# Real-Time Image-Based Lighting of Dynamic Scenes with Moment Soft Shadow Mapping

Tom Kneiphof*
*Supervised by: Stefan Krumpen*†

Institute of Computer Science II - Computer Graphics
University of Bonn
Bonn / Germany

## Abstract

Casting shadows in dynamic scenes under complex illumination is a challenging problem, since no precomputation can be applied. The illumination is provided in form of a light probe, which potentially changes every frame. We propose a method for generating area lights from the light probe, built upon a median cut algorithm for light probe sampling. The shadows from the area lights are efficiently computed by moment soft shadow mapping, which has a lower memory footprint than other techniques. In this process very large area lights are utilized, pushing the limits of moment soft shadow mapping. We demonstrate the use of 6 instead of 4 moments in moment soft shadow mapping, which drastically increases the quality of the shadows for large area lights. We also present some other modifications, in order to improve the resulting shadows. We implemented the algorithm for generating area lights in compute shaders and demonstrate the sampling of animated light probes on the GPU and lighting of a dynamic scene in real time.

**Keywords:** real time, moment shadow mapping, soft shadows, area lights, light probes, environment maps

## 1 Introduction

In real-time applications one usually deals with direct lighting only. The computation of indirect illumination, where light is reflected multiple times in a scene before being observed, is expensive, especially for dynamic scenes, and usually precomputed.

We illuminate a small dynamic scene, where the incident light is affected by a larger scene. In this scenario we want to incorporate single bounce indirect illumination, where the light is reflected once in the larger scene before entering the smaller scene. A light probe is used as an intermediate representation for the incident illumination, which maps each direction to the corresponding radiance entering the smaller scene. Since we do not have depth information of light probe at hand, the sources of the light are assumed to be infinitely far away. In this paper the light probe is given to us, and we do not deal with the creation of light probes.

In the smaller scene, we want to compute the shadows caused by the incident illumination, since shadows are an important visual cue in rendering. They provide valuable information about the spatial relation between objects and contribute to the overall realism. Shadow mapping [13] is the state of the art technique for casting shadows from point lights. Various extensions for shadow mapping exist, which improve the quality of the produced shadows. Other extensions [2, 5, 9] approximate the shadows casted by area lights based on shadow mapping.

In this paper, we approximate the light probe by area lights. Our algorithm for generating the area lights extends the median cut algorithm for light probe sampling [4], but can work with any algorithm that cuts a light probe into regions. We implement our technique for generating the area lights on the GPU, with the aim of generating the area lights every frame, allowing for animated light probes.

The shadows casted from the generated area lights are then computed by moment soft shadow mapping [9]. We compare the performance and visual quality of moment soft shadow mapping [9] and convolution soft shadow mapping [2], as well as percentage closer soft shadows [5].

In Section 2 we lay out other techniques related to the problem approached by this paper. Section 3 lays out the techniques that this paper is built on and Section 4 addresses our changes and extensions and Section 5 describes implementation details. Results are presented in Section 6 before we conclude in Section 7.

## 2 Related Work

Path tracing [6] is able to illuminate a scene by a light probe, by shooting rays into the scene and performing Monte Carlo integration. But it is very costly and is not suitable for real-time applications. However, it provides a ground-truth solution, thus we can use path tracing to evaluate the visual quality of our approach.

*kneiphof@cs.uni-bonn.de
†krumpen@cs.uni-bonn.de

Precomputed radiance transfer [11] uses spherical harmonics to approximate a given light probe by a linear combination of a few basis light probes. The full global illumination of a static scene is precomputed for each basis light probe. The precomputed global illumination is then used to approximate the illumination under the given light probe. An advantage of this technique is, that it also takes interreflections inside of the illuminated scene into account. The major drawback of this technique is, that it uses a static scene. Another disadvantage is, that it handles only low frequency lighting. High frequency light sources, like the sun, are not handled well by this technique.

In this paper, we pursue an alternative approach, which generates light sources from a light probe and uses them to illuminate the scene. Several algorithms have been proposed [4, 12], which partition a light probe into regions and define a point light at the centroid of each region. Once the point lights are created, the illumination of the scene can be computed by applying shadow mapping [13].

In the context of convolution soft shadow mapping [2] another light probe sampling algorithm was presented. It picks the brightest pixel in the light probe, fits an area light around it, removes the corresponding radiance from the light probe, and repeats until the desired number of lights has been created or the light probe is empty.

Soft shadow mapping techniques are based on percentage closer soft shadows (PCSS) [5]. The main idea is to compute the average blocker depth in front of a fragment to be lit under the assumption that all shadow casting geometry shares the same depth in the shadow map frustum. The final filter region for the visibility test is then chosen based on the average blocker depth.

