Eurographics Symposium on Rendering (2007)
Jan Kautz and Sumanta Pattanaik (Editors)

Interactive Illumination with Coherent Shadow Maps

Tobias Ritschel[1] Thorsten Grosch[1] Jan Kautz[2] Stefan Müller[1]

[1]University of Koblenz-Landau [2]University College London

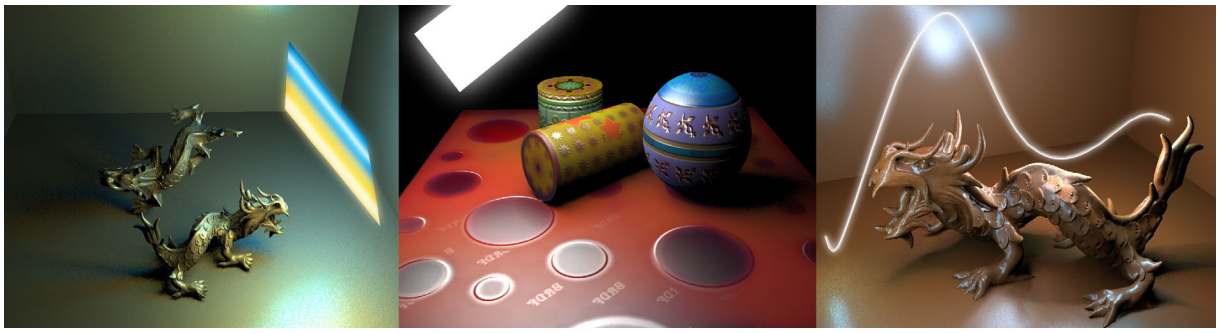


Figure 1: *The left image shows a scene with two dragons (280k faces each), where light position, shape and color, as well as object position, orientation, and all material parameters can be manipulated freely (3.3 FPS). The middle image shows bump, diffuse and specular maps, all with physically plausible shadows (4.6 FPS). In the right image, a fixed local linear light of arbitrary shape is used (1.8 seconds). All images are rendered with an NVIDIA GF 8 at 1024 × 768 pixels.*

Abstract

We present a new method for interactive illumination computations based on precomputed visibility using coherent shadow maps (CSMs). It is well-known that visibility queries dominate the cost of physically based rendering. Precomputing all visibility events, for instance in the form of many shadow maps, enables fast queries and allows for real-time computation of illumination but requires prohibitive amounts of storage. We propose a lossless compression scheme for visibility information based on shadow maps that efficiently exploits coherence. We demonstrate a Monte Carlo renderer for direct lighting using CSMs that runs entirely on graphics hardware. We support spatially varying BRDFs, normal maps, and environment maps — all with high frequencies, spatial as well as angular. Multiple dynamic rigid objects can be combined in a scene. As opposed to precomputed radiance transfer techniques, that assume distant lighting, our method includes distant lighting as well as local area lights of arbitrary shape, varying intensity, or anisotropic light distribution that can freely vary over time.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism; I.3.3 [COMPUTER GRAPHICS]: Color, Shading, Shadowing and Texture

1. Introduction

Physically-based rendering is an important problem in feature films, product visualization and many other areas. However, accurate physically-based rendering is expensive, which is largely due to time-consuming visibility tests. In this paper, we develop a data structure, *coherent shadow maps* (CSMs), that allows fast visibility queries by storing visibility information in the form of shadow maps around an object. To this end, depth maps of an object are created from many different positions and directions, capturing all geometric details. We introduce a lossless compression method that effectively compresses this large collection of depth maps by exploiting coherence between them. Yet, this compressed data structure allows for fast visibility queries. We demonstrate the effec-

tiveness of CSMs with an interactive Monte-Carlo renderer that runs entirely on the GPU. This approach to interactive illumination removes many drawbacks introduced by alternative illumination techniques (e.g., precomputed radiance transfer or soft shadow approximations), as described in the next section.

Our key contributions are:

- We introduce a novel data structure for fast visibility queries based on precomputed depth maps.
- We develop a compression scheme that allows to store the visibility information from a large collection of depth maps in a single texture.
- We demonstrate an interactive Monte-Carlo renderer that enables direct illumination computation on the GPU. We support local and distant lightsources, spatially varying BRDFs, bump mapping, high-frequency reflections, and moving rigid objects.

After reviewing previous work in Section 2, we describe our method in Section 3 and give details on the Monte-Carlo renderer in Section 5. Results are presented in Section 6, before we conclude with Section 7.

2. Previous Work

Static Precomputed Radiance Transfer PRT permits the rendering of various illumination effects on static objects, such as soft shadows and glossy reflections, in real-time [SKS02, NRH03, LSSS04]. The illumination solution is parameterized by the incident lighting, that is assumed to be represented by means of basis functions, such as spherical harmonics [SKS02] or wavelets [NRH03], allowing for efficient rendering. PRT exploits the limitation to static objects by pre-computing all the visibility queries and baking them into the parameterized solution.

Much work has focused on integrating arbitrary BRDFs [LSSS04, NRH04, WTL06, SM06], bump mapping [Slo06], and enhancing the original low-frequency technique to enable all-frequency lighting [NRH03, NRH04, GKMD06, SM06]. Our method seamlessly supports arbitrary BRDFs, bump mapping, and all-frequency lighting combining the advantages of a number of previous methods.

Generally, lighting in PRT is assumed to be distant. However, low-frequency localized lighting can be integrated [AKDS04]. When only indirect illumination is considered, local point and spot lights are possible [KAMJ05, KTHS06, HPB06]. In contrast, our technique supports all kinds of incident lighting, including area sources and light field illumination [GGHS03]).

Dynamic Precomputed Radiance Transfer Dynamic scenes are inherently difficult for PRT techniques, since visibility cannot be precomputed anymore. Mei et al. [MSW04] precompute visibility on a per-object basis for a discrete set of directions and store it in the form of uncompressed shadow maps. This allows them to render multiple rigidly moving objects under low-frequency distant illumination. However, dynamic local light sources remain infeasible. Similar in spirit, Zhou et al. [ZHL*05] propose the use of shadow fields, that allow to move individual rigid objects under semi-local (lights may not enter an object's bounding sphere) or distant illumination, producing correct inter-shadowing. In practice, this technique was limited to low-frequency lighting, as all-frequency lighting updates took several seconds for dynamic scenes.

Related to the previous two techniques, intersection fields [RHCB05] store visibility information along lines in space. For efficiency, shadow computation is separated from illumination computation (in the spirit of ambient occlusion), which prohibits the use of large area sources. No compression is used and memory consumption is very high. Sun et al. [SM06] extend wavelet product integrals in order to separate local and global visibility, which also enables rigidly moving objects. However, interactive updates are limited to fixed view and lighting. In the case of dynamic objects, only low-frequency visibility information can be used that has to be recomputed for every frame.

Our method combines the advantages of previous work. We support all-frequency lighting, dynamically moving rigid objects, local as well as distant light sources, and dynamic material properties. Nonetheless, the method is fast, supports progressive rendering for even faster updates, and is memory-efficient through the use of coherent shadow maps.

Fully dynamic objects are very difficult to handle, as no precomputation can be employed. Hemispherical rasterization [KLA04] and spherical harmonics exponentiation [RWS*06] have been proposed for small- or medium-sized scenes under low-frequency-illumination. The focus of our work is on high-frequency illumination of dynamic, rigid objects.

