit light, although most often simple planar and polygonal light sources are used. It should be noted that even a volumetric light source can be expressed using only its bounding surface as a light source, assuming that no shadow caster or shadow-receiving surface is located inside the bounding surface.

Soft shadows are fundamentally more difficult to compute than hard shadows, since the visibility of an area light source to a point being shaded cannot be expressed simply as 0 or 1. As it will turn out, a scalar value that can assume values between 0 and 1 does not suffice either. We start developing the equation for shadow computation from the classical *rendering equation* first formulated by Kajiya in 1986 [45]. The rendering equation governs the light transport when full global illumination is taken into account. Using slightly more modern notation than what was used in the original paper, we write the rendering equation as follows:

$$u(p \rightarrow \omega_{out}) = e(p \rightarrow \omega_{out}) + \int_{\Omega_p} f_r(p, \omega_{in} \rightarrow \omega_{out}) u(p \leftarrow \omega_{in}) \lfloor \mathbf{n}_p \cdot \omega_{in} \rfloor \, d\omega_{in}. \tag{2.1}$$

Here, $u(p \rightarrow \omega_{out})$ denotes the total radiance flowing from point p into outgoing direction ω_{out}, and $e(p \rightarrow \omega_{out})$ is the radiance emitted from point p towards ω_{out}. Unless p is on a surface of a light source, the emission term $e(p \rightarrow \omega_{out})$ is zero. The integral is taken over the hemisphere Ω_p that is directed towards the normal vector $\mathbf{n}_p$ at p. Function $f_r(p, \omega_{in} \rightarrow \omega_{out})$ is the BRDF [53] of the surface at p, and it tells how big portion of incident light coming from direction ω_{in} to point p is reflected towards outgoing direction ω_{out}.[1] Expression $\lfloor \cdot \rfloor$, commonly used in computer graphics texts, denotes the "clamp-to-zero" function $\max(\cdot, 0)$.

[1] The BRDF (bidirectional reflectance distribution function) of a surface depends on its material, and there are many models for expressing it in analytical forms. Tabulated BRDFs that are measured from real materials are also becoming popular. We will not discuss BRDFs further in this thesis, and merely note their vital role in creating realistic images.

As can be seen, the radiance function u appears both on the left hand side of the equation and also inside the integral, making this a so-called Fredholm equation of the second kind. Since our goal here is to compute direct illumination only, we must replace the radiance function u inside the integral by emission function e, which greatly simplifies the situation.

In addition, we make a somewhat surprising twist at this point, and remove the emission term $e(p \rightarrow \omega_{out})$ outside the integral. This makes the light sources invisible entities that emit light while being invisible to the camera. This is often desirable, and many commercial graphics packages allow treating area light sources exactly this way. The invisibility of light sources gives the artists freedom to e.g. place additional light sources between the camera and the objects visible in the scene, which is impossible in physical world without showing the light sources in the image. If, on the other hand, it is desirable to see the light sources in the image, it is always possible to present the corresponding surfaces twice, once as a light source, and once as plain geometry that may have an emissive material, but that does not act as a light source. We thus get the following equation:

$$u(p \rightarrow \omega_{out}) = \int_{\Omega_p} f_r(p, \omega_{in} \rightarrow \omega_{out}) e(p \leftarrow \omega_{in}) \lfloor \mathbf{n}_p \cdot \omega_{in} \rfloor \, d\omega_{in}. \quad (2.2)$$

Applying this integral directly when computing soft shadows from direct illumination would be impractical. This is mainly because $e(p \leftarrow \omega_{in})$ inside the integral is zero unless a light source is visible from p in direction ω_{in}. We may recast this equation into an area-based formulation [29] by a change of integration variable. Assuming no participating media, the incoming radiance at p from direction ω_{in} is the same as outgoing radiance from point l in direction $-\omega_{in}$, where l is a point on a light source visible from p in direction ω_{in}. This allows us to state that $e(p \leftarrow \omega_{in}) = e(l \rightarrow -\omega_{in})$. The differential solid angle $d\omega_{in}$ can be expressed in terms of differential area as follows [29]:

$$d\omega_{in} = \lfloor \mathbf{n}_l \cdot -\omega_{in} \rfloor \frac{dA}{|l - p|^2}. \quad (2.3)$$

Because of the visibility criterion, we must now multiply the kernel of the integral by the visibility function $\mathcal{V}(p, l)$ to take potential occlusion into account. This yields the following equation, where we integrate over the surface L of a light source:

$$u(p \rightarrow \omega_{out}) = \int_L f_r(p, \omega_{in} \rightarrow \omega_{out}) e(l \rightarrow -\omega_{in}) \times \mathcal{V}(p, l) \frac{\lfloor \mathbf{n}_p \cdot \omega_{in} \rfloor \lfloor \mathbf{n}_l \cdot -\omega_{in} \rfloor}{|l - p|^2} \, dA_l. \quad (2.4)$$

It is important to note that ω_{in} now depends on the integration variable l and is not constant. Dissecting the equation, we see that there are three functions inside the integral: BRDF, emission function and visibility function. Furthermore, there are factors that account for the geometry of the situation: $\lfloor \mathbf{n}_p \cdot \omega_{in} \rfloor$ converts the cross-sectional flow from the direction of ω_{in} to flow per surface area at p, and $\lfloor \mathbf{n}_l \cdot -\omega_{in} \rfloor$ does the same in reverse

direction at the surface of the light source. Attenuation factor $1/|l-p|^2$ accounts for the distance between p and l.

Solving the integral in Equation 2.4 analytically is possible for certain types of BRDFs and visibility functions, but even in these situations, it is much easier to solve the integral using Monte Carlo sampling instead. Integral over an arbitrary function can be approximated by constructing a number of uniform-density sampling points in the domain of integration, and averaging the values of the function in these locations:

$$\int_D f(x)\,dx \approx \frac{1}{M}\sum_{j=0}^{M-1} f(x_j), \tag{2.5}$$

where x_j $(0 \leq j < M)$ are the sampling points placed in the domain D that we integrate over. If we generate M uniform-density samples l_j on the surface of the light source, we can thus approximate Equation 2.4 with the following sum:

$$u(p \to \omega_{out}) \approx \frac{1}{M}\sum_{j=0}^{M-1} \Big[f_r(p, \omega_{in} \to \omega_{out}) e(l_j \to -\omega_{in}) \times \mathcal{V}(p, l_j) \frac{\lfloor \mathbf{n}_p \cdot \omega_{in} \rfloor \lfloor \mathbf{n}_{l_j} \cdot -\omega_{in} \rfloor}{|l_j - p|^2} \Big]. \tag{2.6}$$

In this formulation, computation is easy since we sample all the difficult functions in discrete points only. As sample count M is increased, the approximation converges to the correct result, and if M is too small, the approximation errors result in visible noise in the shadows. A typical value for M would be around 200 in situations where a single, reasonably small light source illuminates the scene, provided that the light samples l_j are generated using a good low-discrepancy distribution.

The most difficult part in applying Equation 2.6 is computing the values of the visibility function $\mathcal{V}(p, l_j)$, as it depends on the entire scene, in contrast to BRDF and emission functions that are purely local in nature. The classical method used in stochastic ray tracing [25] is to cast a separate shadow ray for each l_j, which inevitably gives linear performance with respect to the number of light samples M, regardless of the computational complexity of casting a single shadow ray. Modern approaches attempt to solve $\mathcal{V}(p, l_j)$ for multiple j at once, and possible for multiple p also. This applies to the soft shadow algorithms presented in this thesis as well.

2.3 Some Remarks on Importance Sampling

When the integrand in Equation 2.4 is not constant over the domain of integration, it is generally beneficial to distribute the light samples non-uniformly on the surface of the light source in order to reduce the variance in the estimated value of the integral. This technique is commonly called *importance sampling* [64] in computer graphics. Any distribution of light samples can be used for producing non-biased results, provided that the contributions from the individual light samples are appropriately normalized according to the probability density function (PDF) from which the sampling points are drawn.

Theoretically, the optimal distribution would be obtained by using the integrand as the probability density function, but for normalization purposes we would need to know its integral, which is of course the unknown quantity being estimated. The Metropolis algorithm, brought to computer graphics by Veach [72], automatically samples the integrand according to the correct PDF by mutating the sampling point using application-specific strategies. A drawback is that each sample must be evaluated before generating the next one, which prevents exploiting the coherence between sampling points.

