Real-Time, All-Frequency Shadows in Dynamic Scenes

Thomas Annen[1]* Zhao Dong[1] Tom Mertens[2]† Philippe Bekaert[2] Hans-Peter Seidel[1] Jan Kautz[3]‡

[1]MPI Informatik
Germany

[2]Hasselt University
tUL - IBBT, EDM, Belgium

[3]University College London
UK

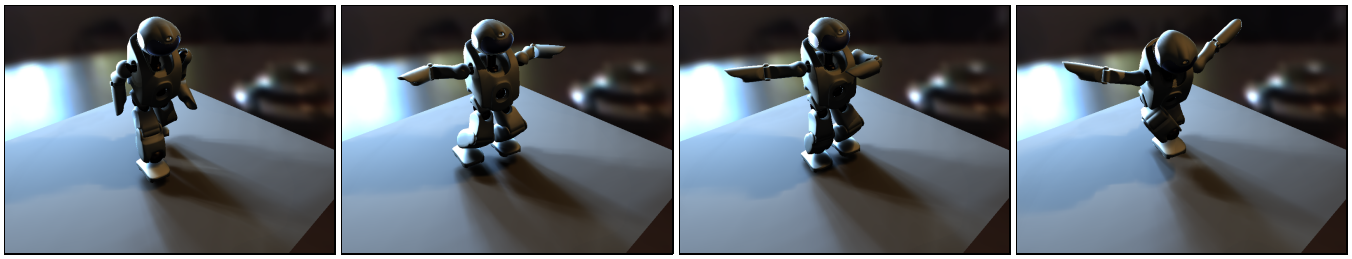


Figure 1: A fully dynamic animation of a dancing robot under environment map lighting rendered at 29.4 fps without any precomputation. Incident radiance is approximated by 30 area light sources (256×256 shadow map resolution each).

Abstract

Shadow computation in dynamic scenes under complex illumination is a challenging problem. Methods based on precomputation provide accurate, real-time solutions, but are hard to extend to dynamic scenes. Specialized approaches for soft shadows can deal with dynamic objects but are not fast enough to handle more than one light source. In this paper, we present a technique for rendering dynamic objects under arbitrary environment illumination, which does not require any precomputation. The key ingredient is a fast, approximate technique for computing soft shadows, which achieves several hundred frames per second for a single light source. This allows for approximating environment illumination with a sparse collection of area light sources and yields real-time frame rates.

Keywords: soft shadows, convolution, environment maps

1 Introduction

Real-time, photo-realistic rendering of computer-generated scenes requires a high computational effort. One of the main bottlenecks is visibility determination between light sources and receiving surfaces, especially under complex lighting such as area light sources or environment maps.

Recent methods for rendering soft shadows from area lights operate in real-time, but either tend to be too intricate and expensive for rendering multiple light sources [Guennebaud et al. 2006; Guennebaud et al. 2007; Schwarz and Stamminger 2007], or break down for detailed geometry [Assarsson and Akenine-Möller 2003]. Furthermore, these methods usually do not support environment map lighting. Other algorithms based on precomputation [Sloan et al. 2002] are good at reproducing shadows from environment maps in static scenes, but have difficulties with fully dynamic objects, despite recent progress [Ren et al. 2006].

The goal of our work is to enable real-time, all-frequency shadows in completely dynamic scenes and to support area light sources as well as environment lighting. The key contribution is a very fast method for rendering plausible soft shadows. It requires only a constant-time memory lookup, thereby enabling us to render soft shadows at hundreds of frames per second for a single area source. Environment-lit scenes can be rendered from a collection of approximating area light sources. Even though shadows are only approximate, the results are virtually indistinguishable from reference renderings, but are produced at real-time frame rates.

2 Related Work

Soft Shadows A complete review of existing shadow algorithms is beyond the scope of this article and we refer the reader to Woo et al. [Woo et al. 1990]. We restrict our discussion to soft shadow techniques. A variety of real-time soft shadowing techniques have been proposed over the past decade [Hasenfratz et al. 2003]. Most of them build on traditional techniques for rendering hard shadows. The classic shadow volume method [Crow 1977] was extended to soft shadows [Assarsson and Akenine-Möller 2003]. Unfortunately, this approach relies on silhouette information and requires frequent frame buffer access, which makes it less suitable for scenes with rich geometry. Methods based on shadow mapping [Williams 1978] scale better with scene size, since a sampling-based scene representation is employed, which is obtained by rasterizing all objects from the light source's view. Early work on shadow mapping extensions borrow ideas from image-based rendering to efficiently average hard shadows [Chen and Williams 1993; Agrawala et al. 2000]. In more recent work [Atty et al. 2006; Guennebaud et al. 2006], researchers have transferred ideas from classical discontinuity meshing [Stewart and Ghali 1994; Drettakis and Fiume 1994] to the shadow mapping domain. Such techniques compute a shadow value as the fraction of coverage of blocker geometry projected back onto the area light. To maintain high performance, the shadow map is used as a piecewise constant approximation of the blocker geometry which may yield either incorrect occluder fusion or light leaking. The work by Guennebaud et al. [2007] and bit-

*e-mail: {tannen, dong, hpseidel}@mpi-inf.mpg.de
†e-mail: {tom.mertens, philippe.bekaert}@uhasselt.be
‡e-mail: j.kautz@cs.ucl.ac.uk

mask soft shadows [Schwarz and Stamminger 2007] remove some of these problems, but increase the algorithmic complexity or computation time. Other methods make crude approximations to soft shadows in order to gain performance [Brabec and Seidel 2002; Wyman and Hansen 2003; Chan and Durand 2003]. These heuristics may produce results that deviate significantly from the actual physically-based solution. We also build on simplifications, yet the visual error is almost unnoticeable.