Convolution soft shadow mapping (CSSM) [2] is an extension of convolution shadow mapping (CSM) [1], which allows for prefiltering shadow maps. By storing additional terms in the shadow map, the average blocker depth can be prefiltered as well. CSM uses the Fourier transform to derive a linearized form of the visibility test, where terms depending on the fragment depth and terms depending on the shadow map value are separated. This way, the shadow map can be filtered independently of the receivers fragment depth. In CSSM the same is done for the average blocker depth as well, which requires additional information to be stored in the shadow map.

Moment shadow mapping (MSM) [10] can be easily extended to moment soft shadow mapping (MSSM) [9] without storing any additional terms in the shadow map. This leads to a very low memory footprint compared to CSSM. For more details see Section 3.2.1.

# 3 Background

In this section we describe the existing techniques that are crucial for this paper.

**Light Probes** A light probe specifies for each direction the incoming radiance in the red, green and blue band. The light probe is internally represented as a cubemap, which is essentially an array of six 2D textures, mapped onto the unit cube. GPUs have methods built in for sampling cubemaps by a directional vector. However, our algorithm will work on the 2D cubemap faces directly.

**Summed Area Tables** A lot of sums have to be computed during shadow mapping and the creation of the area lights from the light probe. Only considering sums over rectangular regions, summed area tables (SATs) can be used to accelerate the computation. A summed area table $S$ on a 2D texture $T$ stores at each texel $(u,v)$ the sum:

$$S(u,v) = \sum_{x=0}^{u} \sum_{y=0}^{v} T(x,y) \tag{1}$$

The sum over a rectangular region $[x_1,x_2] \times [y_1,y_2]$ can then be computed in constant time as follows:

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} T(x,y) = S(x_2,y_2) + S(x_1-1,y_1-1) \tag{2}$$

$$- S(x_2,y_1-1) - S(x_1-1,y_2) \tag{3}$$

## 3.1 Light Probe Segmentation

In order to generate area lights from a light probe, we partition the light probe into a number of regions. For each region, an area light is then created, as described in Section 4.1. A median cut algorithm for light probe sampling [4] is employed for the partitioning step.

Initially for each cubemap face, one rectangular region is created, which spans the whole face. While the number of regions is lower than the number of desired light sources, each region is subdivided along its longer edge. The subdivision of a region is done in such a way, that the illuminance of the original region is split evenly across its two subregions. The illuminance $M_v(R)$ of a region $R$ is the integral over the contained luminance $L_v(R)$. The luminance is obtained from the radiance in red, green and blue bands in the light probe by weighting them with the response function of the human eye. In order to compute the illuminance of a region in constant time during the partitioning, a summed area table is computed beforehand.

## 3.2 Soft Shadow Mapping

Our goal is to compute the shadows casted from area lights, which are generated from a light probe. Shadow mapping [13] assumes the light sources to be point lights. However, this technique can be extended to approximate soft shadows, casted from area lights. In order to achieve this, all blocking geometry between the light source and the receiver is assumed to share the same distance to the receiver [5].

Consider an area light of size $sz_L$ and a receiver $\mathbf{p}$, for which the visibility from the area light shall be computed. The receiver $\mathbf{p}$ has depth $d(\mathbf{p})$ in the shadow map frustum and samples the shadow map at location $\mathbf{x_p}$, giving a shadow map depth of $z(\mathbf{x_p})$. In order to compute the visibility of $\mathbf{p}$ from the area light, a filtered visibility test $s_f(\mathbf{p})$ is evaluated (see Eq. 6). The size of the visibility test is chosen depending on the average blocker depth $z_{avg}(\mathbf{p})$.

At first, we compute the average blocker depth $z_{avg}(\mathbf{p})$ in some search region in the shadow map (see Eq. 4). The size of the search region usually depends on the size of the area light $sz_L$.

$$z_{avg}(\mathbf{p}) := \frac{[w * z \cdot (1 - f(d(\mathbf{p}), z))](\mathbf{x_p})}{[w * (1 - f(d(\mathbf{p}), z))](\mathbf{x_p})} \tag{4}$$

Here, $w$ is a linear filter, corresponding to the search region. $f(d,z)$ is a binary function, comparing fragment depth $d$ with some shadow map $z$, which evaluates to 1 if $d < z$ and 0 otherwise.

Assuming that all blocking geometry share the same depth $z_{avg}(\mathbf{p})$, a filter size $sz_f$ for the visibility test is chosen, such that it covers all shadow map samples which could block the receiver, given that they are located in depth $z_{avg}(\mathbf{p})$ (see Figure 1). Let $sz_L$ be the size of the area light. Then the size $sz_f$ of the filter region for the visibility test is obtained as follows:

$$sz_f = \frac{sz_L}{d(\mathbf{p})} \cdot (d(\mathbf{p}) - z_{avg}(\mathbf{p})) \tag{5}$$

This results in the final filtered visibility test $s_f(\mathbf{p})$, where $w$ is a linear box filter with size $sz_f$:

$$s_f(\mathbf{p}) := [w * f(d(\mathbf{p}), z)](\mathbf{x_p}) \tag{6}$$

The computation of the visibility for an area light requires three filtering operations per receiver, as defined in Eq. 4 and Eq. 6. Percentage closer soft shadows [5] computes these explicitly in the fragment shader, which is very expensive for large filter regions. Therefore, it is highly desirable to use filterable shadow maps to accelerate the filtering operations. Summed area tables are well suited for applying box filters of variable sizes.