Instead of attempting to use the integrand directly as the PDF, we may use a PDF that can be assumed to mimic the overall behavior of the function being integrated over. One possibility would be using e.g. the product of the BRDF, emission and geometry terms, and thus leave only the visibility function to be sampled. The optimal distribution of light samples generally varies between receiver points, and re-computing the distributions for each receiver point may be too costly. In addition, some soft shadow algorithms do not support using different light sample distributions for every receiver point. Of the algorithms presented in this thesis, the SSV algorithm [P4] allows using different light sample distributions for each receiver point, whereas the HPC algorithm [P2] requires the distribution to remain fixed. Even if the light sample distribution needs to be fixed, we may construct it by importance sampling the emission function, which is useful if the emission is not constant over the surface of the light source.

3 RELATED RESEARCH

In this section, we review previous work on shadow algorithms. Our main focus is on exact and physically-based shadow algorithms that approximate the visibility only by sampling, thereby guaranteeing the convergence towards correct result as the number of samples is increased. Approximative soft shadow algorithms will be discussed only quite briefly, as they do not fall into the scope of this thesis, and comprehensive surveys are available elsewhere (e.g. [37, 9]).

We first examine the shadow map algorithm [75] and shadow volumes [26] and their extensions and variants. Other hard shadow algorithms are then discussed briefly. After this, we investigate the various approximations used in approximative soft shadow algorithms, and briefly discuss such algorithms that resemble the physically-based algorithms presented in this thesis. Finally, we review previous work on physically-based soft shadow computation.

3.1 Shadow Maps

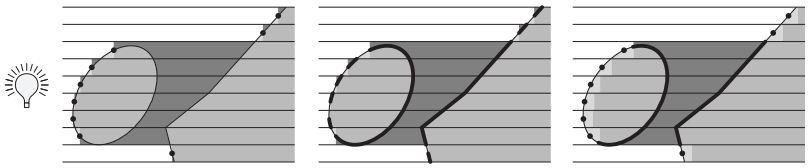
The use of *shadow maps* [75] is a common method for real-time hard shadow rendering, and it is also often used in offline production rendering due to its versatility and simplicity. A shadow map is an intermediate data structure, essentially a depth buffer, that is constructed by rendering the scene using the point light source as the viewpoint. After the shadow map has been constructed, it can be used for answering *shadow queries* for arbitrary points in the scene.

To determine if a world-space point p_W is in shadow, it is transformed into the image space of the shadow map, and the depth of transformed point p_T is compared against the value stored in the shadow map. If the depth of the point is greater than the value stored in the map, the point is in shadow, since there is a surface closer to the light source, i.e. between point p and the light source. Arvo and Aila [11] present a simple technique for avoiding the shadow map lookups for world-space points that get projected outside the shadow map.

Shadow mapping is an approximative shadow algorithm because of the discretization of the intermediate data structure, i.e. the shadow map. In theory, hard shadows computed using shadow maps converge towards the correct result if the resolution of the shadow map is allowed to increase without bounds. Unfortunately, no finite resolution can guarantee correct shadows.

The resampling problem of shadow maps

The finite resolution of the shadow map introduces a fundamental problem in performing the shadow queries. This property was called the *resolution mismatch* problem in Aila's PhD thesis [2], although the problem stems more directly from the resampling of the shadow map instead of the resolution alone. With a finite-resolution shadow map, the transformed



(a) (b) (c)

Figure 3.1: A 2D analogue of the discretization in shadow mapping. A directional light is illuminating the scene from the left, and the horizontal lines separate the pixels in a 1D shadow map. The position of the viewer is unimportant. (a) The depth values are evaluated at the centers of pixels. Based on the depth values, the darkened region seems to be in shadow. (b) The surface points that are in shadow are highlighted with thick lines. Because no bias is used, a lot of incorrect self-shadowing can be seen on illuminated surfaces. (c) With suitable bias, the shadows are effectively pushed away from the shadow casters, yielding mostly correct shadows in this case. Note that with more acutely sloped surfaces, a larger bias would be needed.

world-space query points do not land at the centers of the pixels in the shadow map, where the depth is correct. Instead, the depth value read from the shadow map is generally something else than the actual depth at the query point, which gives rise to various artifacts.

Even a simple planar surface may end up shadowing itself, if the depth values fetched from the shadow map are used as-is. This kind of self-shadowing is sometimes called "surface acne", and it shows up as distracting patterns of false shadows (Figures 3.1a and 3.1b). The originally proposed [75] and still most commonly used way to fight this problem is to introduce an artificial bias value into the lookup, effectively pushing the shadows away from the light source. Note that some bias is always needed in order to account for the quantization of the depth *values* in the shadow map, but this alone is not sufficient, as the bias has to also hide the errors caused by the mislocation of the sampling *positions* in the shadow map. Too small bias value does not remove the self-shadowing artifacts, whereas too large bias eventually detaches the shadows from the shadow casters, giving the impression that objects that in reality touch each other are floating some distance apart. See Figure 3.1c for an illustration of a situation where the bias manages to fix the resampling problem. For any given bias value, it is possible to construct a simple planar surface so that it is sloped enough to present false self-shadowing. What alleviates this problem is that the cosine between the surface normal and light direction approaches zero when the light shines from acute angles, masking the problem in the most pathological cases. Aside from the self-shadowing problem, it is evident that fine details of the shadow casters may be lost if the resolution of the shadow map is too small.

A number of techniques have been developed for coping with the shadow map resampling problem. Second-depth shadow mapping [73] stores the depth of the second-closest surface into the shadow map, which

yields correct shadows and can prevent self-shadowing in many situations. Arvo and Hirvikorpi [12] compress shadow maps in a hardware-friendly way. The compression is based on an observation that storing any depth value between the two closest surfaces gives correct shadows. Manipulating the projection from world-space 3D to light-space 2D in order to concentrate the shadow map resolution near the viewpoint was first proposed by Stamminger and Drettakis [67], and novel projection matrix constructions were subsequently presented by Wimmer et al. [76] and Martin and Tan [49]. Methods for subdividing the shadow map into a number of tiles have been presented by Fernando et al. [31] and Arvo [8]. Brabec and Seidel [19] focus the view frustum used in rendering the shadow map to the visible portion of the scene.

Irregular shadow maps

Shadow mapping can be modified so that the shadow queries can be answered exactly, by first collecting the query points and then constructing the shadow map using the transformed query points as the sampling points for rasterization. This was independently discovered by Aila and Laine [P1] and Johnson et al. [44]. The irregular sampling results in an exact hard shadow algorithm, but with the cost of losing the simplicity of rasterizing into a regular lattice of sampling points. In addition, the execution model of the algorithm is changed, since the query points must be gathered before the shadow map can be constructed.

Current graphics hardware does not directly support rasterization to irregularly located sampling points, but reasonably efficient CPU-based implementations are possible. Hardware for performing irregular rasterization efficiently has been proposed by Johnson et al. [43], and a multi-pass algorithm that utilizes current GPUs by Arvo [10].

3.2 Shadow Volumes

The *shadow volume* algorithm [26] constructs the three-dimensional volumes that represent the shadowed regions, and tests whether the visible surfaces are inside these regions or not. The hardware-accelerated version [40] is currently a popular method for rendering exact hard shadows in real-time. It works by first rendering the scene with ambient lighting only, collecting the depths of the visible surfaces in the depth buffer, and then rendering the boundaries of the shadow volumes into a hardware stencil buffer. The rendering of shadow volume boundaries effectively counts the number of times a ray between a receiver point and a reference point enters and exits the shadow volumes, and if the number of enter events is greater than the number of exit events, the visible surface point is in shadow.

Note that the intermediate data structure in the shadow volume algorithms is the depth buffer computed from the actual viewpoint. As these depth values define exactly the locations where the shadows need to be computed at, no resampling takes place, and consequently no resampling problems are encountered.

The bounding surface of a shadow volume consist of three parts: *light cap*, *dark cap* and *side quads* (Figure 3.2). The light cap is formed by

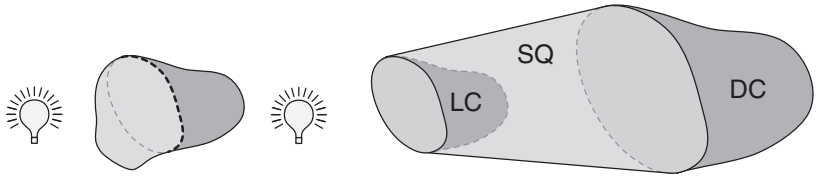


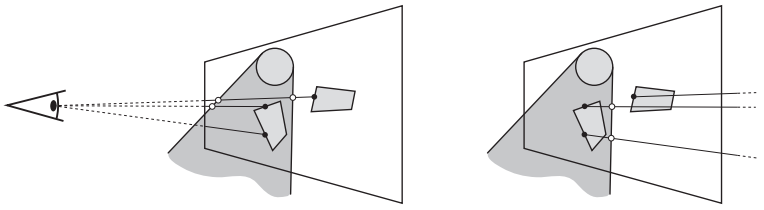
Figure 3.2: An example of a shadow volume. (a) The faces of a shadow-casting object are classified to front-facing and back-facing with respect to the point light source. The edges that separate faces with different facings are silhouette edges, and they are shown as dashed line in the figure. (b) The boundary of the shadow volume consists of three parts. The light cap (LC) is formed by the back-facing triangles of the shadow caster, the side quads (SQ) are extruded from the silhouette edges, and the dark cap (DC) closes the shadow volume. One side quad is constructed for each silhouette edge. It is common to use a replica of the light cap at the extrusion distance as the dark cap, but any surface that closes the volume is equally valid.

the back-facing triangles of the shadow caster, the side quads are extruded from the silhouette edges, and the dark cap closes the shadow volume. Bergeron [17] generalizes shadow volumes so that non-closed shadow casters can also be handled correctly.

