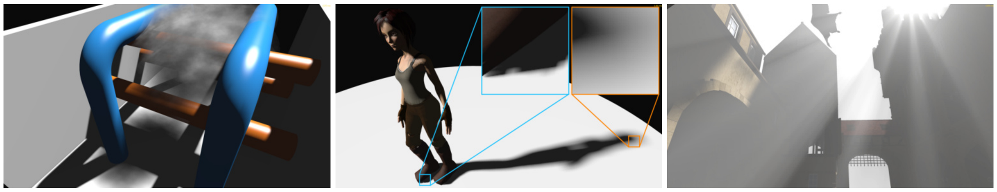
Improved Moment Shadow Maps for Translucent Occluders, Soft Shadows and Single Scattering

Christoph Peters Cedrick Münstermann Nico Wetzstein Reinhard Klein
University of Bonn, Germany



(a) *Translucent occluders, 4.1 ms* **(b)** *Soft shadows, 3.5 ms* **(c)** *Single scattering, 4 ms*

Figure 1. We use moment shadow maps to render shadows for translucent occluders, for area lights and in participating media. The timings shown are full frame times for rendering at 3840 × 2160. (a) and (b) use 4× multisample antialiasing.

Abstract

Like variance shadow maps, the recently proposed moment shadow maps can be filtered directly but they provide a substantially higher quality. We combine them with earlier approaches to enable three new applications. Shadows for translucent occluders are obtained by simply rendering to a moment shadow map with alpha blending. Soft shadows in the spirit of percentage-closer soft shadows are rendered using two queries to a summed-area table of a moment shadow map. Single scattering is rendered through one lookup per pixel in a prefiltered moment shadow map with six channels. As a foundation we also propose improvements to moment shadow mapping itself. All these techniques scale particularly well to high output resolutions and enable proper antialiasing of shadows through extensive filtering.

1. Introduction

Moment shadow maps [Peters and Klein 2015] are a recently proposed kind of filterable shadow maps. Like variance shadow maps [Donnelly and Lauritzen 2006] and other filterable shadow maps, they can be filtered directly. Thus, they enable the rendering of antialiased shadows at a low cost per shaded fragment. Compared to other filterable shadow maps, there are less artifacts and the technique only uses 64 bits per texel for the four channels of the moment shadow map.

Filtered hard shadows are the most immediate application [Peters and Klein 2015] but applications of earlier filterable shadow maps are far more diverse. Translucent occluders may be rendered to filterable shadow maps directly to get plausible shadows [Delalandre et al. 2011; McGuire and Mara 2016]. Percentage-closer soft shadowing [Fernando 2005] filters hard shadows with varying filter size to approximate soft shadows. Using filterable shadow maps and summed-area tables [Crow 1984] this may be done at a fixed cost [Lauritzen 2007; Annen et al. 2008a; Yang et al. 2010; Shen et al. 2013]. Even the accumulation of single scattering along a view ray has been precomputed into filterable shadow maps [Klehm et al. 2014].

In the present paper we demonstrate that all these approaches are compatible with moment shadow mapping. As part of the derivation, we develop novel building blocks such as a moment-based blocker search and moment shadow mapping with six moments. The resulting techniques provide very attractive tradeoffs between quality and run time. They scale particularly well to high output resolutions because they impose a low, fixed overhead per shaded fragment.

As prerequisite of this work, we review moment shadow mapping in Section 2, where we also present improved quantization and biasing schemes that further improve robustness and speed of the technique. In Section 3 we demonstrate that shadows for translucent occluders can be rendered by simply rendering to a moment shadow map with alpha blending. Section 4 covers moment soft shadow mapping, which renders approximate soft shadows using only two queries to a summed-area table per fragment. Finally, we present prefiltered single scattering with four or six moments in Section 5. Sections 3, 4 and 5 may be read independently but all of them rely on notions introduced in Section 2.

The present paper is an invited extension of our earlier work [Peters et al. 2016] with various novel contributions. In particular, we improve upon core aspects of moment shadow mapping (Sections 2.3 to 2.5), increase robustness of the blocker search for moment soft shadow mapping (Section 4.3), describe a simple method to diminish leaking artifacts in prefiltered single scattering (Section 5.3.1) and optimize six moment shadow mapping for greater robustness and speed (Section 5.4). We also discuss the implementation in greater detail and provide more comparisons.

2. Improved Moment Shadow Maps

In the following we introduce some notions that are used throughout the paper and use them to review moment shadow mapping and its related work. We also demonstrate improvements to moment shadow mapping that reduce light leaking when using 128 bits per texel and make the technique faster and more robust.

2.1. Depth Distributions

Depth distributions offer a useful way to formally model shadow map filtering [Donnelly and Lauritzen 2006; Peters and Klein 2015]. Suppose we are given a shadow map and want to apply a specific filter (e.g., a Gaussian) to compute a filtered shadow intensity for a fragment at depth $z_f \in \mathbb{R}$. The filter kernel used has $n \in \mathbb{N}$ weights $w_0, \ldots, w_{n-1} > 0$. The corresponding depths sampled within the filter region are $z_0, \ldots, z_{n-1} \in \mathbb{R}$. These quantities define a depth distribution that we write using Dirac-δ distributions:

$$Z := \sum_{l=0}^{n-1} w_l \cdot \delta_{z_l}.$$

This notation suggests a probabilistic interpretation. The filter region is sampled at random such that the probability to draw depth z_l is given by the filter weight w_l. Now we map each depth z to some vectorial quantity $\mathbf{b}(z) \in \mathbb{R}^{m+1}$ where $m \in \mathbb{N}$. Using the random depth as input, the expectation of the output is given by

$$b := \mathcal{E}_Z(\mathbf{b}) := \sum_{l=0}^{n-1} w_l \cdot \mathbf{b}(z_l) \in \mathbb{R}^{m+1}.$$

Filterable shadow maps are constructed exactly in this manner. Rather than storing z, each texel stores $\mathbf{b}(z)$ for some function $\mathbf{b} : \mathbb{R} \to \mathbb{R}^{m+1}$. Filtering the filterable shadow map with the above filter kernel then yields $b = \mathcal{E}_Z(\mathbf{b})$. The function $\mathbf{b}$ is chosen such that knowledge of b enables an approximate reconstruction of Z.