Convolution Soler and Sillion [1998] propose an image-based shadow algorithm based on convolution. Convolutions can be computed efficiently, even for large penumbrae. Soler and Sillion do not employ a depth buffer and therefore require an explicit notion of blockers and receivers, and cannot directly support self-shadowing. We apply a similar convolution in the context of shadow mapping, which naturally allows for self-shadowing. Variance shadow maps [Donnelly and Lauritzen 2006] and convolution shadow maps [Annen et al. 2007] support fixed convolution kernels. The result can be computed in constant time by using mipmapping or summed area tables (SAT) [Crow 1984]. However, fixed kernels cannot reproduce important visual effects such as hard contact shadows. A recent version of variance shadow maps [Lauritzen 2007] simulates penumbrae more accurately by varying the kernel size based on the average blocker depth, similar to Fernando [2005]. Unfortunately, the cost of computing this average defeats the purpose of constant cost convolution, as it requires brute-force sampling of the shadow map. An important advantage of our approach is that this step can be carried out in constant time as well.

Precomputation and Simplificiation Precomputed radiance transfer [Sloan et al. 2002] calculates and stores an illumination-invariant light transport solution off-line and uses it for real-time relighting. The scene is assumed to be static and storage demands aggressive compression, which may introduce artifacts such as blurring. Even though these limitations have been alleviated [Ng et al. 2003; Zhou et al. 2005; Sloan et al. 2005], it remains challenging to support fully dynamic scenes with arbitrary illumination. Shadow computation for dynamic scenes can be accelerated by simplifying the geometry. Ren et al. [2006] approximate dynamic objects using a sphere hierarchy, whereas Kautz et al. [2004] use a two-level mesh hierarchy. These methods only support model deformation, and assume that object topology remains static.

Environment Map Sampling Agarwal et al. [2003] proposed an efficient point sampling strategy for environment maps, in the context of ray tracing. This was later accelerated [Ostromoukhov et al. 2004]. However, all of these techniques are limited to point samples. Similar to Arbree et al. [2005] we employ an environment sampling strategy based on extended light sources. We approximate an environment with a collection of square light sources, whereas Arbree et al. use disk-shaped sources.

3 Plausible Soft Shadows Using Convolution

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

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

The advantage of computing shadows using convolution is twofold: it is compatible with image-based representations, in particular shadow mapping [Williams 1978] and thus scales well to scenes with a high polygon count. Second, convolutions can be computed efficiently using the Fourier transform [Soler and Sillion 1998], or even in constant time if the shadows have been prefiltered using mipmaps or summed area tables [Lauritzen 2007].

However, applying convolution to shadow maps in order to produce soft shadowing is not trivial. The size of the convolution kernel needs to be estimated based on the blocker distance [Soler and Sillion 1998], but when multiple blockers at different depths are involved there is no single correct blocker distance. To get a reasonable approximation of blocker depth, we compute the average depth of the blockers over the support of the filter. This approach was taken by Fernando [2005] as well as Lauritzen [2007]. Unfortunately, estimating this average is expensive since it (seemingly) requires averaging depths from the shadow map in a brute force fashion. The strength of our technique is that it allows for both efficient filtering of the shadows as well as efficient computation of the average blocker depth. Both of these operations can be expressed with the same mathematical framework, and will be described in Section 3.1.

The main visual consequence of the average blocker depth approximation is that the penumbra width may not be estimated exactly (it is correct for the parallel-planar configuration described above though). We show that this approximation does not produce offensive artifacts, and even closely approximates the ground truth solution. Figure 2 presents an overview of our soft shadow method and will be detailed in the following section.

3.1 Convolution Soft Shadows

As indicated above, soft shadows can be rendered efficiently through shadow map filtering and we therefore build on Convolution Shadow Maps (CSM) [Annen et al. 2007]. As will be shown, CSM can be extended to also compute the average blocker depth, which is required to estimate penumbra widths. We also introduce extensions that allow us to safely reduce the approximation order to further push rendering performance.

Review In order to keep the discussion self-contained, we briefly review CSM. Let $\mathbf{x} \in \mathbb{R}^3$ be the world-space position of a screen-space pixel and the point $\mathbf{p} \in \mathbb{R}^2$ represents the corresponding 2D position of a shadow map pixel. The shadow map itself encodes the function $z(\mathbf{p})$, which represents the depth of the blocker that is closest to the light source for each $\mathbf{p}$, and $d(\mathbf{x})$ is the distance from $\mathbf{x}$ to the light source.