The original shadow volume algorithm, commonly dubbed as *Z-pass* shadow volumes, counts the numbers of times a ray from the camera to a visible surface enters and exits the shadow volume using stencil buffer. See Figure 3.3a for an example. The Z-pass algorithm cannot handle cases where the shadow volumes intersect the near plane of the view frustum, since enter/exit events that occur between the camera and the near plane are missed due to view frustum clipping. Wrong results are obtained also if the camera is in shadow.

A small modification that solves the near-clip and camera-in-shadow problems, so-called *Z-fail* shadow volumes, was independently discovered by Bilodeau and Songy in 1999 and by Carmack in 2000, as explained by Everitt and Kilgard [30]. The core idea of the Z-fail algorithm is to count the events along a ray from the visible surface towards infinity, see Figure 3.3b. Shadow volumes that are clipped to the *far* clip plane are now subject to missing intersections, but conveniently, far clipping can be easily disabled by locating the far clip plane at infinity. Similar trick is not possible with the near clip plane.

Another method for handling the problematic cases was later found by Hornus et al. [41]. Their *ZP+* algorithm handles the shadow volumes lying between the camera and the near clip plane in a separate pass, which allows using the Z-pass algorithm with near-clipping shadow volumes and in cases where the camera is in shadow.



(a) Z-pass (b) Z-fail

Figure 3.3: Z-pass and Z-fail shadow volume algorithms. The round object is the only shadow caster in this example. (a) In Z-pass algorithm, the number of enter/exit events is counted along a ray from the camera to the visible surface point. Because the counting is done using rasterization into stencil buffer, only the events that occur inside the view frustum can be counted, which leads to problems at the near clip plane. In this example, the lowest point shown is incorrectly determined as being lit, because the intersection with the shadow volume boundary lies outside the view frustum. (b) In Z-fail algorithm, the number of events is counted along a ray from the visible surface point to infinity. In this example, correct results are obtained for all three points shown. Obviously, shadow volumes that intersect the back clip plane would cause similar problems as in case (a), but clipping to the far plane of the view frustum can be easily prevented, whereas clipping to the near plane cannot.

Optimizations for shadow volumes

Shadow volume algorithms are often criticized for their excessive fillrate consumption, and various methods for alleviating this problem have been presented. Lengyel [46] bounds the pixel processing region by using scissor tests, and a similar bounding in depth direction can be obtained by using the depth bounds hardware extension [55]. Combining both scissor test and depth bounds was suggested by McGuire et al. [52]. Lloyd et al. [47] gives a method for conservatively culling redundant parts of shadow volumes. Aila and Akenine-Möller [3] present a two-stage shadow volume rendering algorithm that first identifies screen-space tiles that may contain a shadow boundary, and then rasterizes shadow volume boundaries with full resolution only in the potential boundary tiles. This technique requires modifications in the hardware. Décoret [27] introduces a novel data structure, *N-buffer*, that enables performing rectangular occlusion tests in constant time regardless of the size of the rectangle, and presents a shadow volume clamping algorithm that utilizes the new data structure.

3.3 Ray Casting

Ray casting [6] is the most frequently used method for computing both hard and soft shadows in offline rendering. In this thesis, we use term *ray casting* when referring to the process of determining visibility between two points, and reserve *ray tracing* [74] for the process of following illumination paths by performing repeated ray casts.

For a point light source, it is enough to cast a single shadow ray per

receiver point in order to determine whether it is in shadow or not. For an area light source, stochastic shadow ray casting [25] is used for sampling the light source with a large number of shadow rays, essentially evaluating Equation 2.6 that was presented in Chapter 2. Because of the widespread use of ray casting, it was also used as baseline for evaluating the performance of all offline shadow algorithms in this thesis.

Constructing a spatial acceleration structure of shadow-casting geometry makes it possible to perform ray casts in sub-linear time with respect to the number of shadow-casting primitives. Performing ray casts efficiently is a topic of great interest in computer graphics, and vast amount of research has been devoted to it. We will not go into details here, but refer the reader to extensive surveys on the topic [22, 7, 38] and an interesting analysis of worst-case and average-case complexity of different methods by Szirmay-Kalos and Márton [70]. Many of the recent advances in ray-casting methods are related to tracing multiple rays simultaneously. As an example of a recent technique we mention the spectacularly fast multi-level algorithm of Reshetov et al. [60]. A programmable hardware unit for ray casting has been described by Woop et al. [69], and the strongest proponents of ray tracing argue that the traditional rasterization-based rendering will be eventually replaced by ray tracing. Whether this will happen or not, the developments in the field are certainly interesting.

Optimizations specific to shadow rays, where finding any surface that intersects the ray is sufficient, include *shadow caching* [35]. In shadow caching, a list is maintained for primitives that have most recently succeeded in blocking shadow rays. When casting a new shadow ray, it is first tested against the recently encountered occluders, and if intersection is found, the test is finished. Shadow caching does not accelerate the visible shadow ray casts.

3.4 Miscellaneous Hard Shadow Algorithms

Projection shadows [18] can be applied if the surface that receives the shadows is planar. This method is based on construction of a matrix that transforms the triangles of a shadow caster from their world-space position projectively onto the plane of the shadow receiver, and it produces exact results. Conceptually, the shadow-casting object is flattened into its own shadow. The most critical limitation of projection shadows is that only planar shadow receivers can be supported, making the method unsuitable for general-purpose rendering.

The same approach can be used for computing a shadow map that is pixel-exact on a given 3D plane. Chong and Gortler [24] compute such shadow maps for multiple shadow-receiving planes, and perform arbitrary shadow queries by choosing the most suitable shadow map for the query. The queries for points lying on the planes used can be answered exactly, but other queries suffer from the resampling problems.

Chan and Durand [21] present a hybrid method for culling shadow volumes based on shadow maps. In their method, a shadow map is used for finding regions that are in full shadow or fully lit, and shadow volumes are used for determining the shadows in the remaining regions only. Because

of the finite resolution of the shadow map, it is possible that details of the shadow casters are lost due to their absence in the shadow map, and thus the algorithm is not exact. McCool [50] constructs shadow volumes based on shadow maps. The benefit of the method is that the resulting shadow volumes have no geometrical overlap, but again, due to the use of shadow maps, details of the shadow casters may be lost. Sen et al. [62] improves the shadow boundaries in shadow maps by adding edge information to the shadow map entries. The resolution of the shadow map still limits the amount of information that can be stored, as no more than one vertex of the silhouette can be stored in each shadow map pixel. Woo et al. [77] gives a survey of hard shadow algorithms introduced prior to 1990.

3.5 Approximative Soft Shadow Algorithms

In this section, we shall briefly consider some of the approximative soft shadow algorithms, and especially focus on the penumbra wedge-based algorithms [4, 14, 15] that have certain common features with the physically-based soft shadow volume algorithm [P4, P5] presented in this thesis. An extensive survey of real-time soft shadow algorithms is given by Hasenfratz et al. [37]. The overview part of Arvo's PhD thesis [9] also contains a thorough treatment of many approximative soft shadow algorithms.

In approximative soft shadow algorithms, many simplifications are usually made that steer the result away from the correct result that would be obtained if Equation 2.6 were used. Such simplifications include, but are not limited to:

- approximating the visibility function $\mathcal{V}$,
- evaluating the BRDF function f_r only once, e.g. for the direction to the center of the light source,
- assuming constant emission function e,
- assuming constant geometry terms $\lfloor \mathbf{n}_p \cdot \omega_{in} \rfloor$, $\lfloor \mathbf{n}_l \cdot -\omega_{in} \rfloor$ and $1/|l-p|^2$ over the surface of the light source.