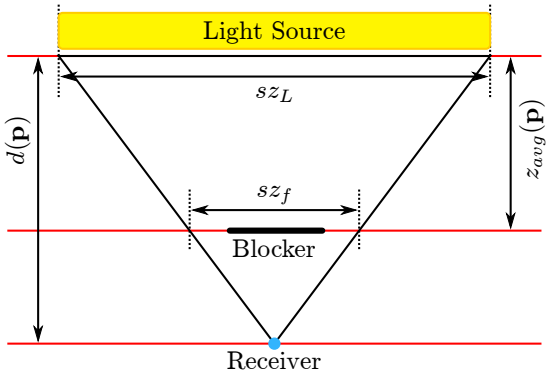



Figure 1: Illustration of the relation between fragment depth $d(\mathbf{p})$, average blocker depth $z_{avg}(\mathbf{p})$, area light size $sz_L$ and filter size $sz_f$.

#### 3.2.1 Moment Soft Shadow Mapping

Moment shadow mapping (MSM) [10] is a filterable shadow mapping technique. It stores $m$ moments $z^1, \ldots, z^m$ in a moment shadow map. A lower and upper bound for the filtered visibility test is then estimated, based on the filtered moments. In practice, the upper bound for the visibility test is used, implying that lit surfaces are never shadowed. However, light leaking can occur, in which shadowed surfaces are illuminated.

MSM can be extended to compute the average blocker depth for a given filter region, resulting in moment soft shadow mapping (MSSM) [9]. In the moment shadow mapping algorithm, a discrete depth distribution $z_1, \ldots, z_n$ with weights $w_1, \ldots, w_n$ is constructed. These depth values can be used to estimate the result of $z_{avg}(\mathbf{p})$ as well. No additional information has to be stored in the shadow map, in order to compute the average blocker depth.

$$z_{avg}(\mathbf{p}) = \frac{\varepsilon \cdot d(\mathbf{p}) + \sum_{i=1, z_i < d(\mathbf{p})}^{n} w_i \cdot z_i}{\varepsilon + \sum_{i=1, z_i < d(\mathbf{p})}^{n} w_i} \tag{7}$$

In order to compute $z_{avg}(\mathbf{p})$ robustly, a constant $\varepsilon$ is introduced, that moves the average blocker depth slightly towards the receiver and ensures that the nominator never approaches zero.

# 4 Methodology

## 4.1 Area Light Generation

Our algorithm for extracting area lights from a light probe builds on a median cut algorithm for light probe sampling [4] as described in Section 3.1. Here, the final partition of the light probe is given, and we describe how area lights are derived from regions on a cubemap face.

#### 4.1.1 Cubemap Weighting

Generating area lights from the light probe requires integration on the unit sphere, where a point on the sphere represents a direction. However, the integration domain is specified in terms of a cubemap face, since the light probe is given as a cubemap. Unfortunately, cubemaps do not sample the unit sphere regularly, meaning that some regions are oversampled and some are undersampled. It is important to account for this fact in order compute the integrals correctly and to conserve the light probe's energy.

Using the transformation formula "from calculus", we can derive a weighting factor $w(u,v)$ depending on normalized cubemap face coordinates $u,v \in [-1,1]$, which accounts for the irregular sampling:

$$w(u,v) := (u^2 + v^2 + 1)^{3/2} \tag{8}$$

When approximating an integral by a finite sum of samples on the cubemap face, each sample is weighted by $w(u,v)$. Integrating 1 over the domain of $[-1,1]^2$ gives 4, so we want the sum over all samples in a cubemap face to be 4. The number of samples in a cubemap face of size $m \times m$ is given by $m^2$. So another factor of $\frac{4}{m^2}$ is applied.

#### 4.1.2 Area Light Dimensions

Consider a region $R$ of the final light probe partition. The direction of the area light corresponds to the centroid $\mu(R)$ within the region $R$ on the cubemap, where the samples $x$ in the region are weighted by their luminance value $L_v(x)$. The size of the area light is estimated based on the spatial variance along the x and y axis of the cubemap face containing $R$, again weighted by luminance.

$$\mu(R) = \frac{\int_R w(x) L_v(x) x \, dx}{\int_R w(x) L_v(x) \, dx} \tag{9}$$

$$\Sigma(R) = \frac{\int_R w(x) L_v(x) (x-\mu)(x-\mu)^t \, dx}{\int_R w(x) L_v(x) \, dx} \tag{10}$$

If the x and y axes are correlated, the covariance matrix can be diagonalized in order to align the area light with the principal directions of spatial variance. However, we find that this does not improve the quality of the generated area lights. Considering the variance along the x and y axes independently leads to sufficiently good results.