We define the shadow function $s()$, which encodes the shadow test, as:

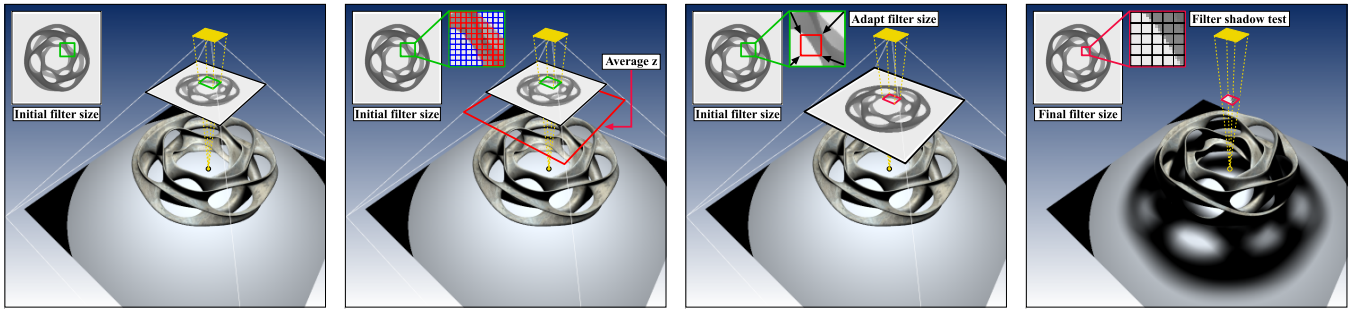
$$s(\mathbf{x}) = f(d(\mathbf{x}), z(\mathbf{p})) := \begin{cases} 1 & \text{if } d(\mathbf{x}) \leq z(\mathbf{p}) \\ 0 & \text{if } d(\mathbf{x}) > z(\mathbf{p}). \end{cases} \tag{1}$$

If the function $f()$ is expanded into a separable series:

$$f(d(\mathbf{x}), z(\mathbf{p})) = \sum_{i=1}^{\infty} a_i(d(\mathbf{x})) B_i(z(\mathbf{p})), \tag{2}$$

we can spatially convolve the result of the shadow test through prefiltering:

$$\begin{aligned} s_f(\mathbf{x}) &= \big[w * f\big(d(\mathbf{x}), z\big)\big](\mathbf{p}) \\ &\approx \sum_{i=1}^{N} a_i\big(d(\mathbf{x})\big)\big[w * B_i\big](\mathbf{p}), \end{aligned} \tag{3}$$



(a) Intersection with SM (b) Average z computation (c) Move SM to average z (d) Filter shadow test

Figure 2: An overview of our soft shadow method. First, an initial filter size is determined according to the cone defined by the intersection of the area light source, the shadow map plane, and the current receiver point (a). This filter size (green) is used to fetch the z_{avg} value from the prefiltered average z-textures. We then virtually place the shadow map plane at the z_{avg} and determine the final filter width (red) for soft shadow computation as shown in (c). In the last step, the incoming visibility value is looked up from the CSM texture (d).

where the basis images B_i are prefiltered with the kernel w, which in practice is achieved through mipmapping each $B_i(\mathbf{p})$ or computing summed area tables [Crow 1984]. At run-time, one only needs to weight the prefiltered basis images by $a_i(d(\mathbf{x}))$ and sum them up.

3.2 Estimating Average Blocker Depth

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

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

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

$$z_{avg}(\mathbf{x}) = \left[w_{avg} * z\right](\mathbf{p}). \tag{4}$$

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

$$\bar{f}(d(\mathbf{x}), z(\mathbf{p})) = \begin{cases} 1 & \text{if } d(\mathbf{x}) > z(\mathbf{p}) \\ 0 & \text{if } d(\mathbf{x}) \le z(\mathbf{p}), \end{cases} \tag{5}$$

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

$$z_{avg}(\mathbf{x}) = \frac{\left[w_{avg} * \left[\bar{f}(d(\mathbf{x}), z) \times z\right]\right](\mathbf{p})}{\left[w_{avg} * \bar{f}(d(\mathbf{x}), z)\right](\mathbf{p})}. \tag{6}$$

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

$$\bar{f}(d(\mathbf{x}), z)z = \sum_{i=1}^{N} \bar{a}_i(d(\mathbf{x}))\bar{B}_i(z(\mathbf{p}))z(\mathbf{p}). \tag{7}$$

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

$$z_{avg}(\mathbf{x}) = \frac{1}{1 - s_f(\mathbf{x})} \sum_{i=1}^{N} \bar{a}_i(d(\mathbf{x})) \left[w_{avg} * \left[\bar{B}_i(z)\, z\right]\right](\mathbf{p}). \tag{8}$$

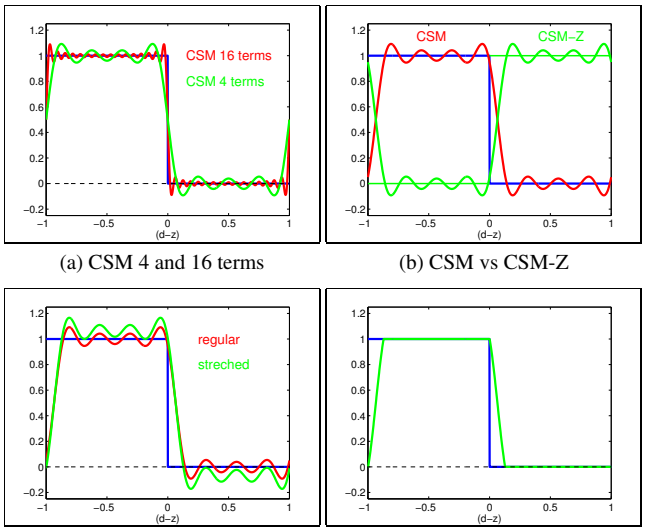
We will therefore compute new basis images $\left[\bar{B}_i(z(\mathbf{p}))z(\mathbf{p})\right]$ alongside the regular CSM basis images. We refer to this new approach for computing the average blocker depth as CSM-Z. See the appendix for a full derivation of the convolution formula for z_{avg}.