Most approximative soft shadow algorithms merely modulate the (approximate) color of the light illuminating the surface by an approximate scalar visibility factor, as in the following equation:

$$u(p \rightarrow \omega_{out}) \approx e(l_c, -\omega_{inc}) f_r(p, \omega_{in} \rightarrow \omega_{out}) \times \frac{\lfloor \mathbf{n}_p \cdot \omega_{inc} \rfloor \lfloor \mathbf{n}_{l_c} \cdot -\omega_{inc} \rfloor}{|l_c - p|^2} \int_L \mathcal{V}(p, l)\, dA_l. \tag{3.1}$$

Here, the emission function e, BRDF function f_r, and the geometry factors are computed only for the center of the light source l_c, and assumed to be constant over the surface of the light source, allowing them to be moved outside the integral. Symbol ω_{inc} denotes the direction from p to the center of the light source. The approximation is reasonable if the light source is small compared to the distance to the surface being shaded, and the BRDF of the surface is not exceedingly reflective. When the light source

is enlarged and moved closer to the receiving surface, the approximation error rises rapidly.

It is sometimes claimed (e.g. by Hasenfratz et al. [37]) that a shadow algorithm that computes the visibility integral $\int_L \mathcal{V}(p,l)\,dA_l$ correctly gives physically-based shadows. Even though this may hold sufficiently well in certain situations, it is evident that Equations 2.4 and 3.1 are different. Because of this, we shall not count these algorithms as being physically-based in this thesis.

Even if we are happy with the approximation of Equation 3.1, the problem of evaluating the visibility integral still remains. Hence, various forms of trickery are used for approximating the visibility integral without sampling, and how this is exactly done depends on the algorithm in question. Many techniques (e.g. Reeves et al. [59], Brabec and Seidel [20], Arvo et al. [13]) compute soft or antialiased shadows by filtering hard shadow boundaries. St-Amour et al. [66] combine several shadow maps into a discretized penumbra deep shadow map (PDSM) that captures the changes in light attenuation in each PDSM pixel. Real-time rendering with soft shadows is possible with a precomputed PDSM, and shadow queries can be made at arbitrary points, allowing e.g. fog to be shadowed.

Approximative real-time *soft shadow volumes* [4, 14, 15] are based on construction of penumbra wedges that are closed volumes containing the points where a single silhouette edge may cast penumbra. The most appealing feature of soft shadow volumes is that only the silhouette edges need to be taken into account, and the number of silhouette edges is typically much smaller than the number of triangles [51].

In these algorithms, the penumbra wedges are rasterized one by one using graphics hardware, and occlusion coverage is collected to the screen-space pixels. The inherent approximations in these techniques are:

- silhouette edges of the shadow casters are computed only once, from the center of the light source, and assumed to hold for all receiver points,
- at most one surface is assumed to block each possible ray between surface being shaded and the light source,
- BRDF and geometry terms are assumed to remain constant.

However, the most recent of these algorithms [14] supports area light sources with spatially varying emission. As silhouette edges are processed one at a time, and the occlusion coverage is stored as a scalar value, it is impossible to store information about which parts of the light source are occluded from the point being shaded. Therefore, areas that are covered multiple times are not handled correctly. A possible solution to the problem of constant silhouettes is executing a fast silhouette extraction algorithm [61, 42] from every receiver point, but this would complicate and slow down the algorithm significantly.

Cone tracing [5] approximates the visibility inside a circular cone, and this can be applied in soft shadow computations. For shading a point, a cone is constructed so that the apex is at the point being shaded, and the base covers the area light source. In practice, occlusion inside the cone

has to be modeled heuristically, since exact clipping of the cone easily leads to unbearable fragmentation and complicated geometry because of the curved surface of the cone. *Pencil tracing* [63] operates on a set of rays that lie inside a cone-shaped beam (with possibly non-circular base) so that in certain situations only the central ray needs to be traced. Optical events such as transport, reflection and refraction of the pencil are modeled using simple matrix algebra, which enables analyzing errors caused by the pencil-spread angle. Validating if the destination of all rays in a pencil can be derived from the central ray is non-trivial.

Soler and Sillion [65] approximate soft shadows using convolution, and present a hierarchical algorithm that drives the approximation error below a threshold value. Agrawala et al. [1] present an image-based soft shadow algorithm that uses layered attenuation maps for fast approximations. A coherent ray tracer is used for generating higher-quality images. Bala et al. [16] approximate soft shadows by computing the shadowed illumination in a sparse set of points, and then filtering the output image by taking into account important discontinuities such as shadow boundaries. Parker et al. [56] render soft shadows at interactive rates in a parallel ray tracer by using only a single sample per pixel and "soft-edged" objects. Their algorithm is very fast, but not physically-based.

3.6 Physically-Based Soft Shadow Algorithms

In physically-based soft shadow algorithms, as defined in a somewhat strict sense in this thesis, the visibility is computed correctly and in a fashion that enables approximating the integral in Equation 2.4.

Beam tracing [39] constructs a polygonal beam between the point to be shaded and the light source. Occlusion inside the beam is correctly modeled by clipping the beam according to occluders. In highly tessellated scenes, the beam geometry quickly becomes prohibitively complex, and the benefits of using analytic geometrical representation instead of a set of rays is lost. Ghazanfarpour and Hasenfratz [33] introduce a variant that subdivides the beam recursively until the entire beam is either free of occluding geometry or blocked by a single triangle. Subdivision is also terminated when a specified subdivision limit is reached. Clipping to occluder geometry is avoided in this approach, but since connected surfaces cannot block the beam, the beam has to be subdivided all the way to the limit when it contains an internal edge of a single occluding surface.

Marks et al. [48] consider the common bounding volume of two quadrilateral patches, and note that shadow rays between these patches can only be blocked by the shadow casters inside the volume. Thus, when casting the shadow rays, all other objects in the scene can be ignored. This idea was extended to *shaft culling* [36, 34, 28] by considering the common bounding volume—called a *shaft*—between two bounding volumes. Finding the objects inside the shaft again allows the shadow rays to consider only those objects, leading to faster visibility queries.

Nishita and Nakamae [54] construct two shadow volumes for bounding the penumbra regions of a light source. Shadow terms are computed analytically for polygons that intersect the penumbra. Shadow casters need

to be decomposed into convex polyhedra, which limits the practicality of the approach. Chin and Feiner [23] take a similar approach by constructing two BSP trees for umbra and penumbra volumes. Again, analytic shadow terms are computed by traversing the BSP trees and clipping away the occluded parts of a polygonal light source.

Tanaka and Takahashi [71] use angular subdivision for culling shadow casters affecting the shadows at a given receiver point. The remaining set of shadow casters is further refined by removing shadow casters that do not lie inside the pyramid formed by the receiver point and the light source. Finally, all silhouette edges of the shadow casters are projected onto the surface of the light source, and analytical shadow term is computed by clipping away the occluded parts of the light source. Stark and Riesenfeld [68] compute exact illumination from diffusely emitting polygons by enumerating the vertices of the shadow casters, including apparent vertices that result from intersecting edges, which avoids polygon clipping.

A method for bounding the penumbrae cast by objects that can be approximated by spheres has been presented by Formella and Łukaszewski [32]. Since conservative decomposition of arbitrary objects into spheres is a non-trivial task, this algorithm has quite limited applicability.

Casting separate shadow rays [6, 25] is still the most popular method for solving the values of the visibility function in Equation 2.6. The multi-level ray-tracing algorithm of Reshetov et al. [60] could be used for accelerating the shadow ray casts as well.

4 SHADOW ALGORITHMS

This chapter outlines the new contributions in the shadow algorithms in this thesis. Publication [P1] presents a method for performing shadow mapping so that depth values are computed exactly at the receiver points, which removes the need for resampling. Extending this method to area light sources [P2] allows rendering physically-based soft shadows. Publication [P3] describes a method for accelerating the rendering of shadow volumes [26, 40] by locally choosing the faster of two existing techniques. A physically-based soft shadow algorithm that uses shadow volumes is presented in Publication [P4], and the geometrical acceleration structure used in the original algorithm is replaced with a more efficient one in Publication [P5].

At the end of the chapter, a couple of observations are made about the general approach taken towards shadow computation in this thesis, and the applicability of hierarchical shadow algorithms in general.

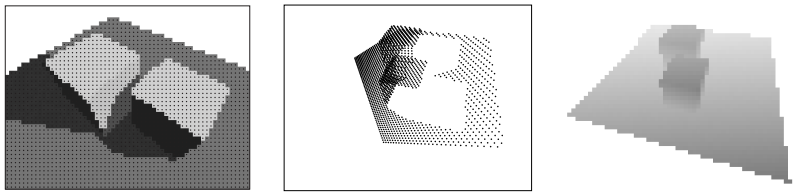
4.1 Alias-Free Shadow Maps

Publication [P1] presents a method for obtaining exact hard shadows using a shadow map-based approach. As was discussed in Chapter 3, the fundamental resampling problem in shadow maps is caused by first constructing the shadow map without *a priori* information about the locations of the receiver points, and after this, performing the shadow queries, i.e. the lookups in the shadow map, that determine which receiver points are in shadow. In alias-free shadow maps, the receiver points are first gathered by rendering the scene from the camera and recording the depth values from the depth buffer (Figure 4.1a). Then, the receiver points are transformed to the image space of the light source, yielding the correct sampling points for the shadow map (Figure 4.1b). A 2D kd-tree is constructed for the sampling points, after which every shadow-casting triangle is hierarchically "rasterized" into the set of sampling points.