In the following, we consider the 1-dimensional case: The area light corresponds to a uniform distribution $\mathcal{U}(\mu - d, \mu + d)$, where $\mu$ is the centroid and $d$ is the size of the area light. Such a distribution has a variance of $\sigma^2 = \frac{1}{3}d$. By measuring the variance $\sigma^2$ from the light probe (see Eq. 10), we can compute the extent of a uniform area light.

Once the size of the area light along the x and y axes of the cubemap face is determined, the shadow map frustum is derived by defining the basis vectors of the shadow map frustum in the world coordinate system. The z axis of the frustum is chosen as the direction of the area light. For the other two basis vectors, the horizontal and vertical basis vectors of the cubemap face are transformed into the world coordinate system, and projected into the orthogonal complement of the z axis. The size along the x and y axes is scaled by the length of the corresponding projected vector. All three vectors are then normalized.

#### 4.1.3 Area Light Illumination

The irradiance of the area lights can be computed from the light probe using summed area tables (see Eq. 11). However, using the irradiance $M_e(R)$ directly for illumination leads to wrong results, which do not conserve energy. Since we cannot afford to perform the integration (see Eq. 13) according to the rendering equation [6] explicitly, we want to find a good closed from expression. In our case, we only consider diffuse surfaces.

$$M_e(R) = \int_R w(x) L_e(x) \, dx \tag{11}$$

In the following, we consider circular directional area lights, which have an opening angle $\alpha_A$, mean direction $\omega_A$ and emit uniform radiance $L_e^A$. The solid angle covered by the area lights is denoted as $\Omega_A$. For such area lights, the integral in the rendering equation can be expressed in closed form for diffuse surfaces, neglecting any occlusion.

Let $\rho(\mathbf{p}) = \rho(\mathbf{p}, \omega_i, \omega_o)$ be the diffuse BRDF model of a point $\mathbf{p}$. The surface normal of $\mathbf{p}$ is $\vec{n}_\mathbf{p}$. Then $\mathbf{p}$ emits radiance $L_e^o(\mathbf{p}, \omega_o)$ along direction $\omega_o$.

$$L_e^o(\mathbf{p}, \omega_o) = \int_{\Omega_A} \rho(\mathbf{p}, \omega_o, \omega_i) L_e^A(\mathbf{p}, \omega_i) \langle \omega_i, \vec{n}_\mathbf{p} \rangle \, d\omega_i \tag{12}$$

$$= \rho(\mathbf{p}) L_e^A \int_{\Omega_A} \langle \omega_i, \vec{n}_\mathbf{p} \rangle \, d\omega_i \tag{13}$$

The integral over $\Omega_A$ can be transformed to polar coordinates in order to solve it.

$$\int_{\Omega_A} \langle \omega_i, \vec{n}_\mathbf{p} \rangle \, d\omega_i = \int_0^{2\pi} \int_0^{\alpha_A} \sin(\theta) \langle \omega_i, \vec{n}_\mathbf{p} \rangle \, d\theta d\phi \tag{14}$$

$$= \pi(1 - \cos^2(\alpha_A)) \langle \omega_A, \vec{n}_\mathbf{p} \rangle \tag{15}$$

We are given a certain region on the light probe, on which we can easily compute the irradiance $M_e^A = M_e(R)$. The radiance $L_e^A$ can be expressed depending on $M_e^A$, since $L_e^A$ is constant across the area light.

$$M_e^A = \int_{\Omega_A} L_e^A d\omega = L_e^A \int_{\Omega_A} 1 \, d\omega \tag{16}$$

$$= L_e^A 2\pi(1 - \cos(\alpha_A)) \tag{17}$$

Considering these identities for Eq. 13 yields:

$$L_e^o(\mathbf{p}, \omega_o) = \rho(\mathbf{p}) \frac{(1 - \cos^2(\alpha_A))}{2(1 - \cos(\alpha_A))} M_e^A \langle \omega_A, \vec{n}_\mathbf{p} \rangle \tag{18}$$

For $\alpha_A \to 0$, the fraction vanishes to 1 and we obtain a point lights as the limit.

### 4.2 Soft Shadow Mapping

In this section we discuss our changes and extensions to moment soft shadow mapping. Large area lights are used heavily in our scenario. Percentage closer soft shadow mapping based techniques suffer from large area lights in general. Moment soft shadow mapping suffers from large area lights even more.

#### 4.2.1 6 Moments for Shadow Mapping

The moment shadow mapping algorithm is defined for arbitrary large number of moments $m \in 2\mathbb{N}$ [10]. However, in practice $m = 4$ moments are stored in the shadow map, which allows simple distributions of three depth values to be reconstructed exactly. In large filter regions, the underlying depth distributions are often more complex,

which results in a poor approximation by only three discrete depth values, and therefore weakened shadows.