Initializing Average Depth Computation When we want to estimate or approximate the penumbra size for a given camera sample we have to do this by finding the area over the shadow map over which we will perform the z_{avg} computation. A first idea is to intersect the frustum formed by the camera sample $\mathbf{x}$ in 3D and the virtual area light source geometry with the shadow map plane (as depicted in Figure 2(a)). Unfortunately, there is no clear definition of such a plane, as the shadow map itself only represents a height field and does not have a certain plane location. We have found the near plane to work well for all our results. However, an iterative procedure is possible where one re-adjusts the location after an initial z_{avg} has been found.

3.3 CSM Order Reduction

Annen et al. [2007] propose to expand f using a Fourier series. Unfortunately, this series is prone to ringing artifacts and the shadows at contact points may appear too bright unless a high order approximation is used as shown in Figure 3(a). We propose two changes that allow us to reduce the order significantly. First, we notice that with appropriate scaling, shifting, and subsequent clamping, ringing can be avoided completely. Figure 3 illustrates this. Scaling and shifting $f(d,z)$ such that ringing only occurs above 1 and below 0 is shown in (c). Whenever the function $f(d,z)$ is reconstructed we clamp its result to $[0,1]$, avoiding any visible artifacts (d).

A second problem with a low order series is that the slope of the reconstructed shadow test is not very steep when $(d-z) \approx 0$, as can



(a) CSM 4 and 16 terms (b) CSM vs CSM-Z

(c) Shift and Scale (d) Final Clamp

Figure 3: Fourier series expansion. (a) depicts the difference between a 16- and 4-term reconstruction. (b) CSM and CSM-Z are exactly opposite to each other. Ringing suppression is possible with appropriate scaling and shifting (c), followed by clamping the function to $[0,1]$ (d).

be seen in Figure 3(d), and yields shadows that are too bright near contact points. A simple solution is to apply a non-linear transformation $G(v) = v^p$ to the filtered shadow value $s_f(\mathbf{x})$ with $p \geq 1$. This tends to darken the shadows and thus hides light leaking. If $p = 1$, nothing changes. On the downside, darkening also removes smooth transitions from penumbra regions, so we want to only apply it where necessary. When $d(\mathbf{x}) - z_{avg}(\mathbf{p})$ is small, we know that $\mathbf{x}$ is near a contact point where leaking will likely occur. Fortunately, this is also where penumbrae should be hard anyway. We therefore compute an adaptive exponent p based on this difference:

$$p = 1 + A\, exp\big(-B\,\big(d(\mathbf{x}) - z_{avg}(\mathbf{p})\big)\big). \tag{9}$$

A controls the strength of the darkening, and B determines the maximal distance of z_{avg} from the receiver point for which darkening is applied to. Figure 4 shows this effect for a varying parameter B.

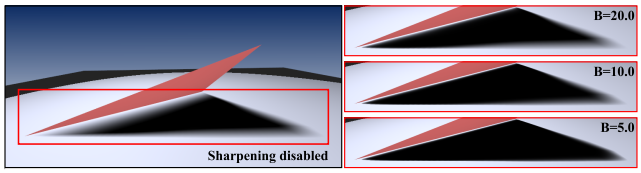


Figure 4: An illustration of the impact of sharpening parameters A and B. A is fixed to 30.0, whereas B is set to 5.0, 10.0, and 20.0 showing how B changes the spatial extend of the sharpening.

4 Illumination with Soft Shadows

4.1 Rendering Prefiltered Soft Shadows

Generating soft shadows with our new algorithm is similar to rendering anti-aliased shadows [Annen et al. 2007]. First, the scene is rasterized from the center of the area light source and the z-values are written to the shadow map. Based on the current depth map two sets of images are produced: the Fourier series basis and its complementary basis images multiplied by the shadow map z-values.

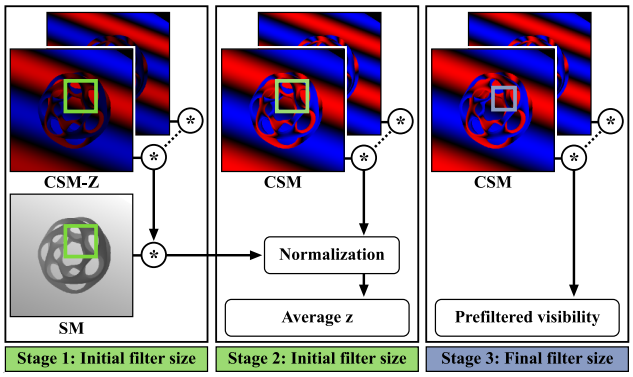


Figure 5: Convolution soft shadows pipeline. Stage 1 reconstructs a prefiltered z_{avg}. The z_{avg} is passed to the 2nd stage for normalization. Thereafter, the final filter size is computed as described in 2(c), and the visibility is evaluated by a regular CSM reconstruction.