Since the shadows are computed exactly at the receiver points, no artificial bias is needed for battling against the geometrical discrepancy that would otherwise result from the resampling. A small constant bias is still needed to account for the quantization of the depth values in the initial rasterization, and for the loss of numerical precision in the transformation of the receiver points into the light space.

4.2 Hierarchical Penumbra Casting

Publication [P2] presents a novel method, hierarchical penumbra casting, for rendering physically-based soft shadows. In traditional stochastic sampling of area light sources [25], every visibility relation, i.e. visibility between a light sample and a receiver point, is sampled separately by casting a shadow ray. Hierarchical penumbra casting is effectively a transpose of this classical approach. First, every receiver point and light sample is collected by i.e.



(a) (b) (c)

Figure 4.1: Illustration of alias-free shadow maps. (a) An image of a simple scene as seen from the camera. The receiver points on the visible surfaces are drawn as black dots. (b) The receiver points as seen from the light source. (c) A traditional shadow map, with shades of grey corresponding to different depth values, shown for comparison purposes. Note that the lookups in (b) do not land at the centers of the pixels in (c), which causes the resampling problem. In alias-free shadow maps, the depth values are computed exactly at the sampling points in (b), i.e. at the light-space projections of the actual receiver points, which removes the resampling problem.

rasterization or ray tracing, which fully defines the set of visibility relations whose statuses need to be computed. Then, each shadow-casting triangle is processed individually, and all visibility relations blocked by each triangle are found and their status updated using a hierarchical traversal. The status of all visibility relations are stored in memory throughout the computation, and after all triangles are processed, the shading can be performed using this data.

An interesting consequence of the transposed computation order is that the full set of shadow-casting triangles does not need to reside in main memory at any point of the computation. Therefore, the memory consumption of the hierarchical penumbra casting algorithm is not affected by the number of triangles in the scene, which allows extremely complex scenes to be rendered. On the other hand, storage space is needed for storing the status of all visibility relations, which makes the memory consumption strongly dependent on the resolution of the image and the number of light samples used. To remedy this, we allocate memory for the status of the visibility relations only when needed, which reduces the average-case memory consumption considerably.

The hierarchical finding of blocked visibility relations is done by traversing a 3D BSP constructed for the receiver points and a three-level hierarchy of the light samples. Determining if a triangle may shadow a group of visibility relations is done by testing intersections of various penumbra volumes against the bounding volumes of the nodes of the receiver point BSP. Two kinds of penumbra volumes are constructed for each triangle: a main penumbra volume constructed from the triangle and the bounding polygon of the light source (Figure 4.2a), and several smaller penumbra volumes spanned by different light sample groups in the three-level light sample hierarchy (Figure 4.2b). Finally, hard shadow volumes formed for each light sample (Figure 4.2c) are used for testing which visibility relations are shadowed by the triangle.

Several optimizations to the basic HPC algorithm are discussed in Pub-

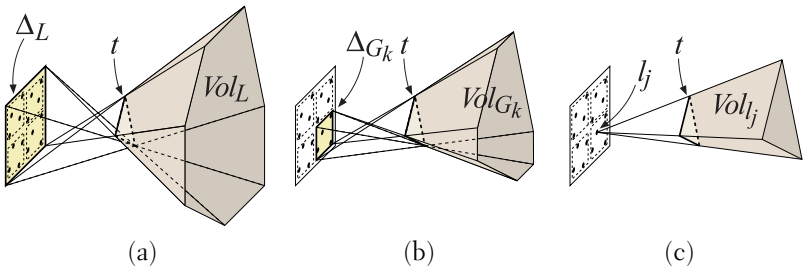


Figure 4.2: The volumes associated with the three levels of the light sample hierarchy. In this illustration, the light source consists of 16 light samples grouped into four light sample groups. (a) The main penumbra volume Vol_L is defined by the bounding polygon Δ_L of the light source and the blocker triangle t. (b) For each light sample group G_k, a penumbra volume Vol_{G_k} is constructed based on the bounding polygon Δ_{G_k} of the light sample group and the blocker triangle t. (c) For each light sample l_j, a hard shadow volume Vol_{l_j} is constructed, based on the location of l_j and the blocker triangle t.

lication [P2]. These include the aforementioned on-demand allocation of visibility relation status bits, tagging fully shadowed receiver point nodes in the receiver point tree to enable early traversal termination in shadowed regions, keeping track of which planes of the penumbra volumes may still intersect the receiver point nodes during traversal, lazy construction of penumbra volumes and hard shadow volumes, and sorting the blockers coarsely before computation according to their estimated occlusion power. With these optimizations, the execution time and memory consumption of the HPC algorithm compare favorably against the classical stochastic ray casting.

4.3 Split-Plane Shadow Volumes

Publication [P3] introduces a novel acceleration technique for rendering shadow volumes. The method is based on a simple observation: Z-pass and Z-fail shadow volumes tend to require different amounts of pixel processing in different parts of the image, depending on the location of the shadow volume relative to the visible geometry.

Figure 4.3 illustrates this effect. In Figure 4.3a, a tree casts a shadow to uneven ground, and also to itself. Z-pass shadow volumes update the stencil buffer when the shadow volume lies in front of visible geometry (Figure 4.3b), whereas Z-fail shadow volumes require stencil buffer updates when the shadow volume is behind the visible geometry (Figure 4.3c). By choosing between Z-fail and Z-pass algorithms locally, i.e. making the choice independently in different parts of the image, the amount of pixel processing can be reduced considerably (Figure 4.3d).

The choice between using Z-pass and Z-fail algorithms is done by constructing a *split plane* that approximately divides the shadow volume in two halves. If the location of the split plane is such that it is visible, Z-fail

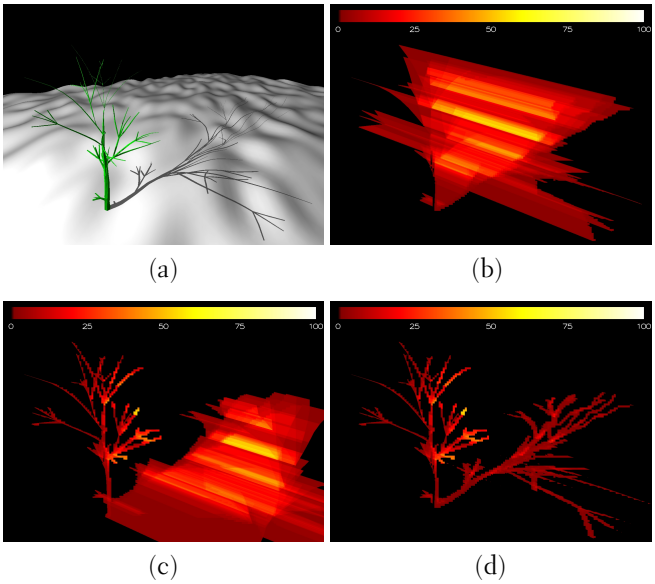


Figure 4.3: Split-plane shadow volumes. (a) A scene where a tree casts a shadow to bumpy ground, and also shadows itself. (b) A false-color image showing how many times each pixel is processed in the stencil buffer update phase using Z-pass shadow volumes. (c) As in (b) but using Z-fail shadow volumes. (d) With split-plane shadow volumes, the amount of pixel processing is reduced considerably, while the resulting shadow quality is not affected.

algorithm is used, and conversely, if the split plane lies behind the visible geometry, Z-pass algorithm is used. Since both Z-pass and Z-fail algorithms render correct shadows, switching between algorithms does not introduce any artifacts in the image. The ZP+ algorithm of Hornus et al. [41] is required for enabling the use of Z-pass shadow volumes even when they intersect the near plane of the view frustum, or when the camera is in shadow.

Accelerating the hardware rendering of shadow volumes requires that tiles of pixels, e.g. 8×8 pixels in size, can be efficiently culled so that their individual pixels are not processed at all. The depth of the split plane would need to be tested against a value fetched from the low-resolution depth buffer that is internally constructed by all modern GPUs. Unfortunately, culling entire pixel tiles programmatically is not possible with current hardware, but future generations of GPUs will hopefully have features for supporting this. The required hardware modifications would be quite small, and the need for constructing and accessing such low-resolution "computation masks" has been recognized in other shadow algorithms as well [21].