When utilizing more moments, the constructed distribution consists of a larger number of depth values, resulting in a better approximation of the original distribution. This way, a sharper estimate for the visibility test and the average blocker depth is achieved.

Making use of $m = 6$ moments reconstructs four depth values from the filtered moments. The algorithm involves finding the roots of a polynomial of degree $\frac{m}{2}$, which is hard for larger numbers of $m$. For $m = 4$, the roots of a quadratic polynomial can be computed robustly. For $m = 6$, the roots of a cubic polynomial must be computed. Peters [8] provides a robust implementation.

The weights of the depth values are computed by solving a $4 \times 4$ linear equation system. Here, utilizing the built-in matrix inverse function gave the fastest and most robust results. We also tried utilizing an explicit LU and QR decomposition in the shader code.

#### 4.2.2 Upper Bound Blocker Depth

A potentially very large filter region is considered for the estimation of the average blocker depth $z_{avg}(\mathbf{p})$. Large filter regions are likely to contain complex depth distributions, which cannot be handled well by moment shadow mapping, leading to a poor lower bound of $z_{avg}(\mathbf{p})$, which in turn will lead to an overestimated filter region for the visibility test. This will not only make shadows appear too soft, but also give a worse lower bound for the visibility test. Overestimating $z_{avg}(\mathbf{p})$, by effectively allowing self shadowing (see Eq. 19), results in hard shadows when in doubt, instead of too soft shadows.

$$z_{avg}(\mathbf{p}) = \frac{\sum_{i=1, z_i \leq d(\mathbf{p})}^{m} w_i \cdot z_i}{\sum_{i=1, z_i \leq d(\mathbf{p})}^{m} w_i} \tag{19}$$

In practice, we clamp $w_i$ to the interval $[\delta_1, \delta_2]$, for $z_i = d(\mathbf{p})$. Choosing $\delta_1 > 0$ guarantees that the nominator is always sufficiently large in order to avoid a division by zero. Choosing $\delta_1 = \delta_2 = \varepsilon$ corresponds to Eq. 7.

#### 4.2.3 Front and Back Face Shadow Maps

In this scenario potentially very large area lights are used for lighting, which requires a large depth bias in order to avoid self shadowing of tilted surfaces. On the other hand, a large depth bias results in Peter panning.

Additionally, we observed that only the silhouette edges of the blocking geometry determine the softness of shadows in the real world. Traditionally, only the front faces are rendered into the shadow map, which will always give a lower bound for the "true" silhouette edge based blocker depth. Analogous, an upper bound is obtained, when only the back faces of the geometry are used. Combining the depth of the front faces and the back faces (before construction of a filterable shadow map) leads to a depth value residing inside of the geometry, rather than on its surface.

Taking the average of the front and back face depth will lead to a better approximation of the silhouette edge based blocker depth. This also allows us to neglect the depth bias during the visibility test, because the shadow map samples are no longer located on the surface of the geometry.

This works only for convex geometry without holes. Concave geometry has to be extended by some faces in its interior, such that it effectively consists of multiple convex geometries. A major disadvantage of this approach is the fact, that rendering of the depth only shadow map is done twice before constructing a filterable shadow map.

## 5 Implementation

We implemented the light probe segmentation, as well as the area light extraction, in compute shaders in order to generate the area lights in real time from the light probe. All integrations are accelerated by summed area tables. The first step is to resolve the original values from the radiance given in the light probe, followed by a horizontal and a vertical summation step.

We need to integrate the luminance, which is the Y channel in the XYZ color space, as well as integrals on all color channels. Therefore, we use the XYZ color space to save one summed area table.

The median cut algorithm is implemented in a compute shader as well, which runs 256 threads in a single thread group working together. Each thread splits one region at a time, or none. After each iteration, the regions are sorted with a bitonic sorter [3] in order to remove "empty" entries from the regions list. Once the final regions are created, a last compute shader resolves the area lights from these regions. The area lights are then fed into a deferred renderer, applying the soft shadow mapping.

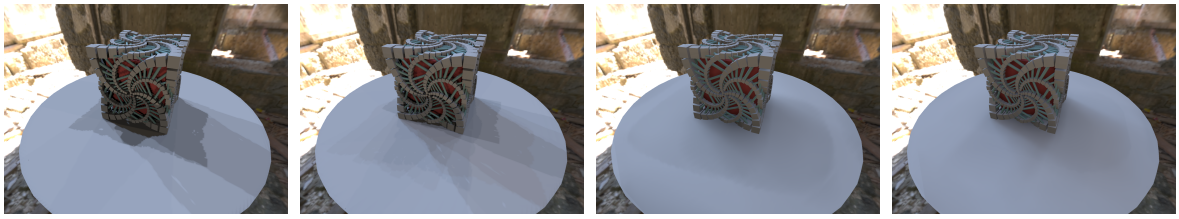
## 6 Results

In order to test the quality of the extracted area lights, path traced images are used as reference and the extracted area lights are rendered using PCSS. In addition, we compare the quality and performance of MSSM with four and six moments and CSSM to PCSS for rendering the area lights.