Shadows Many techniques exist for handling shadows in interactive applications, including the widely used shadow map [Wil78] and shadow volume [Cro77] algorithms. Much work has been done to extend them to soft shadow generation, e.g. [AAM03, GBP06] to name a few, see [HLHS03] for a survey. These real-time soft shadow algorithms are generally suitable for deformable objects. However, they only provide approximate soft shadows to speed up shadow generation. Furthermore, it is worth noting that none of these methods provides accurate integration of BRDF and incident lighting. Instead, it is assumed that the integral of products equals the product of integrals.

We use a data structure that is based on shadow maps, but enhance it such that it can be used for physically-based real-time rendering. Our representation is suitable only for rigid dynamic objects but does provide accurate illumination computation, including soft shadowing, partial and high-frequency glossy reflections, and so on.

Direct illumination from environment maps using a small number of *uncompressed* orthographic shadow maps has been proposed by Havran et al. [HSK*05], which forces visibility information to be heavily discretized. A compression scheme for a single shadow map was introduced by Arvo and Hirvikorpi [AH05]. In contrast to their work, we exploit coherence between many shadow maps for compression, not within a single shadow map.

Monte Carlo We employ Monte Carlo integration [PH04] to accurately compute direct illumination in real-time. Many different strategies exist for enhancing Monte Carlo integration in computer graphics. These strategies, such as efficient sampling for image-based lighting [KK03, ARBJ03, ODJ04, Deb05] or direct sampling of product distributions [VG95, BGH05, TCE05, CJAMJ05, CETC06], are (theoretically) orthogonal to our technique. However, in practice many of these methods are difficult to use/implement on graphics hardware or require an expensive pre-processing step and are therefore not used in this work. Of course, we do make use of importance sampling [PH04].

3. Coherent Shadow Maps

In physically-based rendering, most time is spent on visibility queries. But in case of rigid objects, the query time can be greatly reduced by precomputing visibility. We seek to efficiently store the visibility information around an object in order to evaluate the direct lighting equation (integral of BRDF, lighting, and visibility) as quickly as possible. However, this requires storing the full field of visibility information, such that visibility queries can be performed at every point on the object and for all possible directions.

This poses two main challenges: First, the memory cost for storing the information at an adequate resolution is prohibitive and compression is therefore mandatory. Secondly, discretization introduces aliasing that requires appropriate filtering.

Central to our approach is a lossless compressed representation for visibility based on orthographic or perspective depth maps. We propose to discretize and sample the visibility space by placing a large number of depth maps in the scene, that can be efficiently queried on a GPU at run-time. The depth maps are compressed by exploiting the coherence between neighboring depth maps, as we will detail in the next subsection (Sec. 3.1).

The actual discretization, i. e., the placement and resolution of depth maps, depends on the type of supported light sources and the minimum size of geometric details, which we explain in Section 4 and 4.4. Fig. 2 shows an example of such a discretization. This set of depth maps can be used for visibility queries, as illustrated in Fig. 6. Aliasing can be reduced using a generalized percentage-closer filter or Russian Roulette (Sec. 4.5).

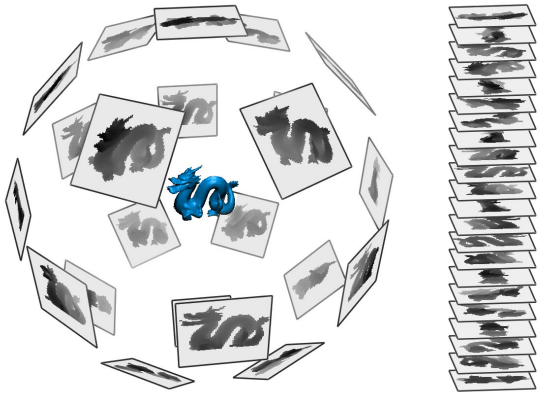


Figure 2: *Discretization example: An orthographic depth map is created for N viewing directions (left). All depth maps are stored in a sequence (right).*

3.1. Compression

In the following, we consider a set of N depth maps, each with a resolution of $M \times M$ pixels. To obtain a high compression rate, a sequence of *coherent* depth maps is required. Two successive depth maps are coherent, if most pixels contain similar depth values. Therefore, the first step of our algorithm creates a highly coherent permutation of the original depth maps. This can be visualized as moving a depth camera through a scene, trying to keep the depth images as similar as possible. Different ways to create such a sequence are described in Section 4.4; for now we assume that the sequence $i = 1 \ldots N$ of depth maps is sorted in a coherent order.

In the following, we consider a pixel at a fixed location (x, y) in an arbitrary depth map i. For clarity, we drop the position index and denote the depth values as $z(i)$ instead of $z_{xy}(i)$. For our compression method, the depth values $z_1(i)$ and $z_2(i)$ of the first and second intersection point of a ray through x, y are required. As noted by Weiskopf and Ertl [WE03], the depth comparison works, as long as the depth map contains a value $z \in [z_1, z_2]$ as shown in Fig. 3. This degree of freedom

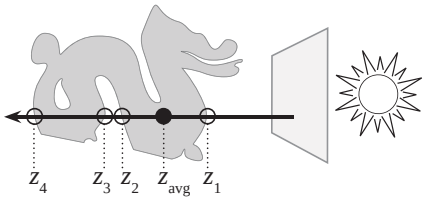


Figure 3: *Dual depth values: A standard depth map stores z_1, the smallest depth value. An arbitrary value z_{avg} between z_1 and z_2 can be used: The depth comparsion of $z_{1\ldots4}$ to z_{avg} still gives the correct result.*

is exploited by using an arbitrary depth function $z(i)$ between

$z_1(i)$ and $z_2(i)$ that is well suited for compression. A simple choice are piecewise constant functions (*segments*). To generate the depth value $z_{avg}(0)$ of the first segment, the largest index i_{end} is determined, such that

$$\max\{z_1(1), z_1(2), \ldots, z_1(i_{end})\} < \min\{z_2(1), z_2(2), \ldots, z_2(i_{end})\}. \quad (1)$$

Intuitively, we are searching for the index i_{end} such that the $z_1(i_{end})$-value is as large as the minimal z_2-value in that range, see Figure 4. Then $z_{avg}(0)$ is set to the average of the min and max-value, as can be seen in Fig. 4. This process is repeated for the following depth values, resulting in n segments. For each segment j, the depth value $z_{avg}(j)$ and the end index $i_{end}(j)$ are stored sequentially in a *segment list*. In this way, we separate the two curves with a small number of horizontal lines. This process is repeated for all pixels of the depth map, resulting in $M \times M$ segment lists.

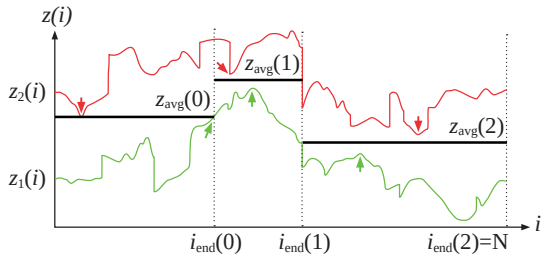


Figure 4: *Depth compression: The lower (green) curve shows a depth pixel $z_1(i)$ at a fixed position, depending on the depth map index i. The upper (red) curve is the depth of the second intersection point $z_2(i)$. The minimum and maximum values are denoted with arrows. For a visibility test, the depth values of this pixel can be represented by a list of only three segments: $(z_{avg}(0), i_{end}(0)), (z_{avg}(1), i_{end}(1))$ and $(z_{avg}(2), i_{end}(2) = N)$.*

Details on Depth Values There are different ways to choose z_{avg}. We usually use the depth values of the first and second intersection point, as shown in Figure 4. Assuming a closed two-manifold surface, no correct shadow on the back of an object is required, because the BRDF is zero anyway and the depth values of the first z_1 and third z_3 intersection point can be used instead. If there is no third intersection, there are two possibilities. One can simply use a large value for z_3, which allows for better compression as z_1 and z_3 are far apart and z_{avg} can be chosen more freely leading to longer segments. However, large z_3-values are likely to lead to large z_{avg} depth values, which are problematic for dynamic scenes with multiple objects (see Section 4.6). In this case other objects might be incorrectly lit. In that case, it is better to use a *spherical far plane* that clips depth values z_3 to the bounding sphere. However, freedom of choosing z_{avg} is reduced here, and compression rates are lower. We use linear depth values in all cases [BAS02a].