4.4 Soft Shadow Volumes for Ray Tracing

Publication [P4] presents an algorithm for computing physically-based soft shadows efficiently for a single receiver point. Although the coherence

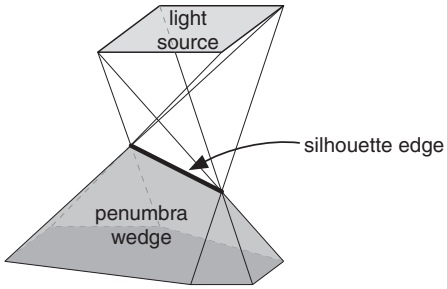


Figure 4.4: A penumbra wedge consists of the separating planes between a polygonal light source and a spanning silhouette edge. When looking towards the light source from any point inside the penumbra wedge, the spanning edge lies in front of the light source. Conversely, the edge does not overlap the light source when seen from any point outside the corresponding wedge. Penumbra wedges can be seen as special cases of penumbra volumes that were illustrated in Figure 4.2.

in shadows between nearby receiver points cannot be used, the execution model of the algorithm is very well suited for general-purpose rendering where e.g. surface shaders may require on-the-fly evaluation of shadow terms due to conditional statements. Also, adaptive anti-aliasing is easiest to implement when shadow queries can be performed whenever necessary, without needing to collect all receiver points before commencing the shadow computation.

In the SSV algorithm, each potential silhouette edge is identified and a corresponding penumbra wedge (Figure 4.4) is constructed for each of these in a preprocessing phase. A penumbra wedge is an infinite volume that defines the region in space where the spanning edge may affect the shadows. When looking towards the light source from any point inside the wedge, the projection of the spanning edge overlaps the light source, whereas from any point outside the wedge, the edge does not overlap the light source, and thus does not affect the shadow. The penumbra wedges of all silhouette edges are conservatively rasterized into a hemicube-like data structure (Figure 4.5a) for quick access during the shadow queries.

During rendering, the penumbra wedges containing the receiver point under consideration are fetched from the acceleration structure, and the set of corresponding potential silhouette edges is pruned to contain only the actual silhouette edges as seen from the receiver point. The silhouette edges are then projected onto the surface of the light source, and the visibility between the receiver point and a number of individual light samples is determined using a variant of a polygon filling algorithm. Because of using only the parts of the silhouette edge loops that project onto the light source, a single reference ray is needed for testing if all light samples are occluded, as this cannot be determined from the silhouette edges alone.

The main weakness of the SSV algorithm is that the potential silhouette edge sets returned by the spatial acceleration structure quickly gets very conservative when the spatial size of the scene is increased. Nevertheless, the algorithm gives impressive speedups compared to traditional ray casting.

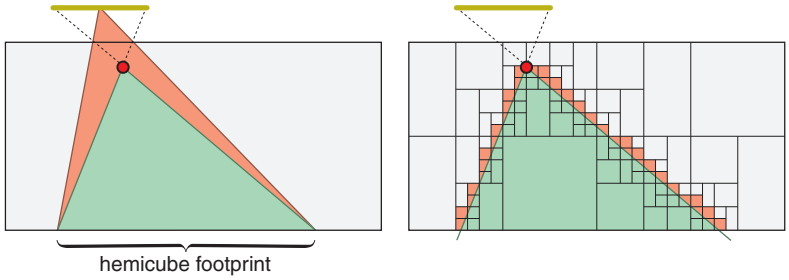


Figure 4.5: Comparison of the hemicube and BSP acceleration structures in the SSV algorithm. This 2D illustration can be thought of a cross section of a corresponding 3D situation. The light source is at the top, the dot marks the silhouette edge, and the triangle below it is the penumbra wedge. The outer region illustrates the conservativeness of the data structure, i.e. the points where the edge is reported as a silhouette edge even though the point does not lie inside the true penumbra wedge. (a) The original hemicube structure considers only the hemicube footprint of the wedge, which is the intersection of the wedge with the surface of the hemicube. The edge is reported for all points lying in the pyramid formed by the penumbra footprint and the center of the light source. (b) In the BSP structure, the wedge is listed in nodes that intersect it. If a node is completely enclosed by a wedge, the wedge is stored into its wedge list, even when the node is not a leaf node of the BSP.

Improved spatial acceleration structure

In Publication [P5], the hemicube acceleration structure of the SSV algorithm is replaced with a more efficient, lazily constructed 3D BSP. In this data structure, each node of the BSP contains the list of wedges that enclose it completely, or if the node is a leaf node, the list of wedges that intersect it at all. A single traversal from the root of the tree to the leaf node containing the point is sufficient for gathering the full list of wedges that contain the point. Figure 4.5 illustrates the differences between the hemicube and BSP acceleration structures.

The 3D acceleration structure allows trimming off the portions of the penumbra wedges from which the spanning edge would not be a silhouette. The *silhouette region* of an edge is defined by the planes of the triangles connected to it, and only the intersection of the penumbra wedge and the silhouette region can contain points that actually need to use the edge for shadow computations. Being able to take the silhouette regions into account is a major improvement, especially with smoothly curved shadow casters where each edge is a silhouette from only a small region in space.

Another feature in the new acceleration structure is that it is constructed lazily so that subdivision occurs only in regions that actually receive shadow queries. Storage space is thus not wasted by subdividing regions where shadows are not computed. Subdivision is terminated when the reduction in the number of wedges becomes too small, or when the spatial size of the BSP node falls below a given threshold.

The new acceleration structure behaves more predictably than the original hemicube structure, meaning that seemingly minor modifications of the scene, i.e. rotating the light source, do not heavily affect the performance. In addition, the tighter set of silhouette edges obtained through the use of silhouette regions was observed to consistently improve the performance of the SSV algorithm, albeit only slightly in situations where the weaknesses of the hemicube structure did not become into play.

4.5 Applicability of hierarchical methods

The algorithms in this thesis make it possible to compute physically-based shadows more efficiently than what was possible before. Two characteristic features can be recognized: the use of hierarchical acceleration structures, and reordering the computations so that hierarchical structures can be exploited.

Obviously, the reason for using hierarchical structures is that they make the execution time of many operations sub-linear with respect to the size of the data. The asymptotic behavior of an algorithm is thus enhanced. However, there are two drawbacks.

The first drawback is that every hierarchy needs to be constructed before it can be used. Even when the hierarchy itself is constructed lazily, the data must be effectively predetermined and readily accessible. Gathering the data beforehand may require reordering the computation, which affects the execution model and memory consumption of an algorithm so that using it usually becomes more cumbersome. For instance, AFSM, HPC and SPSV algorithms need to know all receiver points beforehand, which makes things like adaptive anti-aliasing or certain programmable shaders difficult to implement efficiently. On the other hand, if the receiver points are not processed *en masse*, no such problems occur. As in the SSV algorithm, this yields more relaxed execution model, but linear execution time complexity with respect to the number of receiver points.

The second drawback is that the base cost of traversing hierarchical structures may be high. This is especially true when memory latencies are high and coherent memory access patterns are necessary for maintaining sufficient cache hit rates. Unfortunately, this is the case in current CPUs and especially GPUs. As a consequence, some hierarchies may become beneficial only for impractically large inputs.

Still, when the size of the data is large enough for the sub-linear processing time to manifest itself, hierarchical techniques are extremely efficient, and this is already the case in many real-world rendering scenarios today. It can be expected that the number of primitives in a scene continues to grow. Image resolution and desired supersampling density are probably not going down, either. Hence, algorithms that are sub-linear with respect to those parameters can be assumed to increase their applicability in the future.

The problem of changed, potentially cumbersome execution model is quite severe, as it does not go away with increased processing power or even special-purpose hardware. Sometimes it is simply not practical to require having a complete dataset—e.g. all triangles or all receiver points—known beforehand. In these situations, some algorithmic designs must be

excluded from further consideration. Conversely, whenever some part of the data *is* available before starting the actual computation, and it remains static throughout the computation, one should strive to take advantage of this. The algorithms in this thesis are examples of this design principle.

5 MAIN RESULTS OF THE THESIS AND CONTRIBUTIONS OF THE AUTHOR

Publication [P1] was included in the doctoral thesis of Timo Aila, whereas Publications [P2–P5] have not previously formed a part of another thesis. The main results of the thesis and the author's contributions can be summarized as follows.

Publication [P1]
This article abandons the regular structure of shadow maps. The visible samples are first transformed from screen space to the image plane of a light source. The transformed points are then used as sampling points when the geometry is rasterized into the shadow map. This provides correct sampling points for the shadow map, and matches the result of tracing a shadow ray from each visible sample to the light source. As a result, all problems that result from the finite resolution of traditional shadow maps are eliminated.