2.2. Related Work

Percentage-closer filtering [Reeves et al. 1987] reconstructs the depth distribution through brute force by taking $n \in \mathbb{N}$ samples from the shadow map. The shadow intensity at depth z_f is given by the weighted fraction of samples which are closer to the light than the fragment:

$$Z(\mathbf{z} < z_f) := \sum_{l=0}^{n-1} w_l \cdot \begin{cases} 1 & \text{if } z_l < z_f, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{where} \quad \mathbf{z}(z){:=}z.$$

Variance shadow maps [Donnelly and Lauritzen 2006] are filterable shadow maps using $\mathbf{b}(z) := (1, z, z^2)^\mathsf{T}$. The channel $\mathcal{E}_Z(\mathbf{b}_0) = 1$ does not need to be stored. The other two channels store two moments which provide mean and variance of Z and enable a reconstruction through Cantelli's inequality:

$$\mu := b_1, \quad \sigma^2 := b_2 - b_1^2, \quad Z(\mathbf{z} < z_f) \geq 1 - \frac{\sigma^2}{\sigma^2 + (z_f - \mu)^2} \quad \text{if } z_f \geq \mu.$$

Similarly, exponential shadow maps [Salvi 2008; Annen et al. 2008b] use $\mathbf{b}(z) = (1, \exp(c_{\text{esm}} \cdot z))^\mathsf{T}$ where $c_{\text{esm}} \gg 1$ and reconstruct through Markov's inequality:

$$Z(\mathbf{z} < z_f) \geq 1 - \frac{b_1}{\exp(c_{\text{esm}} \cdot z_f)}.$$

Convolution shadow maps [Annen et al. 2007] set the component functions of $\mathbf{b}$ to Fourier basis functions and use a truncated Fourier series. Exponential variance shadow maps [Lauritzen and McCool 2008] fix parameters $c^+_{\text{evsm}}, c^-_{\text{evsm}} > 0$ and apply variance shadow mapping to two exponentially warped depths, i.e.,

$$\mathbf{b}(z) = (1, \exp(c^+_{\text{evsm}} \cdot z), \exp(c^+_{\text{evsm}} \cdot z)^2, -\exp(-c^-_{\text{evsm}} \cdot z), \exp(-c^-_{\text{evsm}} \cdot z)^2)^\mathsf{T}.$$

2.2.1. *Moment Shadow Mapping*

Moment shadow mapping [Peters and Klein 2015] uses

$$\mathbf{b}(z) = (z^j)_{j=0}^m = (1, z, z^2, z^3, z^4)^\mathsf{T}.$$

The number of moments $m \in \mathbb{N}$ can be any even number but the original technique only uses $m = 4$. In Section 5.4 we demonstrate the use of $m = 6$. As above $b_0 = \mathcal{E}_Z(1) = 1$ does not need to be stored.

When shading a fragment at depth $z_f \in \mathbb{R}$, we need to estimate the shadow intensity $Z(\mathbf{z} < z_f)$, but we are only given a filtered sample from the moment shadow map $b = \mathcal{E}_Z(\mathbf{b})$. In general, there are many possible depth distributions S on $\mathbb{R}$ that are compatible with these moments, i.e., $b = \mathcal{E}_S(\mathbf{b})$. To diminish wrong self-shadowing (also known as surface acne), we use the reconstruction that yields the smallest shadow intensity, i.e., the estimated shadow intensity is

$$\inf\{S(\mathbf{z} < z_f) \mid S \text{ depth distribution on } \mathbb{R} \text{ with } \mathcal{E}_S(\mathbf{b}) = b\}. \tag{1}$$

This minimization problem is non-trivial but has been solved [Kreĭn and Nudel'man 1977]. The optimal depth distribution S always uses only the depth value z_f and $\frac{m}{2} = 2$ other depth values as shown in Figure 2. Such depth distributions are unique and Algorithm 1 computes them efficiently. In this algorithm, Steps 1 to 3 compute the $\frac{m}{2}$ unknown depth values [Peters and Klein 2015, Proposition 10, supplementary, p. 2]. Step 4 and 5 determine the weights that produce the correct moments. Finally, Step 6 constructs and returns the optimal depth distribution S.

Algorithm 1 cannot work correctly if the matrix $B(b)$ is not positive definite. However, the matrix is known to be positive semi-definite for all meaningful inputs $b = \mathcal{E}_Z(\mathbf{b})$. Invalid inputs may arise due to rounding errors. These rounding errors are counteracted by means of linear interpolation towards a fixed constant vector using a small interpolation weight $0 < \alpha_b \ll 1$. This biasing should be kept to a minimum because it makes the approximation less accurate and thus increases light leaking.

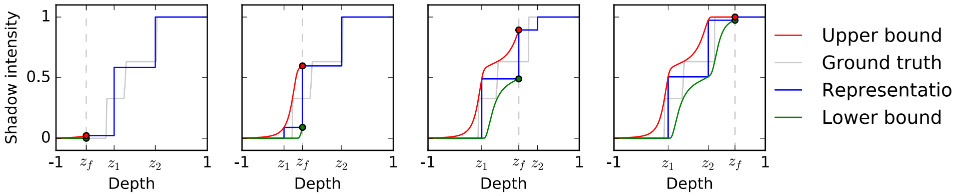


Figure 2. Examples of optimal solutions to Equation (1) for a single ground truth. The ground truth and the four representations share the same moments $b_0, b_1, \ldots, b_4$. The upper and lower bounds touch the representations at the respective z_f. Note that all representations use exactly three depth values z_f, z_1, z_2.

Algorithm 1 Moment shadow mapping, i.e., the solution to Equation (1).
Input: Moments $b \in \mathbb{R}^{m+1}$ for even $m \in \mathbb{N}$ and the fragment depth $z_f \in \mathbb{R}$.
Output: A depth distribution S minimizing the expression in Equation (1).

```
1. Set $B(b) := (b_{j+k})_{j,k=0}^{\frac{m}{2}} = \begin{pmatrix} b_0 & b_1 & \cdots & b_{\frac{m}{2}} \\ b_1 & b_2 & \iddots & b_{\frac{m}{2}+1} \\ \vdots & \iddots & \iddots & \vdots \\ b_{\frac{m}{2}} & b_{\frac{m}{2}+1} & \cdots & b_m \end{pmatrix} \in \mathbb{R}^{(\frac{m}{2}+1)\times(\frac{m}{2}+1)}$.
2. Solve $B(b) \cdot q = (1, z_f^1, \ldots, z_f^{\frac{m}{2}})^\mathsf{T}$ for $q \in \mathbb{R}^{\frac{m}{2}+1}$.
3. Solve the polynomial equation $\sum_{j=0}^{\frac{m}{2}} q_j \cdot z^j = 0$ for $z$ and denote the distinct
   solutions by $z_1, \ldots, z_{\frac{m}{2}} \in \mathbb{R}$.
4. Set $z_0 := z_f$ and $A := (z_l^j)_{j,l=0}^{\frac{m}{2}} = \begin{pmatrix} z_0^0 & \cdots & z_{\frac{m}{2}}^0 \\ \vdots & \ddots & \vdots \\ z_0^{\frac{m}{2}} & \cdots & z_{\frac{m}{2}}^{\frac{m}{2}} \end{pmatrix} \in \mathbb{R}^{(\frac{m}{2}+1)\times(\frac{m}{2}+1)}$.
5. Solve $A \cdot w = (b_0, b_1, \ldots, b_{\frac{m}{2}})^\mathsf{T}$ for $w \in \mathbb{R}^{\frac{m}{2}+1}$.
6. Return $\sum_{l=0}^{\frac{m}{2}} w_l \cdot \delta_{z_l}$.
```

When using only 16 bits for each of the four moments, the rounding errors are too strong. To overcome this problem, an affine transform is applied to the vector $(b_1, b_2, b_3, b_4)^\mathsf{T}$ before its entries are stored in 16-bit fixed-point numbers. The transform is the result of a numerical optimization that maximizes its determinant without violating the representable range of the output data type. The entropy of the stored data grows with the base-2 logarithm of this determinant and therefore loss of information is minimized.