After we have generated both data structures, we can run the prefilter process. Note that when the convolution formula from Eq. 8 is evaluated using a Fourier series, it also requires prefiltering the shadow map due to the constant factor when multiplying $\bar{f}()$ by $z(\mathbf{p})$ (see appendix). In our implementation, we support image pyramids (mipmaps) and summed-area-tables. Other linear filtering operations are applicable as well. When filtering is complete, we start shading the scene from the camera view and employ convolution soft shadows for high-performance visibility queries. An overview of the different steps is given in Figure 5.

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

All three stages together require only six RGBA and one depth texture access (for a reconstruction order $M = 4$).

4.2 Generation of Area Lights for Environment Maps

We propose the following greedy algorithm for decomposing an environment map into a number of area light sources. We assume the environment map to be given as a cube map and proceed by decomposing each cube map face separately.

The process works directly in the 2D domain of the cube map face. We first find the pixel with the largest amount of energy and create a preliminary 1×1 area light for it. We then iterate over all four sides of the area light and try to enlarge each side by one pixel. The area light is enlarged if the ratio between the amount of energy E_{delta} that would be added to the light by enlarging it and the amount of energy E_{total} that the enlarged area light would emit is larger than a given threshold t. We repeat this enlargement process until none of the four sides can be enlarged, or the area light covers the complete cube map face. After the enlargement process has stopped,

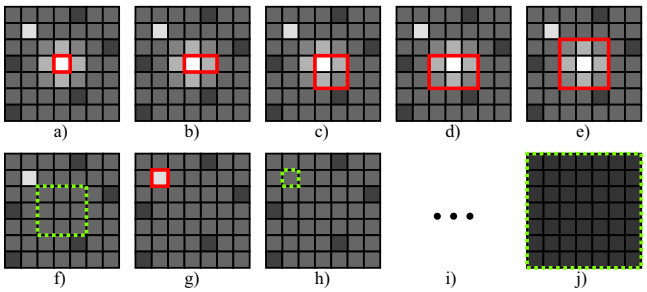


Figure 6: Fitting area lights to a cube map face. We first fit a 1×1 area light to the brightest pixel (a). In turn, we try to enlarge the area light at each side until a stopping criteria is reached (b)-(e). We remove the energy for this area light (but leave some amount to blend it with the area around it) (f), and continue with fitting more area lights (g)-(i), until we have area lights covering the whole face.

we remove the energy of this portion of the cube map face but leave a residual amount of energy to enable better fits in later iterations and create the final area light for it. The residual amount equals the average amount of energy adjusted to the size of the area. We then continue with fitting more area lights until we have covered the whole cube map face. Figure 6 illustrates the process. Note that our method may produce overlapping area lights. The parameter t determines the total number of light sources fitted to each cube map face. Examples are shown in the results section.

5 Limitations and Discussion

Failure Cases Our technique shares the same failure cases as PCF-based soft shadowing [Fernando 2005]. We assume that all blockers have the same depth within the convolution kernel (essentially flattening blockers), similar to Soler and Sillion's method [1998]. This assumption is more likely to be violated for larger area lights. Nevertheless, shadows look qualitatively similar to the reference rendering, as shown in see Figure 7. The use of a single shadow map results in incorrect shadows for certain geometries. This problem is commonly referred to as "single silhouette artifacts", which we share with many other techniques [Assarsson and Akenine-Möller 2003; Guennebaud et al. 2006].

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

Ringing Suppression Our proposed ringing suppression using scaling and shifting followed by clamping does indeed reduce ringing and improves shadow darkness near contact points, but also sharpens shadows slightly as can be seen in Figure 9. However, this process is necessary to keep frame rates high as it allows the use of fewer terms in the expansion and the differences are barely noticeable. See the comparisons in the results section, all of which are rendered using ringing suppression.

Mipmaps vs. Summed Area Tables The quality that our method can achieve depends on the prefiltering process. Mipmaps are computationally inexpensive, but their quality is inferior compared to SATs as they re-introduce aliasing again at higher mipmap levels. However, SATs require more storage due to the need to use floating point textures [Hensley et al. 2005] especially when using many area lights. In the case of multiple area lights, as used for environment mapping, artifacts are masked and mipmapping is a viable option. Figure 8 compares both solutions.

Textured Light Sources Our method cannot handle textured light sources directly as the prefiltering step cannot be extended to include textures. Instead, we decompose complex luminaires such as environment maps into uniform area lights.

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

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

6 Results

In this section we report on the quality and performance of our method. Our technique was implemented in DirectX 10 and all results were rendered on a Dual-Core AMD Opteron with 2.2GHz using an NVIDIA GeForce 8800 GTX graphics card. Our performance timings are listed in Table 1.

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

Table 1: Frame rates for the Buddha scene with 70k faces from Figure 10, rendered using reconstruction order $M = 4$. For many lights and high resolution shadow maps, our method may require more than the available texture memory (reported as missing entries).

The first result shown in Figure 7 compares the shadow quality of several different algorithms to a reference rendering. We analyze two situations in particular, large penumbrae and close-contact shadows (see close-ups). Shadows rendered with our new technique are very close to the reference, bitmask soft shadows perform slightly better at contact shadows and backprojection methods tend to overdarken shadows when the depth complexity increases. Percentage closer soft shadows produce banding artifacts in larger penumbra regions due to an insufficient number of samples.