Dr. Timo Aila invented the original idea of using the transformed screen-space points as shadow map sampling points, and wrote all text in the paper. The author designed the hierarchical rasterization algorithm and implemented the algorithm. The author also constructed the test cases, performed the performance measurements, and also produced all images and illustrations that appear in the paper.

Publication [P2]
A novel approach for rendering physically-based soft shadows is presented. Unlike in standard stochastic shadow computation, where each visibility relation between light sources and visible surfaces is considered separately, the proposed algorithm first gathers all the needed visibility relations and then processes every shadow-casting triangle separately, hierarchically finding the visibility relations blocked by the triangle. As a consequence, the memory consumption of the algorithm does not depend on the geometry content of the scene. A practical implementation with numerous optimizations is presented, and the performance of the algorithm is shown to compare favorably against stochastic ray casting.

The author invented the original idea and designed the algorithm, with the presented optimizations, for finding blocked visibility relations by simultaneously traversing both the light sample and the receiver point hierarchies. The author implemented the algorithm, performed the measurements, and wrote 90% of the paper.

Publication [P3]
This article analyzes the causes for the inefficiency in rendering shadow volumes, and presents a novel method for reducing the amount of redundant pixel processing during rendering. The algorithm is based on choosing locally between two previous algorithms, so that for every screen-space pixel tile, the estimated amount of redundant pixel processing is minimized. Two methods are presented for constructing geometrical split planes that

are used for choosing the shadow volume algorithm with expected better performance. In addition, an optimization based on decomposing the shadow casters into multiple parts is shown to give additional performance benefit, when used in conjunction with the proposed algorithm.

The author is the sole author of this publication.

Publication [P4]

This article presents an algorithm for rendering physically-based soft shadows efficiently in ray tracing-based renderers. The algorithm consists of two phases. In the preprocessing phase, penumbra wedges are constructed for potential silhouette edges, and stored into a spatial acceleration structure for quick access in the query phase. When performing a shadow query, a conservative set of silhouette edges is fetched from the acceleration structure and refined to contain only the relevant edges. These are then projected onto the surface of the light source, and the integral of the changes in the depth complexity caused by the edges, giving relative depth complexities of all light samples, is computed using a custom polygon-filling algorithm. Finally, the visibility of all light samples is solved by casting a single reference shadow ray that determines the actual depth complexity of one of the light samples.

Dr. Timo Aila and Dr. Tomas Akenine-Möller invented the idea of finding the silhouette edges based on penumbra wedges and solving the relative depth complexities. Dr. Tomas Akenine-Möller also proposed using a hemicube acceleration structure for storing the footprints of the penumbra wedges. Dr. Ulf Assarsson proposed casting a single reference shadow ray. The author designed the multiresolution storage structure for the hemicube, the rules used in the depth complexity integration, the bucketing optimization for the integration step, and the optimal revertion strategy for cases where precision issues prevent casting a robust reference ray. The author also implemented the entire algorithm and wrote 70% of the paper.

Publication [P5]

This article identifies and analyzes the weaknesses of the hemicube acceleration structure that was used in the soft shadow volume algorithm presented in Publication [P4], and proposes replacing the hemicube with a lazily-constructed 3D BSP. This acceleration structure is shown to avoid the identified pitfalls of the hemicube, and to give better performance and more predictable execution times.

The author and Mr. Jaakko Lehtinen formulated the proposed acceleration structure and the related algorithms together. The observation that silhouette region optimization became possible with the new data structure was also made together by the author and Mr. Lehtinen. The author was mainly responsible for identifying and analyzing the weaknesses of the hemicube, and also wrote 10% of the paper. Mr. Lehtinen implemented the BSP acceleration structure, performed all measurements, and wrote most of the paper.

BIBLIOGRAPHY

[1] Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laruent Moll. Efficient Image-Based Methods for Rendering Soft Shadows. In *Proceedings of ACM SIGGRAPH 2000*, pages 375–384, 2000.

[2] Timo Aila. *Efficient Algorithms for Occlusion Culling and Shadows.* PhD thesis, Helsinki University of Technology, 2005.

[3] Timo Aila and Tomas Akenine-Möller. A Hierarchical Shadow Volume Algorithm. In *Graphics Hardware*, pages 15–23, 2004.

[4] Tomas Akenine-Möller and Ulf Assarsson. Approximate Soft Shadows on Arbitrary Surfaces using Penumbra Wedges. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 297–305. Eurographics Association, 2002.

[5] John Amanatides. Ray tracing with cones. In *Proceedings of SIGGRAPH '84*, pages 129–135. ACM Press, 1984.

[6] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proc. AFIPS Spring Joint Computing Conference*, pages 37–45, 1968.

[7] James Arvo and David Kirk. *An Introduction to Ray Tracing* (ed. Andrew S. Glassner), chapter A Survey of Ray Tracing Acceleration Techniques, pages 201–262. Academic Press Ltd., London, UK, 1989.

[8] Jukka Arvo. Tiled Shadow Maps. In *Proceedings of Computer Graphics International*, pages 240–247. IEEE Computer Society, 2004.

[9] Jukka Arvo. *Efficient Algorithms for Hardware-Accelerated Shadow Computation.* PhD thesis, Turku Centre for Computer Science and University of Turku, 2005.

[10] Jukka Arvo. Alias-Free Shadow Maps using Graphics Hardware. To appear in *Journal of Graphics Tools*, 2006.

[11] Jukka Arvo and Timo Aila. Optimized shadow mapping using the stencil buffer. *Journal of Graphics Tools*, 8(3):23–32, 2004.

[12] Jukka Arvo and Mika Hirvikorpi. Compressed Shadow Maps. *The Visual Computer*, 21(3):125–138, 2005.

[13] Jukka Arvo, Mika Hirvikorpi, and Joonas Tyystjärvi. Approximate Soft Shadows with an Image-Space Flood-Fill Algorithm. *Computer Graphics Forum*, 23(3):271–279, 2004.

[14] Ulf Assarsson and Tomas Akenine-Möller. A Geometry-Based Soft Shadow Volume Algorithm using Graphics Hardware. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3):511–520, 2003.

[15] Ulf Assarsson, Michael Dougherty, Michael Mounier, and Tomas Akenine-Möller. An Optimized Soft Shadow Volume Algorithm with Real-Time Performance. In *Graphics Hardware*, pages 33–40. ACM SIGGRAPH/Eurographics, 2003.

[16] Kavita Bala, Bruce Walter, and Donald P. Greenberg. Combining Edges and Points for Interactive High-Quality Rendering. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3):631–640, 2003.

[17] Philippe Bergeron. A General Version of Crow's Shadow Volumes. *IEEE Computer Graphics and Applications*, 6(9):17–28, 1986.

[18] James F. Blinn. Me and my (fake) shadow. *IEEE Comput. Graph. Appl.*, 8(1):82–86, 1988.

[19] Stefan Brabec and Hans-Peter Seidel. Practical Shadow Mapping. *Journal of Graphics Tools*, 7(4):9–18, 2002.

[20] Stefan Brabec and Hans-Peter Seidel. Single Sample Soft Shadows using Depth Maps. In *Graphics Interface 2002*, pages 219–228, 2002.

[21] Eric Chan and Frédo Durand. An efficient hybrid shadow rendering algorithm. In *Proceedings of the Eurographics Symposium on Rendering*, pages 185–195. Eurographics Association, 2004.

[22] Allen Y. Chang. A survey of geometric data structures for ray tracing. Technical Report TR-CIS-2001-06, CIS Department, Polytechnic University, Brooklyn, New York, 2001.

[23] Norman Chin and Steven Feiner. Fast Object-Precision Shadow Generation for Area Light Source using BSP Trees. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 21–30. ACM Press, 1992.

[24] Hamilton Chong and Steven Gortler. A Lixel for every Pixel. In *Proceedings of the Eurographics Symposium on Rendering*, pages 167–172. Eurographics Association, 2004.

[25] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. In *Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, pages 137–145. ACM, 1984.

[26] Frank Crow. Shadow Algorithms for Computer Graphics. In *Computer Graphics (Proceedings of ACM SIGGRAPH 77)*, pages 242–248. ACM, 1977.

[27] Xavier Décoret. N-buffers for efficient depth map query. *Computer Graphics Forum*, 24(3), 2005.

[28] Kirill Dmitriev, Vlastimil Havran, and Hans-Peter Seidel. Faster ray tracing with SIMD shaft culling. Research Report MPI-I-2004-4-006, Max-Planck-Institut für Informatik, December 2004.

[29] Philip Dutré, Philippe Bekaert, and Kavita Bala. *Advanced Global Illumination*. AK Peters, 2003.

[30] Cass Everitt and Mark Kilgard. Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering. *http://developer.nvidia.com*, 2002.

[31] Randima Fernando, Sebastian Fernandez, Kavita Bala, and Donald P. Greenberg. Adaptive Shadow Maps. In *Proceedings of ACM SIGGRAPH 2001*, pages 387–390. ACM Press, 2001.