2.3. Signed Depth

Using the quantization transform does not help when moments are stored in single-precision floats because the moments have to be transformed back before providing them to Algorithm 1 in single precision. Still, we want to minimize the increased light leaking that arises from the stronger biasing needed to compensate for the rounding errors.

To this end, we can remap the range of depth values using a simple linear transform. Originally, it was proposed to define depth in the interval $[0, 1]$ [Peters and Klein 2015]. We found that this arbitrary choice is substantially worse than the optimal choice, which is $[-1, 1]$. The near clipping plane of the shadow map is defined to be at depth -1 whereas the far clipping plane is located at depth 1.

Our reasoning in favor of this definition is based on the effect of linear transforms on the moments. Let $x, y \in \mathbb{R}$ with $x \neq 0$ describe a linear transform of depth values $x \cdot z + y$. It induces a linear transform for the moments:

$$\mathcal{E}_Z\left((x \cdot \mathbf{z} + y)^j\right) = \mathcal{E}_Z\left(\sum_{k=0}^{j} \binom{j}{k} \cdot (x \cdot \mathbf{z})^k \cdot y^{j-k}\right) = \sum_{k=0}^{j} \binom{j}{k} \cdot x^k \cdot y^{j-k} \cdot b_k. \quad (2)$$

This linear transform corresponds to a lower triangular matrix with diagonal entries $x^0, \ldots, x^m$. Therefore, its determinant is $\prod_{j=1}^{m} x^j$.

Suppose we define depth in the interval $[-1, 1]$ as proposed. If we apply the above transform, the resulting j-th moment can have a magnitude up to $(|x| + |y|)^j$. Since we are using floating point numbers, the relative precision is not reduced if we divide by this magnitude to get back to a maximal magnitude of one. Combining this with the previous transform yields a determinant of

$$\prod_{j=1}^{m} \left(\frac{x}{|x| + |y|}\right)^j.$$

Obviously, the magnitude of this determinant is maximized by choosing $y = 0$ and then it is invariant under changes of x.

The transform in Equation (2) acts like a special quantization transform. Maximizing its determinant has a positive impact on the entropy of the data stored in the moment shadow map[1] [Peters and Klein 2015]. Thus, the choice of the depth interval $[-1, 1]$ is indeed optimal. For $m = 4$ the transform mapping moments of depth distributions on $[0, 1]$ to moments of depth distributions on $[-1, 1]$ has a determinant of

[1] Strictly speaking this only holds for fixed-point numbers [Peters and Klein 2015] but for robustness reasons we should consider the worst case and thus we may view floating-point numbers as fixed-point numbers with the worst case precision.

1024 which corresponds to an approximate increase in entropy of $\log_2 1024 = 10$ bits. When using 128 bits per texel of the moment shadow map, this diminishes the negative effects of rounding errors. A lower bias can be used and light leaking is reduced. At the same time, this change does not induce a significant cost and is easy to implement.

2.4. Sparse Quantization Transform

When using a moment shadow map with four 16-bit fixed-point channels, the rounding errors introduced during storage outweigh the rounding errors arising in Algorithm 1 by far. These rounding errors are minimized most efficiently by using an arbitrary quantization transform. Thus, using signed depth does not improve the precision. However, it does enable an optimization.

The cost for application of the original quantization transform [Peters and Klein 2015] is significant. It requires multiplication of a vector by a dense 4×4 matrix per texel of the moment shadow map and per shaded fragment. Compared to direct storage of moments arising from depth values in $[-1, 1]$, this globally optimal transform increases the entropy by 4.28 bits.

We found that the search space in the optimization of the transform can be constrained substantially without a large reduction in entropy. In particular, the transform

$$\Theta_4^\star \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} := \begin{pmatrix} \frac{3}{2} & 0 & -2 & 0 \\ 0 & 4 & 0 & -4 \\ \frac{1}{2} \cdot \sqrt{3} & 0 & -\frac{2}{9} \cdot \sqrt{3} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{pmatrix} \tag{3}$$

still increases entropy by 4.21 bits. Therefore, the quality of the results is practically unchanged but since half of the entries are zero, application of this transform is roughly twice as fast. The odd and even moments are transformed separately as shown in Figure 3.

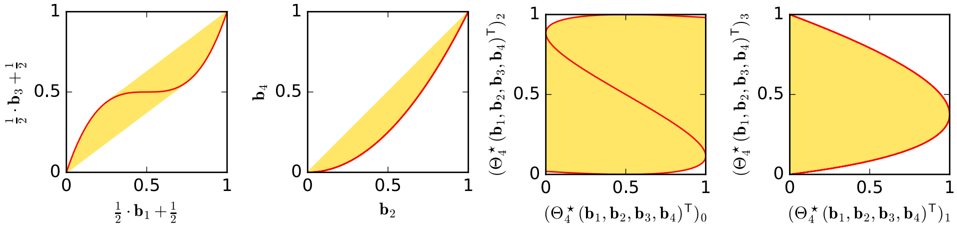
2.5. Biasing for the Worst Case

We have also improved the original biasing scheme. Speaking in terms of moments from depth values in $[-1, 1]$, the original scheme replaces a vector of moments $b \in \mathbb{R}^5$, which is corrupted by rounding errors, by the vector

$$b' := (1 - \alpha_b) \cdot b + \alpha_b \cdot b^\star \quad \text{where} \quad b^\star := (1,\, 0,\, 1,\, 0,\, 1)^\mathsf{T} \in \mathbb{R}^5.$$

This scheme has been derived to offer best results in the average case [Peters and Klein 2015].

Though, there is a relevant worst case where it fails. Depth values may be clamped to the interval $[-1, 1]$ to better utilize the depth range without clipping relevant shadow



(a) *Naïve quantization* **(b)** *Optimized quantization* $\Theta_4^\star$

Figure 3. A visualization of the quantization transform in Equation (3). The red line is the curve indicated by the axis labels. The yellow area is its convex hull, which contains the filtered values. Note that the convex hull has a substantially larger area for the optimized quantization and that the graphs for the optimized quantization arise from the ones for naïve quantization through a simple affine transform.

casters. In this case, depth distributions of the form

$$Z = (1 - w_1) \cdot \delta_{-1} + w_1 \cdot \delta_1$$

might arise, i.e., the filter region contains solely clamped depth values. To demonstrate why this is a problem, we write the vector $b^\star$ as $b^\star = \mathcal{E}_{Z^\star}(\mathbf{b})$ where

$$Z^\star = \frac{1}{2} \cdot \delta_{-1} + \frac{1}{2} \cdot \delta_1.$$

Then the biased vector b' corresponds to the depth distribution

$$(1-\alpha_b)\cdot Z+\alpha_b\cdot Z^\star = \left((1-\alpha_b)\cdot(1-w_1)+\frac{\alpha_b}{2}\right)\cdot\delta_{-1}+\left((1-\alpha_b)\cdot w_1+\frac{\alpha_b}{2}\right)\cdot\delta_1.$$

Obviously, this depth distribution only uses the two depth values -1 and 1. Such depth distributions correspond to vectors of moments on the boundary of the valid domain [Peters and Klein 2015, p. 11]. Arbitrarily small rounding errors may invalidate them. The biasing has failed to accomplish its purpose. When clamping depth values, this leads to visible instabilities in some places.