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

We begin with a side-by-side comparison between mipmap- and SAT-based soft shadows in Figure 8. Mipmaps produce less accurate results compared to summed-area-tables for rendering single lights due to aliasing artifacts. For complex lighting environments, however, shadows from many light sources are averaged, which makes mipmapping artifacts less noticeable (Fig. 10 and 11).

(a) Ground Truth (Ray-Traced) (b) Our Method – SAT 4 Terms (60 fps) (c) Our Method – SAT 16 Terms (23 fps)

(d) Percentage Closer Soft Shadows (18 fps) (e) Backprojection (41 fps) (f) Bitmask Soft Shadows (19 fps)

Figure 7: Shadow quality comparison of several methods (SM was set to 512×512, scene consists of 212K faces): ray-tracing (a), our method using SATs – 4 terms (b) and 16 terms (c), percentage closer soft shadows [Fernando 2005] (d), backprojection [Guennebaud et al. 2006] (e), and bitmask soft shadows [Schwarz and Stamminger 2007] (f).

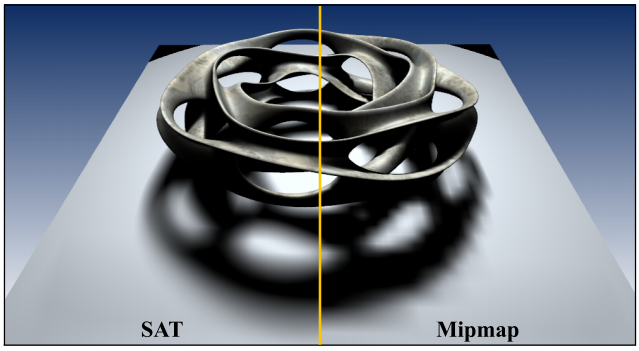


Figure 8: The difference in filter quality when using a summed-area-table (left) and a mipmap (right). Successive down sampling with a 2×2 box-filter introduces aliasing at higher mipmap levels.

Figure 9 illustrates the influence of the reconstruction order and sharpening. We render a foot bone model of high depth complexity and demonstrate the effect of the sharpening function $G()$. While contact shadows (toe close-up) are darkened and slightly sharper than the results rendered with $M = 16$, their larger penumbra areas are not influenced, which maintains the overall soft shadow quality.

Figure 10 shows the influence of the number of light sources used for approximating the environment map. Below the renderings we show the fitted area light sources and a difference plot. Rendering with 30 lights (Fig. 10(d)) already looks quite similar to the reference but some differences are noticeable. With 45 area lights, the differences to the reference are significantly reduced and the result is visually almost indistinguishable. This example illustrates that mipmapping produces adequate results, while offering a more than threefold speedup compared to summed-area tables (see Figure 11). The reference images in Figure 10 and 11 have been generated with 1000 environment map samples [Ostromoukhov et al. 2004] using ray tracing. Figure 11 also compares brute force GPU-based shadow rendering with 500 samples, which achieves a much slower frame rate compared to our method.

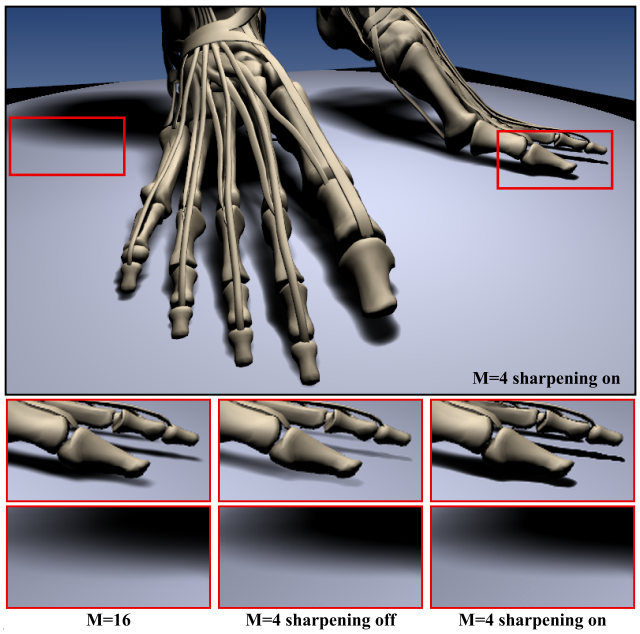
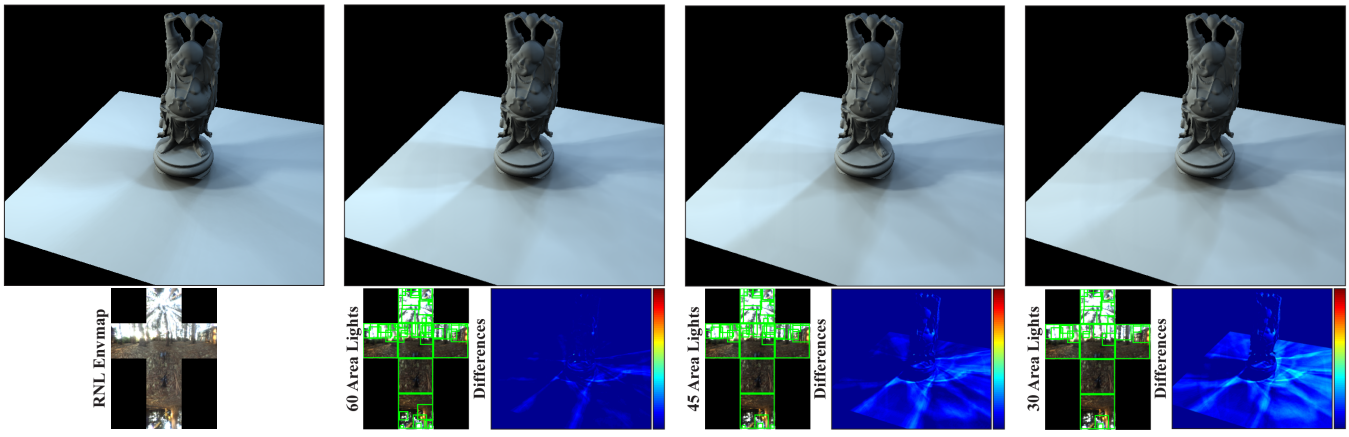