3.2. Depth Comparison Using CSMs

Using the compressed depth map for visibility testing from location $\mathbf{p}$ in a depth map i works like classic shadow mapping. First, the point $\mathbf{p}$ is projected into the coordinate space of the i-th depth map by transforming it with the *depth map matrix* $\mathbf{D}_i$ of depth map i: $\mathbf{q} = \mathbf{D}_i\mathbf{p}$. After perspective division, the local pixel position $\mathbf{q}_{xy}$ and the depth value $\mathbf{q}_z$ of the point $\mathbf{p}$ for depth map i are known. To determine the depth value z_{xy} stored at position $\mathbf{q}_{xy}$, the segment j that contains i has to be found:

$$i_{end}(j-1) < i \leq i_{end}(j).$$

Because the sequence of i_{end} values is monotonically increasing, a binary search to locate segment j in $O(\log n)$ is possible. Finally, $z_{xy} = z_{avg}(j)$ is used for a depth comparison with $\mathbf{q}_z$.

3.3. Depth Cube Maps

A different form of depth maps are *depth cube maps* [BAS02b], which allow to store depth information from a point in all directions by storing six perspective depth maps into six sides of a cube map. To compress a set of N depth

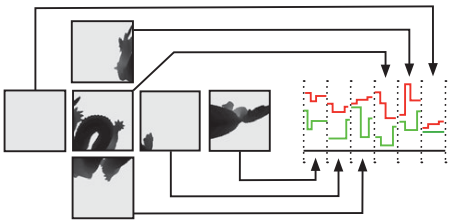


Figure 5: *Concatenation of depth functions for depth cube maps.*

cube maps, the depth functions $z(i)$ for each pixel with length N of the six individual cube sides are concatenated to a list of length $6N$ (cf. Fig. 5). To decompress a depth value in direction $\mathbf{d}$ for depth cube map i, first the cube side index $k = 0\ldots5$ is determined from $\mathbf{d}$ and then the depth value $z(kN+i)$ is used.

4. Light Source Visibility Tests Using CSMs

For a visibility test, we differentiate between two different types of the CSM data structure:

- CSM based on orthographic depth maps
- CSM based on perspective depth cube maps

The CSM type to use depends on the types of light sources that we support: *Infinite distant lights*, *local lights* and so-called *semi-local lights*. Although each type of light has a slightly different technique for visibility calculations, their common ground is the use of precomputed depth maps.

Mapping Function As a general description, each type of light implements a continuous function $\mathcal{D}(\mathbf{x})$ which maps a d-dimensional parameter vector $[0\dots1]^d$ to the location of a depth map in $\mathbb{R}^4$. The number of required dimensions d depends on the type of light, as described in the following subsections. To distinguish between positions and directions, we describe the location of a depth map as $(x,y,z,w) \in \mathbb{R}^4$. If $w=0$, the depth map of an orthographic camera, looking in direction (x,y,z) is used. If $w=1$, the center of the camera is placed at (x,y,z) and a perspective depth cube map is generated. A visibility test can now be computed between $\mathcal{D}(\mathbf{x})$ and arbitrary points $\mathbf{p} \in \mathbb{R}^3$. In most cases, $\mathcal{D}(\mathbf{x})$ is a point on the surface of a light and $\mathbf{p}$ is a point on the surface of an object. For compression, a discretization $\mathcal{D}(\mathbf{x}_i)$ is used to generate the N depth maps. All visibility queries select one of the precomputed depth maps — no depth maps are generated during rendering.

4.1. Infinite Distant Lights

The typical application for infinite distant lights is the rendering with natural light, captured in an environment map, as shown in Fig. 9. Here $d=2$ and $\mathcal{D}$ is the standard lat-long mapping from spherical coordinates to cartesian directions $(w=0)$. Visually speaking, an orthographic camera is placed at different locations on the bounding sphere of the object, looking at the center. The extends of the orthographic projection are adjusted to the extend of the bounding sphere. A visibility test between a point $\mathbf{p}$ in a direction $\mathbf{d}$ can be performed based on the CSM, as described in Fig. 6. The artifacts introduced by snapping to a direction $\mathcal{D}(\mathbf{x}_i)$ of the discretization are discussed in Section 4.5.

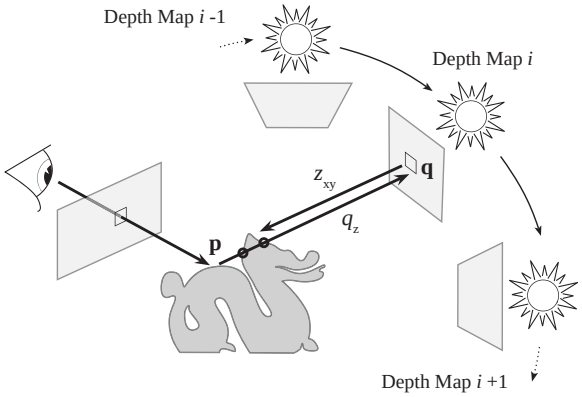


Figure 6: *Visibility test for infinite distant light: To test if a ray, starting from point* $\mathbf{p}$ *in direction* $\mathbf{d}$*, intersects any object, we first select a precomputed (orthographic) depth map i with a viewing direction most similar to* $\mathbf{d}$*. Now we compute the depth map matrix* $\mathbf{D}$ *and project* $\mathbf{p}$ *as* $\mathbf{q} = \mathbf{Dp}$*. The ray is blocked, if* $\mathbf{q}_z$ *(the depth of* $\mathbf{p}$ *in view i) is larger than* z_{xy} *(the depth value stored in the precomputed depth map).*

4.2. Local Lights

Local lights can be placed at arbitrary positions, even close to the surface of an object. Therefore, depth cube maps are used to view each part of the object from the light position. The mapping $\mathcal{D}$ computes points on the surface (or inside) the light source $(w=1)$. The dimension d depends on the shape of the light source: Linear lights $(d=1)$, area lights $(d=2)$ or volumetric lights $(d=3)$ are possible choices. Local lights can have an arbitrary shape and radiance distribution but their position has to be fixed. However, variation in intensity distribution can be used to simulate dynamic lights. Similar to distant lights, a visibility test is performed by comparing the depth value of a point with the depth value stored in a depth cube map, as can be seen in Fig. 7. Fig. 1 shows an example of illumination from a linear local light.