To derive a bias that behaves robustly in all cases, we ask for maximal efficiency in the worst case. More precisely, we ask for a vector of moments $b^\star \in \operatorname{conv} \mathbf{b}([-1, 1])$ (i.e., it can be represented by a depth distribution on $[-1, 1]$ [Peters and Klein 2015]) having maximal distance to the boundary of the valid domain $\operatorname{conv} \mathbf{b}(\mathbb{R})$. For the originally proposed vector $b^\star$ this distance would be zero.

We find this vector by means of a brute-force search in a reduced search space. This reduction exploits that $\mathbf{b}(\mathbb{R})$ and $\mathbf{b}([-1, 1])$ have a mirror symmetry along each odd dimension. The result of the optimization is

$$b^\star = (1,\ 0,\ 0.375,\ 0,\ 0.375)^\mathsf{T}.$$

This is the biasing scheme that we use in absence of a quantization transform. When the moments are stored in single-precision floats, we found that a moment bias of $\alpha_b = 3 \cdot 10^{-7}$ is sufficient.

When the quantization transform in Equation (3) is used, the distance to the boundary of $\operatorname{conv} \mathbf{b}(\mathbb{R})$ should be computed after application of $\Theta_4^\star$ because the strong rounding errors apply to the transformed moments. This distorted metric leads to the optimum

$$b^\star = (1,\, 0,\, 0.628,\, 0,\, 0.628)^\mathsf{T}.$$

Using 64 bits per texel, we observed that a moment bias of $\alpha_b = 6 \cdot 10^{-5}$ eliminates artifacts reliably.

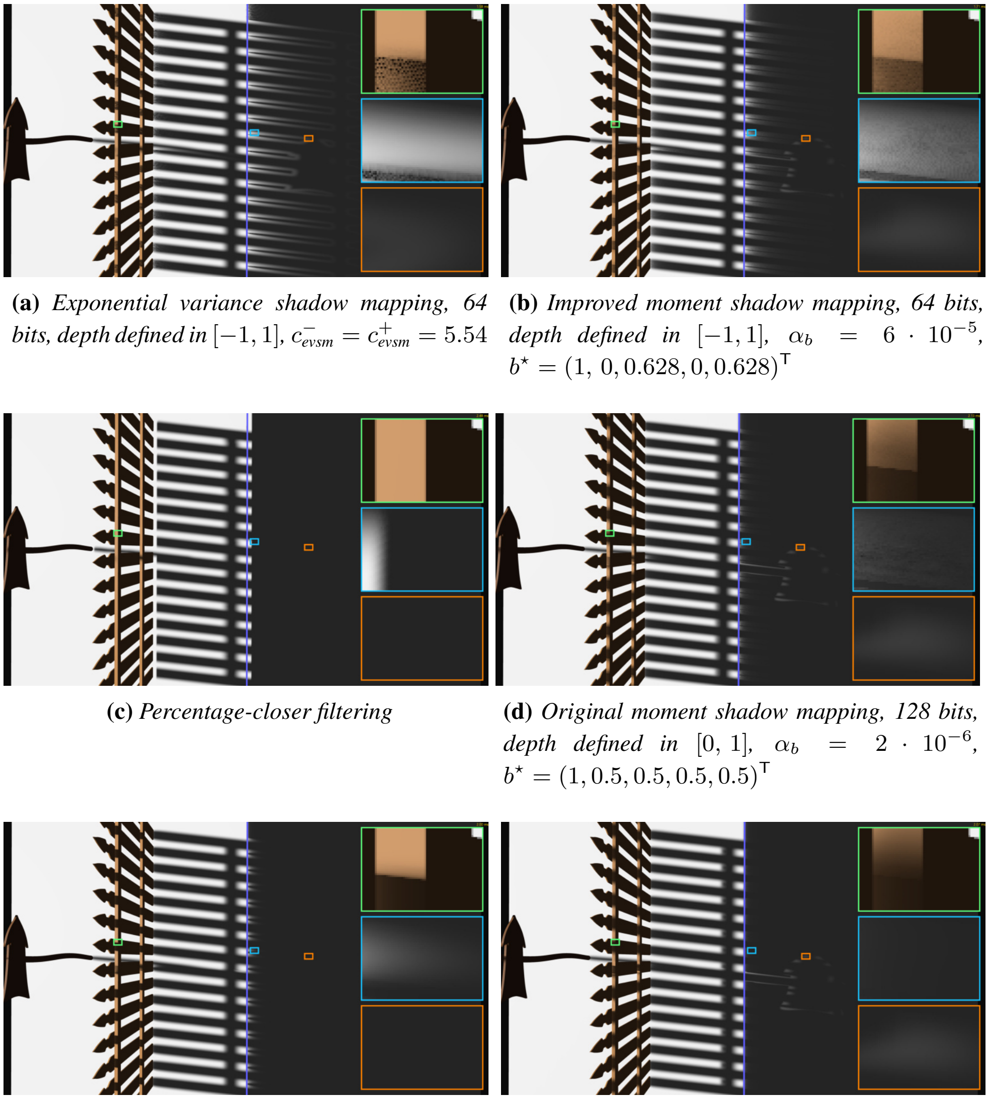
2.6. Results and Discussion

For more details on our implementation we refer to Appendices D.1 and D.2. We note that all imagery in this paper uses the sRGB color space. Images in previous publications inappropriately used a linear color space [Peters and Klein 2015; Peters et al. 2016]. Since the sRGB conversion increases small values, it tends to make faint light leaking noticeable. Over darkening, proposed by Annen et al. [2007], diminishes this artifact. For example, shadow intensities may be divided by 98% and then clamped to $[0,\, 1]$ such that light leaking below 2% disappears. We do this for the surface shadows in all figures except for Figures 4 and 6, which exhibit light leaking deliberately.

Figure 4 compares several techniques on a scene that is chosen to provoke artifacts. Even percentage-closer filtering exhibits some missing contact shadows due to the required depth bias (Figure 4c). Moment shadow mapping with 64 bits per texel suffers from slight quantization noise and light leaking in short-range shadows (Figure 4b). When using 128 bits with depths in $[0,\, 1]$, both artifacts are much weaker but still present (Figure 4d). When depth values are defined in $[-1, 1]$, these artifacts vanish almost entirely (Figure 4f).