[32] Arno Formella and Andrzej Łukaszewski. Fast penumbra calculation in ray tracing. In *Proceedings of WSCG'98*, pages 238–245, 1998.

[33] Djamchid Ghazanfarpour and Jean-Marc Hasenfratz. A Beam Tracing with Precise Antialiasing for Polyhedral Scenes. *Computer Graphics*, 22(1):103–115, 1998.

[34] Eric Haines. A shaft culling tool. *Journal of Graphics Tools*, 5(1):23–26, 2000.

[35] Eric Haines and Donald Greenberg. The Light Buffer: A Ray Tracer Shadow Testing Accelerator. *IEEE Computer Graphics and Applications*, 6(9):6–16, 1986.

[36] Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.

[37] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. In *Eurographics*. Eurographics, Eurographics, 2003. State-of-the-Art Report.

[38] Vlastimil Havran. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

[39] Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. *Proceedings of SIGGRAPH '84*, pages 119–127, 1984.

[40] Tim Heidmann. Real Shadows, Real Time. *Iris Universe*, 18:28–31, 1991.

[41] Samuel Hornus, Jared Hoberock, Sylvain Lefebvre, and John Hart. ZP+: Correct z-pass stencil shadows. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 195–202. ACM Press, 2005.

[42] David Johnson and Elaine Cohen. Spatialized Normal Cone Hierarchies. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 129–134. ACM Press, 2001.

[43] Gregory S. Johnson, Juhyun Lee, Christopher A. Burns, and William R. Mark. The irregular Z-buffer: Hardware acceleration for irregular data structures. *ACM Transactions on Graphics*, 24(4):1462–1482, 2005.

[44] Gregory S. Johnson, William R. Mark, and Christopher A. Burns. The Irregular Z-Buffer and its Application to Shadow Mapping. Technical report, The University of Texas at Austin, Department of Computer Sciences, April 2004.

[45] James T. Kajiya. The Rendering Equation. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, pages 143–150. ACM Press, 1986.

[46] Eric Lengyel. The Mechanics of Robust Stencil Shadows. *http://www.gamasutra.com*, 2002.

[47] Brandon Lloyd, Jeremy Wendt, Naga K. Govindaraju, and Dinesh Manocha. CC Shadow Volumes. In *Proceedings of the Eurographics Symposium on Rendering*, 2004.

[48] Joseph Marks, Robert Walsh, Jon Christensen, and Mark Friedell. Image and intervisibility coherence in rendering. In *Proceedings of Graphics Interface '90*, pages 17–30, 1990.

[49] Tobias Martin and Tiow-Seng Tan. Anti-aliasing and Continuity with Trapezoidal Shadow Maps. In *Proceedings of the Eurographics Symposium on Rendering*, pages 153–160. Eurographics Association, 2004.

[50] Michael D. McCool. Shadow Volume Reconstruction from Depth Maps. *ACM Transactions on Graphics,*, 19(1):1–26, 2000.

[51] Morgan McGuire. Observations on Silhouette Sizes. *Journal of Graphics Tools*, 9(1):1–12, 2004.

[52] Morgan McGuire, John F. Hugues, Kevin T. Egan, Mark Kilgard, and Cass Everitt. Fast, Practical and Robust Shadows. Technical Report CS03-19, Brown University, October 2003.

[53] Fred E. Nicodemus, Joseph C. Richmond, Jack J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric Considerations and Nomenclature for Reflectance. NBS Monograph 160, National Bureau of Standards, 1977.

[54] Tomoyuki Nishita and Eihachiro Nakamae. Half-Tone Representation of 3-D Objects Illuminated by Area Sources or Polyhedron Sources. In *IEEE Computer Software and Application Conference*, pages 237–242. IEEE, 1983.

[55] NVIDIA. NVIDIA GeForceFX 5900, 5700 and Go5700 GPUs: UltraShadow Technology. Technical report, *http://www.nvidia.com*, 2003.

[56] S. Parker, P. Shirley, and B. Smits. Single Sample Soft Shadows. Technical report, University of Utah, UUCS-98-019, 1998.

[57] Pierre Poulin and John Amanatides. Shading and Shadowing with Linear Light Sources. In *Eurographics '90*, pages 377–386. Eurographics, 1990.

[58] Paul Rademacher, Jed Lengyel, Ed Cutrell, and Turner Whitted. Measuring the Perception of Visual Realism in Images. In *12th Eurographics Workshop on Rendering*, pages 235–248. Eurographics, 2001.

[59] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering Antialiased Shadows with Depth Maps. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, pages 283–291. ACM, 1987.

[60] Alexander Reshetov, Alexei Soupikov, and Jim Hurley. Multi-level ray tracing algorithm. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3):1176–1185, 2005.

[61] Pedro V. Sander, Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. Silhouette Clipping. In *Proceedings of ACM SIGGRAPH 2000*, pages 327–334. ACM Press, 2000.

[62] Pradeep Sen, Make Cammarano, and Pat Hanrahan. Shadow Silhouette Maps. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3):521–526, 2003.

[63] Mikio Shinya, Tokiichiro Takahashi, and Seiichiro Naito. Principles and applications of pencil tracing. In *Proceedings of SIGGRAPH '87*, pages 45–54. ACM Press, 1987.

[64] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996.

[65] Cyril Soler and François X. Sillion. Fast Calculation of Soft Shadow Textures Using Convolution. In *Proceedings of ACM SIGGRAPH 98*, pages 321–332, 1998.

[66] Jean-François St-Amour, Eric Paquette, and Pierre Poulin. Soft shadows from extended light sources with penumbra deep shadow maps. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 105–112, 2005.

[67] Marc Stamminger and George Drettakis. Perspective Shadow Maps. *ACM Transactions on Graphics (SIGGRAPH 2002)*, 21(3):557–562, 2002.

[68] Michael M. Stark and Richard F. Riesenfeld. Exact Illumination in Polygonal Environments using Vertex Tracing. In *11th Eurographics Workshop on Rendering*, pages 149–160. Eurographics, 2000.

[69] Jörg Schmittler Sven Woop and Philipp Slusallek. Rpu: A programmable ray processing unit for realtime ray tracing. In *Proceedings of ACM SIGGRAPH 2005 (to appear)*, July 2005.

[70] László Szirmay-Kalos and Gábor Márton. Worst-case versus average case complexity of ray-shooting. *Computing*, 61(2):103–131, 1998.

[71] Toshimitsu Tanaka and Tokiichiro Takahashi. Fast Analytic Shading and Shadowing for Area Light Sources. *Computer Graphics Forum*, 16(3):231–240, 1997.

[72] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, 1997.

[73] Yulan Wang and Steven Molnar. Second-Depth Shadow Mapping. Technical Report TR94-019, The University of North Carolina at Chapel Hill, 1994.

[74] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23(6):343–349, 1980.

[75] Lance Williams. Casting Curved Shadows on Curved Surfaces. In *Computer Graphics (Proceedings of ACM SIGGRAPH 78)*, pages 270–274. ACM, 1978.

[76] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. Light Space Perspective Shadow Maps. In *Proceedings of the Eurographics Symposium on Rendering*, pages 143–151. Eurographics Association, 2004.

[77] Andrew Woo, Pierre Poulin, and Alain Fournier. A Survey of Shadow Algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, 1990.

HELSINKI UNIVERSITY OF TECHNOLOGY
PUBLICATIONS IN TELECOMMUNICATIONS SOFTWARE AND MULTIMEDIA

TML-A1	Håkan Mitts Architectures for wireless ATM	
TML-A2	Pekka Nikander Authorization in agent systems: Theory and practice	
TML-A3	Lauri Savioja Modeling techniques for virtual acoustics	
TML-A4	Teemupekka Virtanen Four views on security	
TML-A5	Tapio Lokki Physically-based auralization – Design, implementation, and evaluation	
TML-A6	Kari Pihkala Extensions to the SMIL multimedia language	print
TML-A7	Kari Pihkala Extensions to the SMIL multimedia language	pdf
TML-A8	Harri Kiljander Evolution and usability of mobile phone interaction styles	
TML-A9	Leena Eronen User centered design of new and novel products: case digital television	
TML-A10	Sanna Liimatainen and Teemupekka Virtanen (eds.) NORDSEC 2004, Proceedings of the Ninth Nordic Workshop on Secure IT Systems	
TML-A11	Timo Aila Efficient algorithms for occlusion culling and shadows	
TML-A12	Pablo Cesar A graphics software architecture for high-end interactive TV terminals	

ISBN 951-22-8357-3 (printed version)
ISSN 1456-7911
ISBN 951-22-8358-1 (electronic version)
ISSN 1455-9722