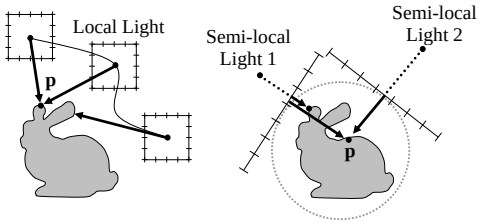


Figure 7: *Left: Fixed local light sources. For a set of sampling points on the surface of the light, a depth cube map is created. Right: Semi-local light sources. To test whether a point is visible, the orthographic depth map perpendicular to the light direction is selected.*

4.3. Semi-Local Lights

To remove the restriction of a fixed light position, we introduce the concept of *semi-local* light sources. The basic idea for this type of light is to re-use the information stored in the orthographic depth maps: To test if there is any occluder between a point $\mathbf{p}$ and a point $\mathbf{l}$ on the light source, we compute the direction vector $\mathbf{d} = \mathbf{l} - \mathbf{p}$ and perform the same visibility test as described for infinite distant lights (cf. Fig. 7). Therefore, $\mathcal{D}$ and $\mathbf{d}$ are identical to infinite distant light sources, but an additional description of the actual shape of the light is required, as described in Section 5. Semi-local lights can be placed at arbitrary positions outside the *convex hull* of the object, because the CSM contains only the depth information when viewing the object from *outside*. A light source in the convex hull would require a depth map generated at the position of the light source to obtain the depth information *inside* the object. Beside this limitation, semi-local lights contain all the abilities of local lights.

4.4. Enumeration and Discretization

A discretization divides the continuous d-dimensional domain of $\mathcal{D}$ into $N = \prod_{k=1}^{d} N_k$ depth map locations. Here, a

uniform or an adaptive discretization can be chosen. We use a uniform discretization in all our examples. An enumeration is a bijective mapping from this discretization to $\{1,\ldots,N\}$. The simplest enumeration is *scan-line* enumeration: First all depth maps in one dimension are concatenated. Afterwards, the next dimension is added, and so on. To obtain a high coherence, we use a zig-zag or hilbert space filling curve (cf. Fig. 8). We refer to Section 6 for different compression rates with different strategies.

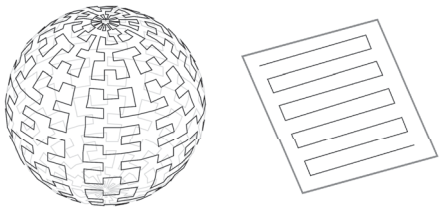


Figure 8: *The distant Hilbert (left) and zig-zag area (right) curve*

4.5. Filtering

Like most sampling-based approaches, shadow mapping suffers from aliasing. In our case this results in jagged shadow boundaries when using a low depth map resolution M and banding artifacts (cf. Fig. 9) when using a small number of depth maps N. We propose two different filtering techniques to reduce the aliasing artifacts that arise from the spatial and angular discretization: *Generalized PCF* (Percentage Closer Filtering) and *Russian Roulette*. Generalized PCF selects the nearest 2^d depth maps (angularly) and performs four spatial occlusion tests with each, giving 2^{d+2} depth comparisons. A weighted sum of of each depth comparison's binary result is computed and returned as the shadow value [RSC87]; the weights are based on the fractional error from nearest neighbor sampling. To avoid this time-consuming process, Russian Roulette can be used to randomly select one out of 2^{d+2} possible spatial and angular combinations. The weights computed for generalized PCF interpolation serve as probabilities for this selection. Russian Roulette filtering converges to the same result as the d-linear filtering from generalized PCF, but a visually pleasing result is obtained faster. Fig. 9 shows a comparison between the different techniques.

4.6. Extension to Multiple Objects

Although we assume static geometry, multiple *moving* objects are possible. To this end, we build one CSM for each object. Objects illuminated by distant or semi-local lights can then be moved, rotated and scaled freely (Fig. 1). To decide whether a point **p** is visible from a semi-local or distant light source, we first test if the point is occluded by the object itself. In case of no occlusion, we inspect the depth maps of the other objects. Fig. 10 shows an example with three objects. To determine

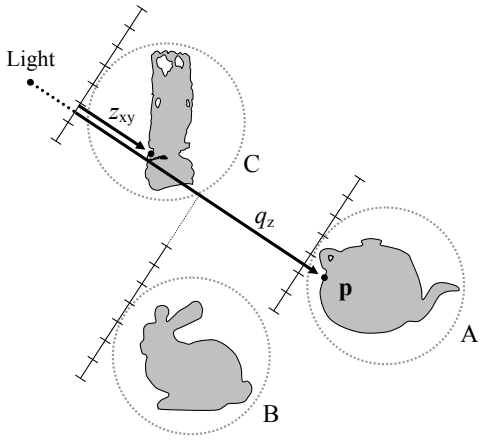


Figure 10: *Moving objects: To test if there is any occlusion between point* **p** *(on object A) and the light source, we first compare the depth values of object A. Because there is no self-occlusion, we inspect the depth values of the other objects. For each object, we select the depth map perpendicular to the light direction and compare the two depth values. Object B does not occlude* **p** *because the pixel is outside its depth map. In the depth map of object C, we compute two different depth values: Object C occludes* **p**.

the correct depth values, the original depth map matrix $\mathbf{D}$ of the moving object has to be modified: If the moving object is transformed by a matrix $\mathbf{T}$, the depth map matrix must be replaced by $\mathbf{D}' = \mathbf{D}\mathbf{T}^{-1}$.

Receiver-Only Objects If an object is only receiving a shadow (like a ground plane, which is not generating a shadow on other objects), no CSM is required for this object at all. The reason is, that for each pixel of this object, we only have to decide whether it is shadow or not. Therefore, we compare the depth values of all CSMs with the depth value of the current pixel (like for moving objects). If the object is planar (or at least convex), we can ignore self-shadowing of the receiving object and omit the CSM. This even allows for shadow-tests for points in a volume.

5. Illumination using CSMs

This section describes the GPU implementation of a Monte Carlo renderer using coherent shadow maps which solves the direct lighting equation:

$$L(\mathbf{x},\omega_o) = \int_\Omega f_r(\mathbf{x},\omega_o,\omega) L_{in}(\mathbf{x},\omega) V(\mathbf{x},\omega)\cos(\theta)\mathrm{d}\omega$$

where f_r is the BRDF and L_{in} is the incoming radiance from direction ω. The binary visibility term V can be efficiently evaluated based on the CSM. Rasterization is done in a conventional way, but a Monte Carlo simulation is run for each pixel in a fragment program. As no interpolation between

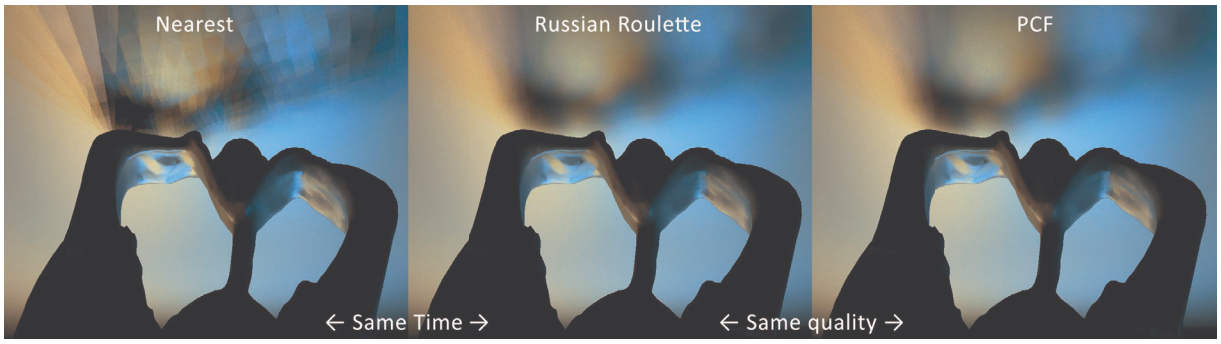


Figure 9: *Comparing different filters to reduce aliasing: PCF (3.3 minutes) gives the same quality as Russian Roulette (15.4 seconds) that requires nearly the same time as nearest filtering (11.5 seconds). It shows, that PCF is between 10 and 15 times slower, because it always decodes 16 depth values. To exaggerate the effect of aliasing the discretization is set artficially low and the number of samples is artficially high in this figure.*