At 64 bits per texel exponential variance shadow mapping yields substantially stronger quantization noise and light leaking than moment shadow mapping (Figs. 4a and 4b). At 128 bits per texel the two techniques have different advantages. Exponential variance shadow mapping has difficulties with shadows near boundaries of shadow casters (Figure 4e, green inset) and still exhibits noticeable light leaking (Figure 4e, cyan inset). Moment shadow mapping does not exhibit these artifacts but suffers from light leaking when three distinct surfaces are present in the filter region (Figure 4f, orange inset).

The reduction in run time due to the sparse quantization transform is not always significant because it may be covered up by other bottlenecks. For a significant measurement, we use a close-up of the smoke in Figure 5 rendered at 3840×2160 to get strong overdraw. In this setting the optimized quantization transform reduces the overhead for moment shadow mapping with 64 bits per texel from 5.2 ms to 4.7 ms.



(a) *Exponential variance shadow mapping, 64 bits, depth defined in* $[-1,1]$, $c^-_{evsm} = c^+_{evsm} = 5.54$

(b) *Improved moment shadow mapping, 64 bits, depth defined in* $[-1,1]$, $\alpha_b = 6 \cdot 10^{-5}$, $b^\star = (1,\ 0, 0.628, 0, 0.628)^\mathsf{T}$

(c) *Percentage-closer filtering*

(d) *Original moment shadow mapping, 128 bits, depth defined in* $[0,\ 1]$, $\alpha_b = 2 \cdot 10^{-6}$, $b^\star = (1, 0.5, 0.5, 0.5, 0.5)^\mathsf{T}$

(e) *Exponential variance shadow mapping, 128 bits, depth defined in* $[-1,1]$, $c^-_{evsm} = 5.54$, $c^+_{evsm} = 40$

(f) *Improved moment shadow mapping, 128 bits, depth defined in* $[-1,1]$, $\alpha_b = 3 \cdot 10^{-7}$, $b^\star = (1, 0, 0.375, 0, 0.375)^\mathsf{T}$

Figure 4. A bird's eye view of a scene with a direction sign, a fence and a thin wall. The three magnified insets receive shadow from the fence only (green), the fence and the thin wall (cyan) or the direction sign, the fence and the thin wall (orange). This scenario provokes various artifacts.

3. Shadows for Translucent Occluders

Filterable shadow maps are compatible with all sorts of linear filtering. Alpha blending is one such operation. In this section, we show that rendering translucent geometry to a filterable shadow map with alpha blending is reasonable and we demonstrate that it works particularly well with moment shadow mapping. The same moment shadow map is used for opaque and translucent occluders. Thus, there is almost no overhead.

3.1. Related Work

Transmittance from the light to a surface can depend upon the depth in complex ways when translucent occluders are present. It has been represented as a piecewise linear function but then filtering is non-trivial [Salvi et al. 2010]. Other approaches use percentage-closer filtering and randomly discard fragments in proportion to their translucency [Enderton et al. 2010; McGuire and Enderton 2011].

Fourier opacity mapping [Jansen and Bavoil 2010] introduced the idea of using filterable shadow maps, namely convolution shadow maps [Annen et al. 2007]. The absorption function (i.e., the logarithm of the transmittance) is represented by a Fourier series. Absorption can be accumulated additively and thus no sorting is needed. Translucent shadow maps [Delalandre et al. 2011] take a similar approach but represent the transmittance function such that sorting is needed. Phenomenological scattering [McGuire and Mara 2016] represents transmittance through a variance shadow map but avoids sorting by stochastically discarding fragments. This technique also adds heuristic caustics.

3.2. Moment Shadow Maps for Translucent Occluders

Our approach is like translucent shadow maps in that the moment shadow map represents transmittance. Representing an absorption function would require an additional channel for the total absorption b_0. Besides we want to use a single moment shadow map for opaque and translucent occluders but opaque occluders correspond to infinite absorption.

The disadvantage of this choice is that we require a method for order-independent transparency when rendering to the moment shadow map. We consider this orthogonal to our contribution and any existing method should work (e.g., stochastic transparency [Enderton et al. 2010]). Our experiments rely on sorted geometry.

We now demonstrate that alpha blending produces the vector of moments of a depth distribution Z modeling transmittance of translucent occluders correctly. Given $n_s \in \mathbb{N}$ surfaces along a light ray at depths $z_0 < z_1 < \ldots < z_{n_s-1}$ with opacities $\alpha_0, \ldots, \alpha_{n_s-1} \in [0, 1]$, the amount of light transmitted to depth $z_f \in \mathbb{R}$ is the product of the relevant transmittance factors $\prod_{k=0,\, z_k<z_f}^{n_s-1}(1-\alpha_k)$. This transmittance is

precisely modeled by the depth distribution

$$Z := \sum_{j=0}^{n_s-1} \left(\prod_{k=0}^{j-1} 1 - \alpha_k \right) \cdot \alpha_j \cdot \delta_{z_j}$$

because at depth z_j the fraction α_j of the remaining light is blocked.

Suppose we render these surfaces back to front to a moment shadow map using the over operator for blending. It is safe to assume $z_{n_s-1} = \alpha_{n_s-1} = 1$ because we clear the moment shadow map accordingly. When rendering surface $j \in \{0, \ldots, n_s - 1\}$, the source color is multiplied by α_j, the destination color is multiplied by $(1 - \alpha_j)$ and the results are added. Thus, the vector of moments $\mathbf{b}(z_j)$ is first multiplied by α_j and subsequently by $(1 - \alpha_{j-1})$, ..., $(1 - \alpha_0)$. In the end, we obtain the vector of moments

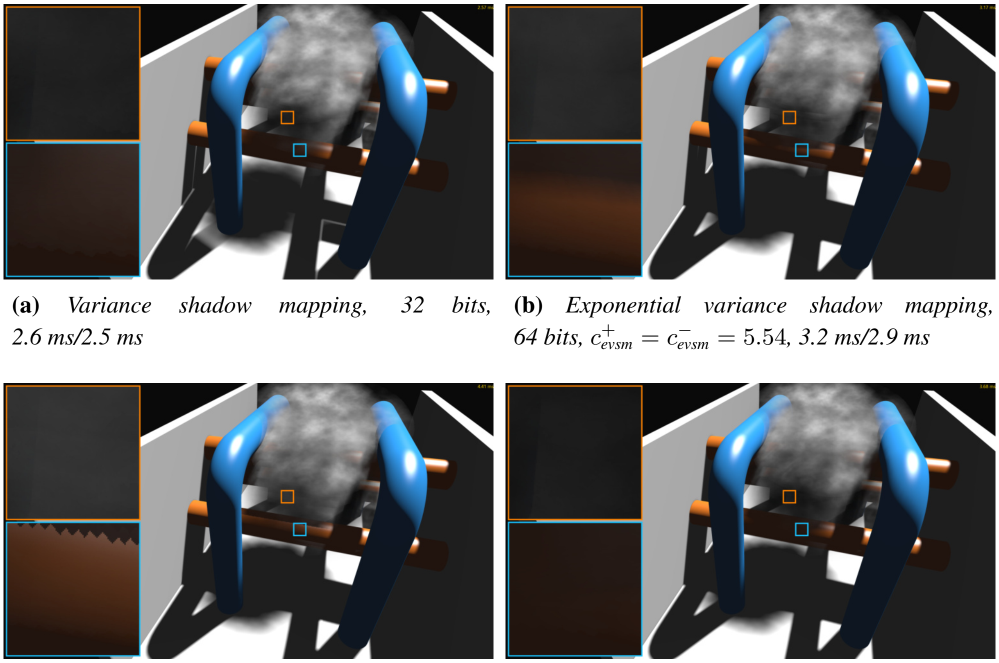
$$b := \sum_{j=0}^{n_s-1} \left(\prod_{k=0}^{j-1} 1 - \alpha_k \right) \cdot \alpha_j \cdot \mathbf{b}(z_j) = \mathcal{E}_Z(\mathbf{b})$$

which is exactly the sought-after result. Approximation errors are only introduced during reconstruction of the shadow intensity from the moments through Algorithm 1. Note that b_0 still does not need to be stored because it corresponds to the total alpha of all surfaces blended together and due to $\alpha_{n_s-1} = 1$ we know $b_0 = 1$.