We implemented PCSS, CSSM and MSSM, as well as the algorithm for area light extraction in OpenGL 4.5. We used an Intel i7 4790k with 16GB RAM and an NVIDIA GeForce GTX 1080 running Ubuntu 16.04 as our test system. The runtime of individual draw calls and compute shader invocations have been measured using the Linux Graphics Debuger [7].

### 6.1 Area Light Generation

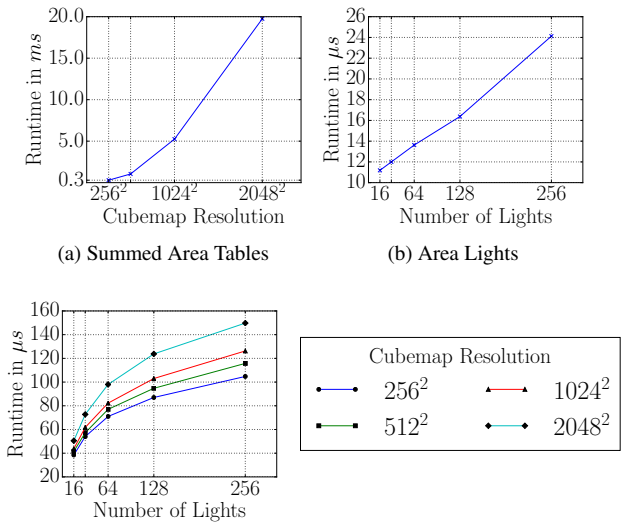
In Figure 2 we compare the illumination by point lights and area lights, extracted from a light probe, with the ground truth illumination. When using only 16 area lights



(a) 16 Point Lights (b) 32 Point Lights (c) 16 Area Lights (d) 32 Area Lights

(e) 64 Point Lights (f) 256 Point Lights (g) 64 Area Lights (h) Ground Truth

Figure 2: Scene illuminated by different amounts and types of light sources, extracted from the light probe. The ground truth (h) was rendered using a path tracer considering direct illumination only.




(a) Summed Area Tables (b) Area Lights

(c) Light Probe Segmentation

Figure 3: Runtime for generating area lights on the GPU. (a) shows the runtime for computing the SATs from the raw light probe. (c) shows the runtime for the median cut algorithm, and (b) show the runtime required to retrieve the parameters of the area lights from the final regions.

to approximate the whole light probe, very large area lights are generated, for which the soft shadow mapping technique produces artifacts (see Figure 2c). Utilizing 32 area lights results in plausible shadows. When using as much as 256 point lights from the light probe, hard shadow boundaries are still visible. Thereby, utilizing less area lights results in more plausible shadows than many point lights.

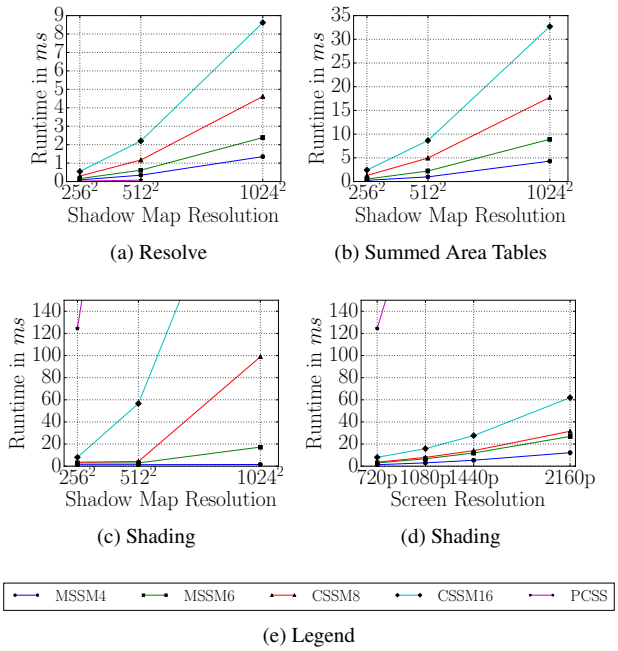
We measured the runtime of the light probe sampling algorithm on the GPU for different light probe resolutions and amounts of light sources to generate (Figure 3). The runtime for creating the actual area lights from the final regions is very low and grows linearly with the number of area lights. The runtime for creating the SATs grows quadratic with the sidelength of a cubemap face, which is as expected. Even for a cubemap resolution of $256 \times 256$, the creation of the summed area tables takes the major amount of time. The runtime of the segmentation on the other hand is little impacted by the cubemap resolution, thanks to the summed area tables, and is logarithmic in the number of area lights to create, which means that the segmentation scales well with the number of lights to create.