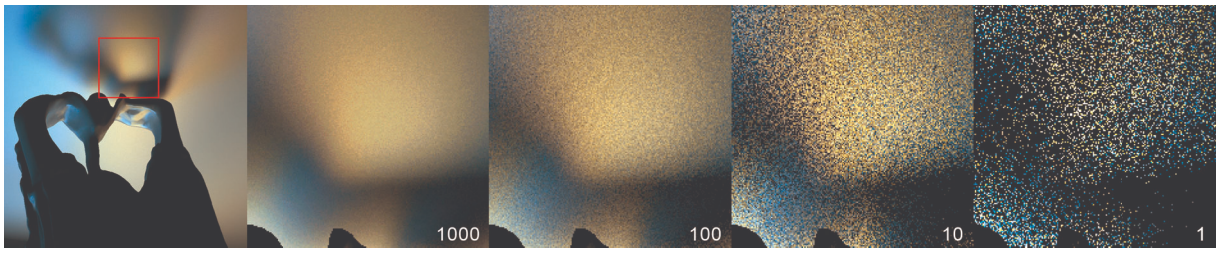


Figure 11: *Progressive rendering (1024 × 768) with a decreasing number of samples per pixel (11.2 s, 1.2 s, 207 ms and 30 ms).*

vertices is used, the shading parameters can be adjusted per pixel by a texture and change over time, just as in any other Monte Carlo renderer. We have successfully used diffuse, specular and normal maps as demonstrated in Fig. 1.

Visibility Queries As described in Section 3, depth values are stored in compressed form as segment lists. For each visibility test, the segment containing the index i of the selected depth map has to be found. The concatenation of all segment lists is stored sequentially in a texture. To locate the correct entry point for the segment list describing the depth values of pixel q_{xy}, we use a $M \times M$ lookup texture that maps the pixel position q_{xy} to the starting point of the corresponding segment list. The segment containing i is then found with binary search, requiring $\log(n)$ texture reads, and some conditional adds.

To locate the segment containing i in a depth cube map at direction $\mathbf{d}$ requires two steps. First, a cube map of size 1×1 is used to map $\mathbf{d}$ to an index $k = 0 \ldots 5$. Secondly, the segment containing $kN + i$ is found in the same way as for orthographic depth maps.

Lights In addition to $\mathcal{D}$, each light is defined by two functions: $\mathcal{A} : [0 \ldots 1]^{d'} \rightarrow \mathbb{R}^4$ describes the surface of the light source. For distant and local lights, $\mathcal{A}$ is identical to $\mathcal{D}$ and $d = d'$. For semi-local light sources, $\mathcal{A}$ is an arbitrary shape (with arbitrary dimension d') and $\mathcal{D}$ is the standard lat-long mapping. To describe a spatially-varying radiance distribution, a function $\mathcal{L} : [0 \ldots 1]^{d'} \rightarrow \mathbb{R}^3$ is used which maps the parameter vector to an RGB-value (L_{in}). Here, a (video-) environment map with natural lighting, a synthetic sky, or any other pattern can be used. Both functions may freely vary over time. Additionally, we allow non-uniform emitters, such as Lambertian or directional emitters. Currently, different versions of $\mathcal{A}$, $\mathcal{D}$ and $\mathcal{L}$ are stored in textures with linear filtering (See Fig. 12).

5.1. Sampling

Importance Sampling We use importance sampling to solve the direct lighting equation with a set of S incoming directions ω_k, generated from a probability density function p:

$$L(\mathbf{x}, \omega_o) \approx \frac{1}{S} \sum_{k=1}^{S} \frac{f_r(\mathbf{x}, \omega_o, \omega_k) L_{in}(\mathbf{x}, \omega_k) V(\mathbf{x}, \omega_k) \cos(\theta_k)}{p(\omega_k)}$$

Therefore, the inverse cumulative density function (CDF) of the incoming radiance P_L^{-1} and the BRDF P_f^{-1} are precomputed and stored in textures. To generate samples proportional to a density p, we read the corresponding inverse CDF texture at a uniform random location. For interactive manipulations

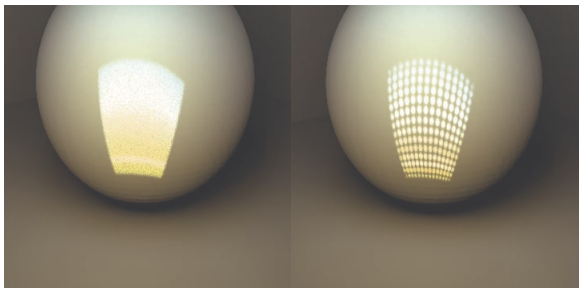


Figure 12: *Our implementation uses linear filtering for $\mathcal{A}$, $\mathcal{D}$ and $\mathcal{L}$ that results in a continuous highlight (Left). A discretization into point lights would result in multiple isolated highlights (Right).*

of materials, different versions of P_f^{-1} for Phong BRDFs with increasing glossiness are stored. For a Lafortune BRDF, we preintegrate the single lobes and select one of the lobes with Russian Roulette. Then, the same sampling strategy as described for Phong BRDFs is used. Integrating more sophisticated BRDFs would also be possible.

Random Numbers We use a stratified random pattern for sampling. In contrast to more sophisticated patterns [PH04], a stratified pattern benefits from coherent memory access, which is crucial for GPUs and increases performance by a substantial factor of 1.2 to 1.5. For each pixel, one out of 256 different pre-generated two-dimensional patterns of length S is used. The patterns are stored into a texture of size $256 \times S$. The sampling pattern to use for each pixel depends on the pixel position. To switch the pattern for each frame, it is also chosen based on another quasi random index, stored in a texture with the same resolution as the framebuffer.

Sampling Distant Illumination For distant illumination, samples can be drawn proportional to f_r or to the radiance distribution L_{in} stored in the environment map. To reduce the amount of noise, different strategies are applied for diffuse and specular surfaces [PH04]: For diffuse surfaces P_L^{-1} is used while specular surfaces always use P_f^{-1}. If a surface is both diffuse and specular, the decision for a diffuse or specular sample is implemented by Russian Roulette based on the material roughness k_d.

Sampling Local Illumination For local and semi-local light sources, samples are always drawn proportional to their radiance distribution $\mathcal{L}$, because samples selected from the BRDF do not hit the light source in most cases.

5.2. Progressive Rendering

Reducing the noise introduced by Monte Carlo rendering is straightforward: All parameters are kept fixed and the average of a few successive frames is computed (cf. Fig. 11). Therefore, a number of m frames is rendered and blended together weighting each frame's contribution by $\frac{1}{m}$. This is a clear advantage over other methods that prevent noise by band-limiting the signals. A user can navigate through the scene or manipulate it at high interactive rates. After the manipulation, the image converges to an exact solution over time.