Since the over operator is required, translucent occluders have to be rendered to the moment shadow map directly rather than generating the entire moment shadow map from a depth buffer. Besides, we need to work around a limitation of current graphics APIs. The opacity value used for blending cannot be independent from the values written to RGBA textures. Hence, we use two RG textures, each with 16 bits per channel, instead of a single RGBA texture. Rendering is done using hardware support for multiple render targets, so performance is only mildly reduced. Of course, it is still beneficial to use the sparse quantization transform. More details on our implementation are provided in Appendix D.3.

3.3. Results and Discussion

While we have formulated the approach above for moment shadow maps, it is applicable to any kind of filterable shadow map and related methods utilize that [Delalandre et al. 2011; McGuire and Mara 2016]. Figure 5 compares results obtained with different filterable shadow maps. All shown techniques underestimate the shadow intensity, so darker results are necessarily closer to the ground truth. We observe that moment shadow mapping yields the darkest self-shadowing in the smoke and the least light leaking on the pipes (Figure 5d). Overall it performs best, although the run time increase in comparison to 64-bit exponential variance shadow maps is a bit higher than usual due to the high shading rate. Using 128-bit moment shadow maps is not beneficial here because the negative effect of the biasing is less significant when depth distributions are complex in the first place.



(a) *Variance shadow mapping, 32 bits, 2.6 ms/2.5 ms*

(b) *Exponential variance shadow mapping, 64 bits, $c^{+}_{\text{evsm}} = c^{-}_{\text{evsm}} = 5.54$, 3.2 ms/2.9 ms*

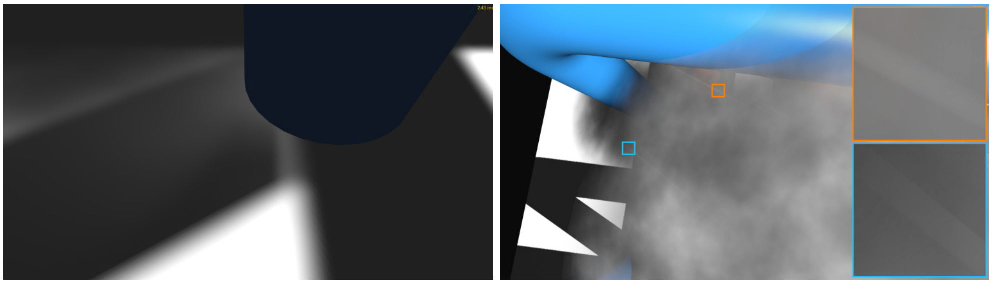
(c) *Exponential variance shadow mapping, 128 bits, $c^{+}_{\text{evsm}} = 40$, $c^{-}_{\text{evsm}} = 5.54$, 4.4 ms/3.5 ms*

(d) *Moment shadow mapping, 64 bits, $\alpha_b = 6 \cdot 10^{-5}$, 3.7 ms/3.3 ms*

Figure 5. A scene with walls, colored pipes and smoke consisting of 30 textured planes. Various filterable shadow maps are used to compute the shadows. Results exhibit different amounts of self-shadowing within the smoke and partial shadow of the smoke on the opaque surfaces. The shadow map resolution is 1024^2 and images are rendered at 3840×2160 with $4\times$ multisample antialiasing. Timings are full frame times when rendering to the filterable shadow map with/without alpha blending on an NVIDIA GeForce GTX 970.

64-bit exponential variance shadow maps yield slightly weaker self-shadowing in the smoke and there is strong light leaking at the boundary of the pipe (Figure 5b). The higher exponent in 128-bit exponential variance shadow mapping actually makes both artifacts worse (Figure 5c). With variance shadow mapping, shadows cast by the smoke are reconstructed almost as well as with 64-bit moment shadow mapping but there is unacceptable light leaking on opaque surfaces (Figure 5a).

Figure 6 demonstrates artifacts encountered with moment shadow mapping for translucent occluders. The many layers of the smoke in our test scene add to the complexity of depth distributions and thus light leaking on the surfaces increases (Figure 6a). 64-bit exponential variance shadow maps exhibit very similar artifacts. Dividing shadow intensities by 95% resolves the issue in this example. More complex depth distributions degrade the approximation quality at all depths. Therefore, the silhouette of the blue pipe in the background leads to increased light leaking along



(a) *Light leaking on surfaces* **(b)** *Volumetric light leaking*

Figure 6. Screenshots of two typical artifacts of shadows for translucent occluders rendered with 64-bit moment shadow mapping. The complex depth distributions increase light leaking on surfaces and in the volume. To make the artifacts more visible, shadow intensities are not scaled and the image on the right does not use a two-pass Gaussian for shadow filtering.

elongated stripes for the self-shadows of the smoke in the foreground in Figure 6b.

Note that rounding errors may accumulate through alpha blending. In some experiments with 64-bit moment shadow maps we observed corresponding artifacts. Accumulation of rounding errors is particularly strong when there are many overlapping layers with a very low opacity. We found that a simple alpha test discarding fragments with an opacity below 1% removed these artifacts. If an alpha test is not an option, one may use 128-bit moment shadow maps or a method for transparency other than alpha blending.

In spite of these artifacts, we believe that the technique is robust enough for use in production. It is particularly attractive due to its simple implementation and its low overhead (e.g., 0.4 ms for the scene in Figure 5d). The combination with stochastic shadow maps [Enderton et al. 2010] seems compelling for situations where sorting the translucent geometry is not practical. The technique can also be extended to colored translucent occluders by using one moment shadow map per color channel.

4. Moment Soft Shadow Mapping

So far, all shown results use a directional light, i.e., a point light at infinite distance. In reality, light sources always have some area and partial occlusion leads to soft shadows. Filtered hard shadows resemble soft shadows but they do not harden at contact points. In the present section, we extend moment shadow mapping to use an adaptive kernel size. This way, we obtain a highly efficient approximation to soft shadows.