## 6.2 Soft Shadow Mapping

We divided runtime measurements of the whole shading process into a resolve step, a step to create the SATs and a shading step (see Figure 4). None of these steps depends on the scene complexity, since they all work in image space. The rendering of the shadow map and the rendering of the scene into the geometry buffer is not covered here, since these steps are the same for all techniques.

For CSSM$m$, $m$ refers to the number of terms stored in CSM. Since CSSM stores additional terms in the shadow map, CSSM$m$ stores $2m+1$ terms in the shadow map. For MSSM$m$, only $m$ terms are stored in the shadow map.

Since PCSS explicitly computes the convolutions in the fragment shader, the shading pass takes a lot of time, compared to MSSM and CSSM, and even more when increasing the shadow map or screen resolution. It is therefore not suitable for real-time applications in this context. Also when shading 16 area lights simultaneously, the shading pass of CSSM16 scales poorly with shadow map resolution. Only for MSSM4, the runtime of the shading pass is not affected by a shadow map resolution of $1024^2$.




(a) Resolve (b) Summed Area Tables

(c) Shading (d) Shading

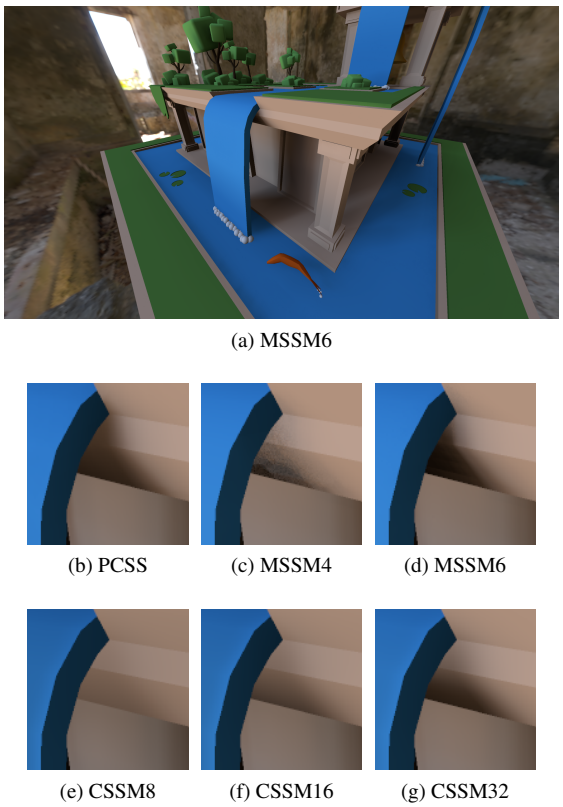
(e) Legend

Figure 4: Runtimes for applying the shading for 16 area lights simultaneously. In (a) the multisampled front- and back-face shadow maps are combined and the filterable shadow maps are created. In (b) the SATs are computed for the CSSM and MSSM techniques. (c) and (d) show the runtime for the actual shading, depending on the shadow map resolution and screen resolution, respectively. In (c) the screen resolution is locked to $1280 \times 720$ and in (d) the shadow map resolution is locked to $256^2$ pixels.

CSSM8 and CSSM16 give reasonable results, even though the overall appearance of the scene is slightly darkened. CSSM8 stores 17 and CSSM16 33 values in the shadow map, resulting in a poor runtime, compared to MSSM4, which stores only 4 values in the shadow map and MSSM6, which stores 6 values. The MSSM techniques score a better runtime in the shading pass although they are computationally more intensive.

The variants of MSSM are compared in Figure 7. The shadows from MSSM4 are quite weak. When using 6 moments, the shadows are stronger, since a sharper bounds are achieved. In both cases, using an upper bound for the average blocker depth makes the shadows less weak.

When using a large shadow map resolution in conjunction with large area lights, the maximum filter region for MSSM is problematic, since a given filter region in normalized coordinates spans a larger number of pixels in the shadow map. This leads to filtered hard shadows instead of plausible soft shadows in some cases. Since the shadows caused by large area lights are very smooth anyway, aliasing is less of an issue. We used a shadow map resolution of $256^2$ and a maximum filter size of $27^2$ pixels for all techniques with $4\times$MSAA where applicable.



(a) MSSM6

(b) PCSS (c) MSSM4 (d) MSSM6

(e) CSSM8 (f) CSSM16 (g) CSSM32

Figure 5: Comparison of soft shadow mapping techniques. The light probe is approximated by 64 area lights.

## 7 Conclusion

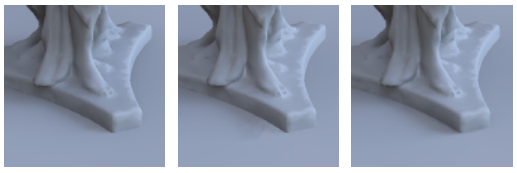
We demonstrated the use of MSSM for efficiently casing plausible soft shadows from area lights and compared it to PCSS and CSSM. When generating light sources from a light probe, using fewer area lights results in more plausible shadows than using more point lights.