6. Results

We demonstrate our approach for different scenes and types of lightsources, as can be seen in the accompanying video. The system used is an Intel Core 2 Duo 6300 with 2 GB RAM and an NVIDIA Geforce 8800. Compression and timing results are summarized in Table 1 and Table 2.

Compression Rates As can be seen in Table 1, the compression rate increases with N: It grows roughly with the square root of the number of depth maps N. This is, because CSMs 'extract' the relevant depth information from a large number of depth maps. Note, that the visibility information from more than a million depth maps of the Buddha model (that require several minutes to render) can be squeezed into a single GPU texture. The average number of segments required to compress a depth function $z(i)$ depends on the geometric complexity of the object. Objects similar to a sphere (only one segment) are well suited, while objects with a strong variation in depth values and thin surfaces $(z_1 \approx z_2)$ have smaller compression ratios. As expected, a higher compression is achieved when using the Hilbert enumeration (row three and four).

Quality When using shadow maps, the most interesting question is how the spatial and angular discretization affects the image quality (see Fig. 13 and Fig. 14). A finite number of shadow maps creates banding while a finite resolution for each shadow map prevents fine geometric details to cast shadows. However, the resulting jagged shadow boundaries and banding artifacts are less visible for realistic natural illumination and area light sources than for synthetic point lights, especially in combination with filtering. Besides filtering, the display quality can be improved by using more depth maps. Due to the good compression behaviour (Table 1), large values for N are possible; this only increases the precomputation time, which grows linear with N. Because our compression is based on [WE03], self-shadowing artifacts are mostly removed. However, a depth bias is still required for pixels which start a new segment $(z_1 = z_2)$.

Several techniques (e.g., [MSW04]) have parameterized the visibility function over the surface as a discrete binary function of direction for each vertex. We do not consider this option, as there is no obvious way to recombine such representations when multiple occluders are used. Furthermore, they require a fine tessellation of all receivers while our technique does not.

Speed A higher compression ratio also decreases the time for a visibility query, because fewer operations are required to

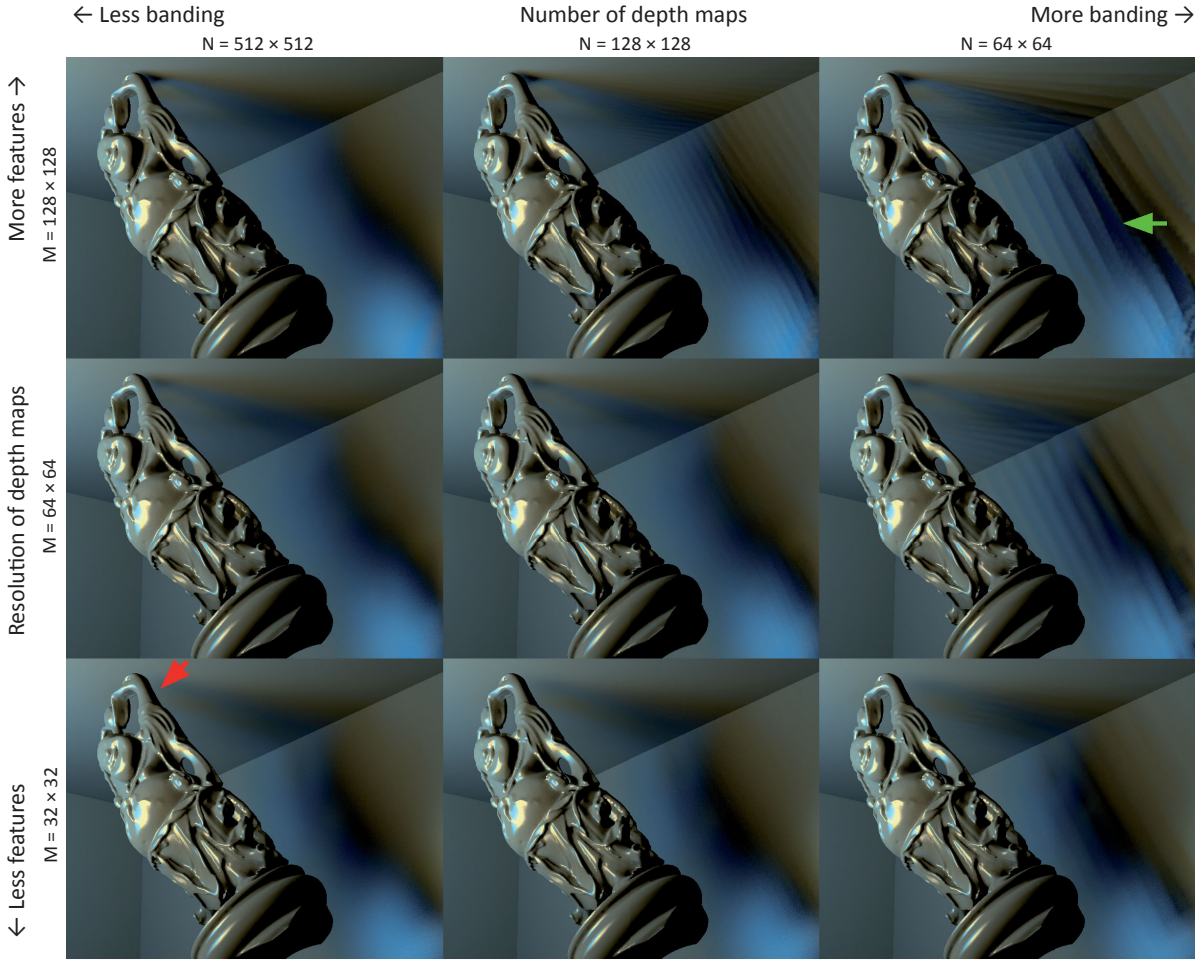


Figure 13: *Discretization error for converged images. CSMs introduce two forms of discretization: A finite number of shadow maps creates banding (right, top image) while a finite resolution for each shadow map prevents fine geometric details to cast shadows (left, bottom image).*

find the location of a segment. However, from our experience this has only a small impact on the rendering speed due to the logarithmic time complexity of the segment search. As can be seen in Table 2, the total rendering time for an 1024×768 image with 625 samples is 4.86 s, which means at least 101.1 M monte carlo samples and (non-coherent) shadow tests per second. This is an order of magnitude faster than the number of coherent intersection tests of current ray tracing systems for comparable scene complexity [BWSF06]. However, our approach is only an approximation due to the discretization.

7. Conclusions

We have developed a method for fast visibility queries based on compressed shadow maps. We have proposed to sample visibility around an object by precomputing a large number of depth maps in an offline process. The large amount of memory that is required for capturing all the visibility details is reduced by a new shadow map compression method. At run-time, visibility queries are performed by indexing into the precomputed and compressed depth maps.