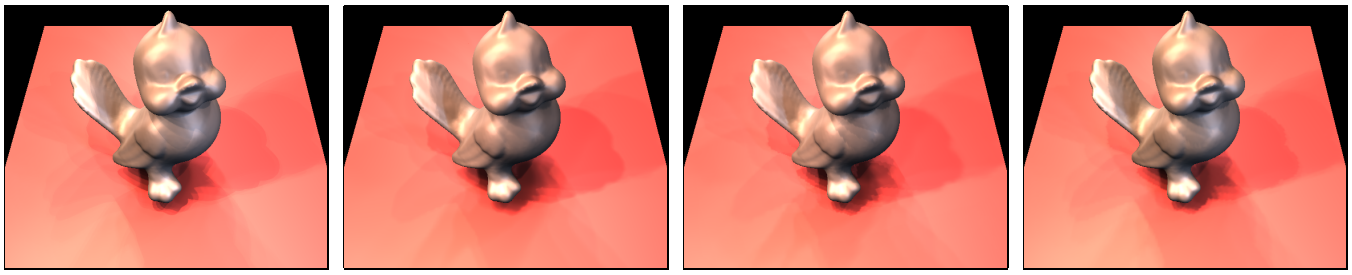
Figure 9: Influence of reconstruction order M and sharpening. The close-ups show that shadow darkening is restricted to contact points whereas larger penumbra areas remain unaffected and smooth.

Figure 12 shows that our method can easily deal with complex geometry while delivering high quality renderings. The closeups



(a) RT SMs (b) 60 ALs, $t = 0.035$ (9.8 fps) (c) 45 ALs, $t = 0.025$ (14.1 fps) (d) 30 ALs, $t = 0.015$ (18.4 fps)

Figure 10: Comparison between ray-tracing 1000 point lights (a), our technique with mipmaps using 60 (b), 45 (c), and 30 (d) area light sources. Each image shows the environment map with the the fitted light sources in green. SM resolution was set to 256×256.



(a) Ray-Tracing (b) SAT 30 ALs, 7.5 fps (c) Mipmap 30 ALs, 25.4 fps (d) GPU SMs, 1.4 fps

Figure 11: In this figure we compare our rendering results with 30 ALs (St.Peters Basilica EM) against ray-tracing 1000 point lights and standard GPU-based shadow mapping. (a) ray-tracing, (b) our technique with SATs (c), our technique with mipmaps, and (d) GPU-based shadow mapping which achieves similar quality (500 shadow maps). SM resolution was set to 256×256.

Figure 12: A very complex model illuminated by 1 AL with varying light source sizes from left (small) to right (large).

show how shadows soften as the area light size is increased. To capture fine geometric details we used a $1K \times 1K$ shadow map.

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

7 Conclusions and Future Work

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

As future work, we intend to explore the use of area lights for indirect illumination, which could be an important step toward interactive global illumination for fully dynamic scenes.

Acknowledgements

We thank Kun Xu and the Graphics and Geometric Computing Group at Tsinghua University for providing the robot 3D animation sequence, Michael Schwarz for the bitmask- and backprojection soft shadow demo, and Thomas Luft for providing the leaf model from Figure 12. Thanks to Paul Debevec for his light probes (http://www.debevec.org) and to the Stanford University Computer Graphics Laboratory for the happy Buddha model. The model from Figure 8 was generated by 'Sculpture Generator 1' by Carlo H. Séquin, UC Berkeley. Part of the research at EDM is funded by the European Regional Development Fund and the Flemish government, and part of the research at UCL is funded by EPSRC (grant EP/E047343/1).

Appendix

Our z_{avg} computation uses the Fourier series [Annen et al. 2007] to approximate $\bar{f}()$ and yields the following:

$$\bar{f}(d(\mathbf{x}),z(\mathbf{p})) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \frac{1}{c_k} \sin\left[c_k(d(\mathbf{x}) - z(\mathbf{p}))\right], \tag{10}$$

with $c_k = \pi(2k-1)$. Then the convolution from Eq. 8 becomes:

$$\begin{aligned} z_{avg}(\mathbf{x}) &\approx \frac{1}{1-s_f(\mathbf{x})}\left[w_{avg} * \left(\frac{1}{2} + \sum_{k=1}^{M} \frac{2}{c_k} \sin\left[c_k(d(\mathbf{x}) - z)\right]\right) z\right](\mathbf{p}) \\ &\approx \frac{1}{1-s_f(\mathbf{x})}\left[w_{avg} * \frac{z}{2} + \frac{2}{c_k}\sum_{k=1}^{M} \sin\left(c_k d(\mathbf{x})\right)\left(w_{avg} * z\cos(c_k z)\right) - \right. \\ &\qquad \left. \frac{2}{c_k}\sum_{k=1}^{M} \cos\left(c_k d(\mathbf{x})\right)\left(w_{avg} * z\sin(c_k z)\right)\right](\mathbf{p}). \end{aligned} \tag{11}$$

This means there is an additional basis image containing $z/2$ values (basically corresponding to a shadow map, see Figure 5), which needs to be filtered.