Our results show that MSSM is faster than CSSM and has a lower memory consumption. Due to the large size of the sampled area lights, PCSS requires a huge computational effort and cannot compete with neither MSSM nor CSSM. However, MSSM has problems with very large area lights as well. We demonstrated, that significantly sharper bounds of the visibility test can be achieved for complex depth distributions, by using six moments in the shadow map. Also, using an upper bound for the average blocker depth helps a lot with handling complex depth distributions. This way, hard shadows are approached instead of missing shadows in challenging situations.

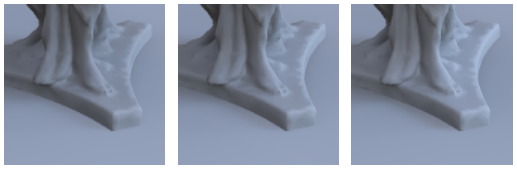
The algorithm for extracting area lights from light probes we presented scales well with light probe resolution and the number of area lights to extract from the light probe. Furthermore the energy of the light probe is preserved, which avoids large changes in illumination, while the incident light changes only a little. This allows to illuminate our scene by animated light probes.

Future work would involve, rendering the light probe

(a) MSSM6



(b) PCSS (c) MSSM4 (d) MSSM6



(e) CSSM8 (f) CSSM16 (g) CSSM32

Figure 6: Comparison of soft shadow mapping techniques. The light probe is approximated by 64 area lights.

in real time instead of loading precomputed light probes onto the GPU. In doing so, the depth buffer of the light probe would be available, and could be used to place area lights at finite positions. Placing the light sources at finite positions causes the shadow boundaries to approach a finite point. In comparison, the shadow boundaries of directional area lights are parallel to each other.

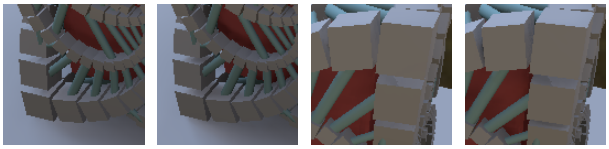
## References


[1] Thomas Annen et al. *Convolution shadow maps*. In Proceedings of the 18th Eurographics conference on Rendering Techniques, pages 51–60. Eurographics Association, 2007.

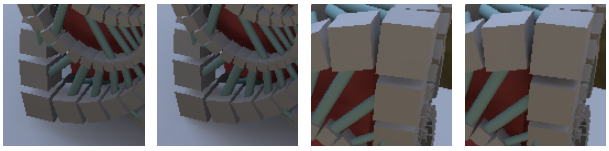
[2] Thomas Annen et al. *Real-time, all-frequency shadows in dynamic scenes*. In ACM transactions on graphics (TOG), volume 27, page 34. ACM, 2008.

[3] Kenneth E Batcher. *Sorting networks and their applications*. In Proceedings of the April 30–May 2, 1968, spring joint computer conference, pages 307–314. ACM, 1968.

[4] Paul Debevec. *A median cut algorithm for light probe sampling*. In ACM SIGGRAPH 2008 classes, page 33. ACM, 2008.

(a) MSSM4 LB (b) MSSM4 UB (c) MSSM4 LB (d) MSSM4 UB



(e) MSSM6 LB (f) MSSM6 UB (g) MSSM6 LB (h) MSSM6 UB

Figure 7: Comparison of upper (UB) and lower (LB) bound of $z_{avg}(\mathbf{p})$ with MSSM4 and MSSM6.


[5] Randima Fernando. *Percentage-closer soft shadows*. In ACM SIGGRAPH 2005 Sketches, page 35. ACM, 2005.

[6] James T Kajiya. *The rendering equation*. In ACM Siggraph Computer Graphics, volume 20, pages 143–150. ACM, 1986.

[7] NVIDIA. *Linux Graphics Debugger*. `https://developer.nvidia.com/linux-graphics-debugger`. (visited on 09/28/2016).

[8] Christoph Peters. *How to solve a cubic equation, revisited*. `http://www.momentsingraphics.de/?p=105`, 2016. (visited on 09/28/2016).

[9] Christoph Peters et al. *Beyond hard shadows: moment shadow maps for single scattering, soft shadows and translucent occluders*. In Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pages 159–170. ACM, 2016.

[10] Christoph Peters and Reinhard Klein. *Moment shadow mapping*. In Proceedings of the 19th Symposium on Interactive 3D Graphics and Games, pages 7–14. ACM, 2015.

[11] Peter-Pike Sloan, Jan Kautz, and John Snyder. *Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments*. In ACM Transactions on Graphics (TOG), volume 21, pages 527–536. ACM, 2002.

[12] Kuntee Viriyothai and Paul Debevec. *Variance minimization light probe sampling*. In SIGGRAPH'09: Posters, page 92. ACM, 2009.

[13] Lance Williams. *Casting curved shadows on curved surfaces*. In ACM Siggraph Computer Graphics, volume 12, pages 270–274. ACM, 1978.