This new visibility test is used to perform interactive illumination on the GPU. We have demonstrated that Monte Carlo rendering using this visibility test is well-suited for graphics hardware, enabling interactive display of complex local illumination effects that were previously impossible. Our illumination method is not band-limited, it converges to the correct solution when given enough time, and the frame rate does not depend on scene complexity. While our method assumes rigid objects with dynamic placement, all light sources and shift-variant BRDFs can be changed on-the-fly with no overhead. Furthermore, we have developed filtering techniques

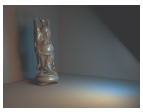
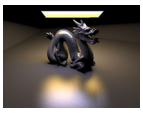
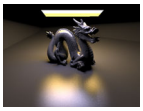
Scene		Depth Maps (N / total)		Uncompressed	Compressed	Time	Ratio
Distant Lat-long Buddha		32×32	1024	32.0 MB	3.0 MB	3 s	10.7 : 1
		128×128	16384	512.0 MB	18.7 MB	46 s	27.4 : 1
		512×512	262144	8.1 GB	96.8 MB	744 s	85.3 : 1
		1024×1024	1048576	32.7 GB	202.1 MB	2253 s	162.1 : 1
Distant Lat-long XYZ Dragon		32×32	1024	32.0 MB	5.4 MB	5 s	5.0 : 1
		128×128	16384	512.0 MB	44.2 MB	281 s	11.5 : 1
		512×512	262144	8.0 GB	242.8 MB	1094 s	33.9 : 1
Distant Lat-long Bunny		32×32	1024	32.0 MB	3.9 MB	4 s	8.2 : 1
		128×128	16384	512.0 MB	19.9 MB	50 s	25.4 : 1
		512×512	262144	8.0 GB	242.8 MB	844 s	86.2 : 1
Distant Lat-long Dragon		32×32	1024	32.0 MB	8.3 MB	3 s	3.9 : 1
		128×128	16384	512.0 MB	50.0 MB	46 s	10.2 : 1
		512×512	262144	8.0 GB	249.8 MB	1817 s	32.9 : 1
Local Scanline Dragon		8×8	384	12.0 MB	2.3 MB	3 s	5.2 : 1
		32×32	6144	192.0 MB	13.1 MB	16 s	14.7 : 1
		128×128	98304	3.0 GB	63.8 MB	256 s	48.2 : 1
		256×256	393216	12.2 GB	137.1 MB	1355 s	89.6 : 1
Local Hilbert Dragon		8×8	384	12.0 MB	1.9 MB	3 s	6.3 : 1
		32×32	6144	192.0 MB	10.0 MB	15 s	19.2 : 1
		128×128	98304	3.0 GB	43.0 MB	261 s	71.4 : 1
		256×256	393216	12.2 GB	90.1 MB	1344 s	136.4 : 1
Local Hilbert Plant		32×32	6144	192 MB	4.5 MB	13 s	42 : 1
		64×64	24576	768 MB	8.0 MB	62 s	96 : 1
		128×128	98304	3.0 GB	15.6 MB	255 s	170 : 1
		256×256	393216	12.2 GB	29.4 MB	988 s	418 : 1

Table 1: *Compression rates for different objects with shadow map resolution $M = 128$. For local lights, the total number of depth maps is $6N$ (depth cube maps).*

to remove discretization artifacts introduced by the shadow maps.

There are several directions for future research. Depth map compression could be performed on the GPU, possibly even at near-interactive rates for simple scenes. We would like to consider global illumination by distributing point-lights over an object's surface, which requires to arrange them in a coherent order that is suited for compression. Alternative search methods for segment location, e. g., hash tables, are an interesting avenue of research. Finally, we want to investigate adaptive methods for CSM creation that minimize the number of required depth maps for a given scene.

Acknowledgments We would like to thank M. Geimer, M. Biedermann, R. Trappe and A. Langs for proofreading the paper and S. Pohl for the chair model. The 3D models are courtesy of Stanford University. The light probe images are courtesy of P. Debevec.

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware. *ACM Trans. Graph. 22*, 3 (July 2003), 511–520.

[AH05] ARVO J., HIRVIKORPI M.: Compressed Shadow Maps. *Vis. Comput. 21*, 3 (2005), 125–138.

[AKDS04] ANNEN T., KAUTZ J., DURAND F., SEIDEL H.-P.: Spherical Harmonic Gradients for Mid-Range Illumination. In *15th Eurographics Symposium on Rendering* (2004), pp. 331–336.

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured Importance Sampling of Environment Maps. *ACM Trans. Graph 22*, 3 (2003), 605–612.

[BAS02a] BRABEC S., ANNEN T., SEIDEL H.-P.: Practical shadow mapping. *Journal of Graphics Tools 7*, 4 (2002), 9–18.

[BAS02b] BRABEC S., ANNEN T., SEIDEL H.-P.: Shadow Mapping for Hemispherical and Omnidirectional Light Sources. In *Advances in Modelling, Animation and Rendering (Proceedings Computer Graphics International 2002)* (Bradford, UK, 2002), Vince J., Earnshaw R., (Eds.), Springer, pp. 397–408.

[BGH05] BURKE D., GHOSH A., HEIDRICH W.: Bidirectional Importance Sampling for Direct Illumination. In *16th Eurographics Symposium on Rendering* (2005), pp. 147–156.

[BWSF06] BENTHIN C., WALD I., SCHERBAUM M., FRIEDRICH

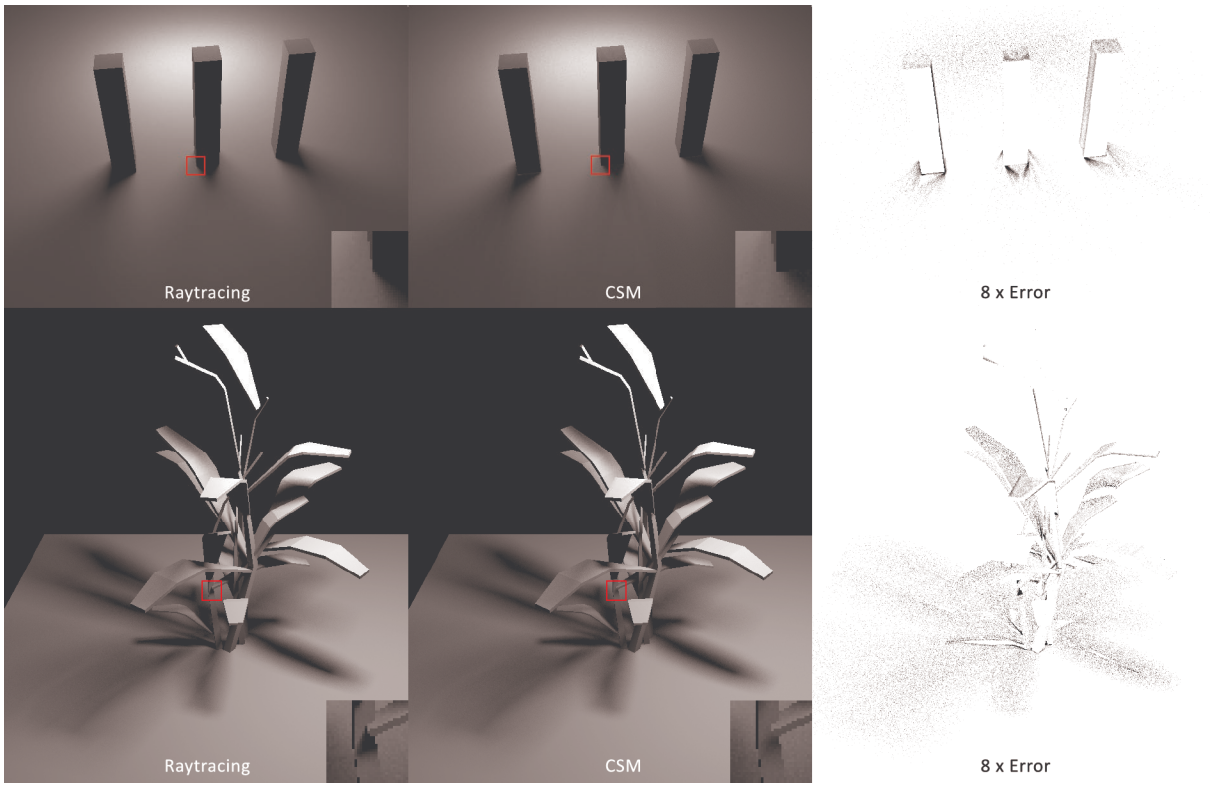


Figure 14: *Comparsion (right) between raytracing (left) and CSMs (middle). The depth map discretization can only approximate a continuous location of sharp features (eg. the cube's corner) with a discrete location, which results in slightly displaced shadows (red square). The top row uses a distant CSM and the plant a local CSM.*

H.: Ray Tracing on the CELL Processor. In *IEEE Symposium on Interactive Ray Tracing* (2006), pp. 25–23.