4.1. Related Work

Like most other techniques that are suitable for performance-sensitive real-time applications, our approach uses the framework of percentage-closer soft shadows [Fernando 2005]. This technique uses a common shadow map. For each fragment, it searches a neighborhood in the shadow map for relevant blockers and computes their average depth. Then an adequate filter size is computed. For directional lights this filter size is simply proportional to the distance between receiver and caster. Percentage-closer filtering computes the filtered shadow intensity. Percentage-closer soft shadowing generates plausible results with few noticeable artifacts but the cost is high due to excessive sampling.

Summed-area variance shadow maps [Lauritzen 2007] try to avoid the sampling by means of a summed-area table. A summed-area table [Crow 1984] is a prefiltered representation of a texture where each texel stores the integral over a rectangle from the left top to its location. The integral over an arbitrary rectangle is queried by sampling the summed-area table at the four corners of this rectangle (Figure 7a). A summed-area table of a variance shadow map enables filtering with an arbitrary filter size in constant time but the blocker search still requires sampling. Filtering is done using a box kernel which corresponds to a rectangular area light.

Variance soft shadow mapping [Yang et al. 2010] accelerates the blocker search using a heuristic based on a single query to a summed-area variance shadow map. Difficult situations are handled in more expensive ways. Convolution soft shadow mapping [Annen et al. 2008a] uses either a summed-area table or mipmaps to filter based on convolution shadow maps. The blocker search uses a second set of filterable textures. Exponential soft shadow mapping [Shen et al. 2013] uses summed-area tables over smaller regions of an exponential shadow map to avoid unacceptable precision loss. Again additional textures are needed for the blocker search. The authors use kernel subdivision to better approximate Gaussian filter kernels.

4.2. Summed-Area Tables with Four Moments

Our technique follows the same basic steps as variance soft shadow mapping but never requires more than two queries to the summed-area table. We generate a summed-area table of a moment shadow map, use it to estimate average blocker depth during the blocker search, estimate the appropriate filter size and use the summed-area table to perform the filtering. Our discussion begins with the generation of the summed-area table.

The summed-area table is created in two passes. The first one creates horizontal prefix sums and the second one creates vertical prefix sums on the output of the first pass. Both passes are implemented in a compute shader using one thread per row/column as recommended by Klehm et al. [2014].

For small variance shadow maps the precision provided by summed-area tables with single-precision floats may be sufficient but for moment shadow maps it is generally insufficient. We instead use 32-bit integers and modular arithmetic because this allows us to exploit prior knowledge about maximal kernel sizes [Lauritzen 2007].

Suppose that the largest used kernel covers $n_t \in \mathbb{N}$ texels (e.g., $n_t = 784$ for a 28×28 kernel). The transformed moments $\Theta_4^\star(b_1, b_2, b_3, b_4)$ (see Equation (3)) stored in the moment shadow map initially lie in $[0,\ 1]^4$. If we multiply them by $\frac{2^{32}-1}{n_t}$ and round to integers afterwards, the sum of moments in the largest relevant kernel is known to lie in $\{0, \ldots, 2^{32} - 1\}$. Thus, this number can be represented by a 32-bit unsigned integer. At the same time, we still have a precision of

$$\log_2 \frac{2^{32} - 1}{n_t},$$

which evaluates to 22.4 bits for the example above. This precision is only slightly worse than the precision of single-precision floats and we found that a moment bias of $\alpha_b = 6 \cdot 10^{-7}$ is sufficient. For larger kernels, greater values are needed.

In our implementation we generate a 128-bit moment shadow map and generate an integer summed-area table for it. During this step overflows will occur frequently but they can be safely ignored because they only subtract multiples of 2^{32}. When we query the summed-area table in a kernel containing n_t texels or fewer, we know that the result has to lie in $\{0, \ldots, 2^{32} - 1\}$ and thus computing it in integer arithmetic necessarily leads to the correct result.

Having a summed-area table, mipmapping becomes unnecessary. Therefore, the memory requirements compared to 64-bit moment shadow mapping only increase by 50%:

$$\frac{128\,\text{bits}}{64\,\text{bits} \cdot \frac{4}{3}} = \frac{6}{4} = 150\%.$$

4.3. Blocker Search

During the blocker search we perform a single look-up in the summed-area table to query four moments for the search region. We then use Algorithm 1 to turn the biased moments and the unbiased fragment depth z_f into a matching depth distribution $Z := \sum_{l=0}^{2} w_l \cdot \delta_{z_l}$ consisting of three depth values $z_0 = z_f$, $z_1, z_2 \in \mathbb{R}$ with probabilities $w_0, w_1, w_2 > 0$.

Our assumption is that this reconstruction matches up with the ground truth. If the search region contains one or two surfaces, the reconstruction is known to be nearly perfect [Peters and Klein 2015, Proposition 4]. When the search region contains three surfaces but one of them is at depth z_f, the distribution is still uniquely determined by the moments and is reconstructed correctly [Peters and Klein 2015, Proposition 8]. For this reason, it is beneficial to use the unbiased fragment depth. More complicated cases are rare.

Since this distribution is correct by assumption, we can derive the average blocker depth in analogy to percentage-closer soft shadows:

$$\frac{\sum_{l=1,\, z_l<z_f}^{2} w_l \cdot z_l}{\sum_{l=1,\, z_l<z_f}^{2} w_l}.$$

However, this formulation is not robust. The divisor is exactly the shadow intensity $Z(\mathbf{z} < z_f)$ computed for the search region. It can be arbitrarily small or even exactly zero. In this case the expression becomes meaningless. A small shadow intensity implies an unoccluded fragment. For such fragments the blocker search should return $z_0 = z_f$ to indicate that a small filter kernel is to be used.

This requirement is incorporated into the above formula robustly by setting the average blocker depth to

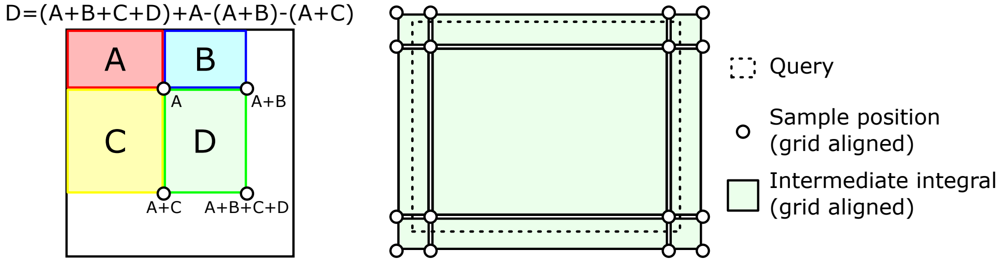
$$\frac{\varepsilon_{z_0} \cdot z_0 + \sum_{l=1,\, z_l<z_f}^{2} w_l \cdot z_l}{\varepsilon_{z_0} + \sum_{l=1,\, z_l<z_f}^{2} w_l} \tag{4}$$

where $\varepsilon_{z_0} > 0$ is a small constant. We found that this parameter is not crucial for the quality. Larger values move all shadow casters slightly towards the receivers thus making shadows harder. For small values, the average blocker depth may be too far away leading to an unnecessarily large filter kernel. However, this typically affects fully lit fragments, so the final result does not change. We use $\varepsilon_{z_0} = 10^{-3}$ in all of our experiments.