References

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

AGRAWALA, M., RAMAMOORTHI, R., HEIRICH, A., AND MOLL, L. 2000. Efficient Image-Based Methods for Rendering Soft Shadows. In *Proc. of SIGGRAPH 2000*, 375–384.

ANNEN, T., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2007. Convolution Shadow Maps. In *Rendering Techniques 2007 (Proc. of EGSR)*, 51–60.

ARBREE, A., WALTER, B., AND BALA, K. 2005. Pre-Processing Environment Maps for Dynamic Hardware Shadows. Tech. rep., Dept. of Computer Science, Cornell University.

ASSARSSON, U., AND AKENINE-MÖLLER, T. 2003. A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware. *ACM Trans. Graph. 22*, 3, 511–520.

ATTY, L., HOLZSCHUCH, N., LAPIERRE, M., HASENFRATZ, J.-M., HANSEN, C., AND SILLION, F. 2006. Soft Shadow Maps: Efficient Sampling of Light Source Visibility. *Computer Graphics Forum 25*, 4.

BRABEC, S., AND SEIDEL, H.-P. 2002. Single Sample Soft Shadows Using Depth Maps. In *Graphics Interface 2002*, 219–228.

CHAN, E., AND DURAND, F. 2003. Rendering Fake Soft Shadows with Smoothies. In *Rendering Techniques 2003 (Proc. of EGSR)*, 208–218.

CHEN, S., AND WILLIAMS, L. 1993. View Interpolation for Image Synthesis. In *Proc. of SIGGRAPH '93*, 279–288.

CROW, F. 1977. Shadow Algorithms for Computer Graphics. *Computer Graphics (Proc. of SIGGRAPH '77)* (July), 242–248.

CROW, F. C. 1984. Summed-Area Tables for Texture Mapping. *Computer Graphics (Proc. of SIGGRAPH '84)*, 207–212.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance Shadow Maps. In *Proc. of SI3D '06*, 161–165.

DRETTAKIS, G., AND FIUME, E. 1994. A Fast Shadow Algorithm for Area Light Sources Using Backprojection. In *SIGGRAPH '94*, 223–230.

FERNANDO, R. 2005. Percentage-Closer Soft Shadows. In *ACM SIGGRAPH 2005 Sketches*, 35.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2006. Real-time Soft Shadow Mapping by Backprojection. In *Rendering Techinques 2006 (Proc. of EGSR)*, 227–234.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2007. High-Quality Adaptive Soft Shadow Mapping. *Computer Graphics Forum (Proc. of Eurographics 2007) 26*, 3 (Sept.).

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

HENSLEY, J., SCHEUERMANN, T., SINGH, M., AND LASTRA, A. 2005. Interactive Summed-Area Table Generation for Glossy Environmental Reflections. In *ACM SIGGRAPH 2005 Sketches*, 34.

KAUTZ, J., LEHTINEN, J., AND AILA, T. 2004. Hemispherical Rasterization for Self-Shadowing of Dynamic Objects. In *Rendering Techniques 2004 (Proc. of EGSR)*, 179–184.

LAURITZEN, A. 2007. Summed-Area Variance Shadow Maps. In *GPU Gems 3*, H. Nguyen, Ed.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-Frequency Shadows Using Non-linear Wavelet Lighting Approximation. *ACM Trans. Graph. 22*, 3, 376–381.

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

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

SCHWARZ, M., AND STAMMINGER, M. 2007. Bitmask Soft Shadows. *Computer Graphics Forum (Proc. of Eurographics 2007) 26*, 3 (Sept.), 515–524.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph. 21*, 3, 527–536.

SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, Deformable Precomputed Radiance Transfer. *ACM Trans. Graph. 24*, 3, 1216–1224.

SOLER, C., AND SILLION, F. 1998. Fast Calculation of Soft Shadow Textures Using Convolution. In *Proc. of SIGGRAPH '98*, 321–332.

STEWART, A. J., AND GHALI, S. 1994. Fast Computation of Shadow Boundaries Using Spatial Coherence and Backprojections. In *Proc. of SIGGRAPH '94*, 231–238.

WILLIAMS, L. 1978. Casting Curved Shadows on Curved Surfaces. *Computer Graphics (Proc. of SIGGRAPH '78)* (August), 270–274.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A Survey of Shadow Algorithms. *IEEE Computer Graphics & Applications 10*, 6, 13–32.

WYMAN, C., AND HANSEN, C. 2003. Penumbra Maps: Approximate Soft Shadows in Real-Time. In *Rendering Techniques 2003 (Proc. of EGSR)*, 202–207.

ZHOU, K., HU, Y., LIN, S., GUO, B., AND SHUM, H.-Y. 2005. Precomputed Shadow Fields for Dynamic Scenes. *ACM Trans. Graph. 24*, 3, 1196–1201.