[CETC06] CLINE D., EGBERT P. K., TALBOT J. F., CARDON D. L.: Two Stage Importance Sampling for Direct Lighting. In *17th Eurographics Symposium on Rendering* (2006), pp. 103–114.

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. *ACM Trans. Graph. 24*, 3 (Aug. 2005), 1166–1175.

[Cro77] CROW F.: Shadow Algorithms for Computer Graphics. In *Proceedings of ACM SIGGRAPH* (July 1977), pp. 242–248.

[Deb05] DEBEVEC P.: A Median Cut Algorithm for Light Probe Sampling. Poster at SIGGRAPH 2005, 2005.

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Real-time Soft Shadow Mapping by Backprojection. In *17th Eurographics Symposium on Rendering* (2006), pp. 227–234.

[GGHS03] GOESELE M., GRANIER X., HEIDRICH W., SEIDEL H.-P.: Accurate Light Source Acquisition and Rendering. *ACM Trans. Graph. 22*, 3 (July 2003), 621–630.

[GKMD06] GREEN P., KAUTZ J., MATUSIK W., DURAND F.: View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proceedings of ACM Symposium in Interactive 3D Graphics and Games* (Mar. 2006), pp. 7–14.

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of Real-Time Soft Shadows Algorithms. *Computer Graphics Forum 22*, 4 (Dec. 2003), 753–774.

[HPB06] HASAN M., PELLACINI F., BALA K.: Direct-to-Indirect Transfer for Cinematic Relighting. *ACM Trans. Graph. 25*, 3 (2006), 1089–1097.

[HSK*05] HAVRAN V., SMYK M., KRAWCZYK G., MYSZKOWSKI K., SEIDEL H.-P.: Importance Sampling for Video Environment Maps. In *16th Eurographics Symposium on Rendering* (Konstanz, Germany, 2005), Bala K., Dutré P., (Eds.), ACM SIGGRAPH, pp. 31–42.

[KAMJ05] KRISTENSEN A. W., AKENINE-MÖLLER T., JENSEN H. W.: Precomputed Local Radiance Transfer for Real-Time Lighting Design. *ACM Trans. Graph. 24*, 3 (2005), 1208–1215.

[KK03] KOLLIG T., KELLER A.: Efficient Illumination by High Dynamic Range Images. In *14th Eurographics Workshop on Rendering* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 45–50.

[KLA04] KAUTZ J., LEHTINEN J., AILA T.: Hemispherical Rasterization for Self-Shadowing of Dynamic Objects. In *15th Eurographics Symposium on Rendering* (June 2004), pp. 179–184.

[KTHS06] KONTKANEN J., TURQUIN E., HOLZSCHUCH N., SILLION F. X.: Wavelet Radiance Transport for Interactive In-

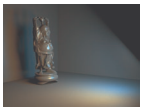
Scene		Faces	Resolution	S	Time	MS / s
Local Linear XYZ Dragon		280k	320×240	16	45.0 ms	27.3
			1024×768	16	170.0 ms	74.2
			1024×768	64	600.0 ms	83.8
			1024×768	625	4860.0 ms	101.1
Distant Lat-long Buddha		80k	1024×768	16	190.0 ms	66.2
			1024×768	64	720.4 ms	69.9
			1024×768	625	6000.0 ms	81.9
Random Visibility Buddha		80k	1024×768	16	148.0 ms	85.0
			1024×768	625	3707.3 ms	132.5
Coherent Visibility Buddha		80k	1024×768	16	77.0 ms	163.5
			1024×768	625	261.3 ms	190.4

Table 2: *Timings for different scenes. Here $N = 128^2, M = 128$ and a two-lobe Lafortune BRDF with $k_d = 0.8$ is used as in Fig. 9. The number of visibility samples per pixel is denoted with S. For* random visibility*, shadow tests are performed in a uniform random direction. For* coherent visibility*, shadow tests are performed in the same direction for every pixel. The last column lists million Monte Carlo samples per second.*

direct Lighting. In *17th Eurographics Symposium on Rendering* (June 2006), pp. 161–172.

[LSSS04] LIU X., SLOAN P.-P., SHUM H.-Y., SNYDER J.: All-Frequency Precomputed Radiance Transfer for Glossy Objects. In *15th Eurographics Symposium on Rendering* (June 2004), pp. 337–344.

[MSW04] MEI C., SHI J., WU F.: Rendering with Spherical Radiance Transport Maps. *Comp. Graph. Forum 23*, 3 (2004), 281–290.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-Frequency Shadows Using Non-linear Wavelet Lighting Approximation. *ACM Trans. Graph. 22*, 3 (July 2003), 376–381.

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 477–487.

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast Hierarchical Importance Sampling with Blue Noise Properties. *ACM Trans. Graph 23*, 3 (2004), 488–495.

[PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann, August 2004.

[RHCB05] REN Z., HUA W., CHEN L., BAO H.: Intersection Fields for Interactive Global Illumination. *The Visual Computer 21*, 8-10 (2005), 569–578.

[RSC87] REEVES W. T., SALESIN D., COOK R. L.: Rendering Antialiased Shadows with Depth Maps. *Computer Graphics (Proceedings of ACM SIGGRAPH '87)* (1987), 283–291.

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-Time Soft Shadows in Dynamic Scenes using Spherical Harmonic Exponentiation. *ACM Trans. Graph. 25*, 3 (2006), 977–986.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *Proceedings of ACM SIGGRAPH* (July 2002), pp. 527–536.

[Slo06] SLOAN P.-P.: Normal Mapping for Precomputed Radiance Transfer. In *Symposium on Interactive 3D Graphics and Games* (2006), pp. 23–26.

[SM06] SUN W., MUKHERJEE A.: Generalized Wavelet Product Integral for Rendering Dynamic Glossy Objects. *ACM Trans. Graph. 25*, 3 (July 2006), 955–966.

[TCE05] TALBOT J., CLINE D., EGBERT P.: Importance Resampling for Global Illumination. In *16th Eurographics Symposium on Rendering* (2005), pp. 139–146.

[VG95] VEACH E., GUIBAS L. J.: Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of ACM SIGGRAPH 95* (Aug. 1995), pp. 419–428.

[WE03] WEISKOPF D., ERTL T.: Shadow Mapping Based on Dual Depth Layers. In *Eurographics 2003 Short Papers* (2003), pp. 53–60.

[Wil78] WILLIAMS L.: Casting Curved Shadows on Curved Surfaces. In *Proceedings of ACM SIGGRAPH* (August 1978), pp. 270–274.

[WTL06] WANG R., TRAN J., LUEBKE D.: All-Frequency Relighting of Glossy Objects. *ACM Trans. Graph. 25*, 2 (2006), 293–318.

[ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed Shadow Fields for Dynamic Scenes. *ACM Trans. Graph. 24*, 3 (2005), 1196–1201.