4.4. Filtering

Once the average blocker depth is available, the penumbra estimation [Fernando 2005] provides an adequate filter size. Combining it with the texture coordinate of the fragment in the shadow map, we can compute the left top and right bottom texture coordinates of the filter region. In general these will not lie in the center of texels. This necessitates interpolation for our integer summed-area tables.

Conversely to what one might expect, it is incorrect to apply bilinear interpolation directly to samples at the four corners of the filter region because the underlying values of adjacent texels differ by unknown multiples of 2^{32}. Such problems can be avoided by operating exceptionally on integrals over regions containing less than n_t texels. Figure 7b demonstrates how the filter region can be partitioned into nine such regions. For each of these regions it is safe to convert the held moments back to floating point values. Then the results from the individual regions are summed, weighting them by the area of their intersection with the filter region. This works reliably but since $4 \cdot 4 \cdot 4 \cdot 32 = 2048$ bits need to be loaded per fragment, the cost is significant (see Section 4.6.2). As an alternative we tried randomized dithering but found that the noise is too strong at hard shadow boundaries.


D=(A+B+C+D)+A-(A+B)-(A+C)
A
B
C
D
A
A+B
A+C
A+B+C+D
Query
Sample position (grid aligned)
Intermediate integral (grid aligned)

(a) *Rectangular query* **(b)** *Rectangular query with interpolation*

Figure 7. (a) Summed-area tables enable computation of the integral over a rectangle D from four samples. (b) To compute the integral over an arbitrary rectangle, we load values of 16 texels and compute the integrals over the nine shown rectangles.

Having the four filtered moments, moment shadow mapping (Algorithm 1) yields the final shadow intensity. Due to the potentially large filter size, it is important to use a sufficient depth bias. We recommend increasing it in proportion to the filter size. Additionally, an adaptive depth bias may be used [Dou et al. 2014].

4.5. Optimization

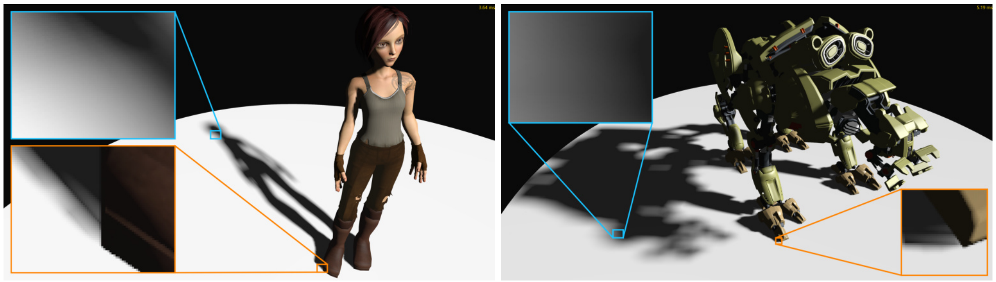
The most efficient way to optimize the technique is to reduce the number of texture loads. To avoid the cost of interpolation during the blocker search, we extend the search region to match the texel grid. Having grid-aligned search regions leads to small discontinuities in the soft shadows but is easily justified by the considerable speedup.

Another way to avoid texture loads is to skip filtering when the blocker search reveals that the fragment lies in the umbra. We already compute the shadow intensity $\sum_{l=1,\, z_l<z_f}^{2} w_l$ as part of Equation (4). If it surpasses a threshold $1-\varepsilon_u$ where $\varepsilon_u > 0$, we assume that the fragment lies in the umbra and immediately return a maximal shadow intensity. In our experiments a value of $\varepsilon_u = 0.01$, coupled with division of the shadow intensity by 99% or less, yields robust results while reducing the need for texture loads in large, connected regions.

We also tried skipping the filtering step for fully lit fragments but the lower bound provided by moment shadow mapping leads to too many false positives. It is possible to use the upper bound instead but then only few fragments are classified as fully lit. Therefore, we do not recommend this approach and do not use it in our experiments.

4.6. Results and Discussion

We compare moment soft shadow mapping against percentage-closer soft shadows and a naïve implementation of variance soft shadow mapping in a forward renderer using a single directional light. All images in this section use a shadow map resolution



(a) *Sintel* **(b)** *Quadbot*

Figure 8. Moment soft shadow mapping with a search region of 27^2 rendering shadows for two models above a plane. Note that the shadows are contact hardening.

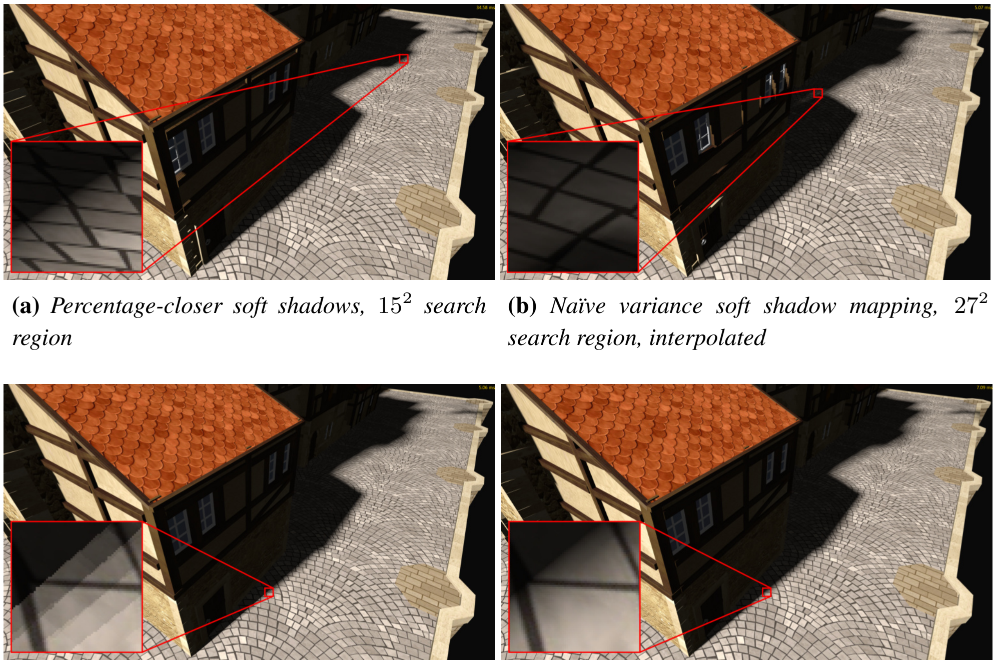
of 1024^2. For percentage-closer soft shadows we benefit from hardware-accelerated 2×2 percentage-closer filtering to take four samples at once in the filtering step. The blocker search loads all texels in the search region to avoid artifacts for fine structures. Our implementation of variance soft shadow mapping uses neither a hierarchical shadow map nor kernel subdivision. It is essentially identical to moment soft shadow mapping but with two instead of four moments. Thus, we expect it to be faster than the actual technique [Yang et al. 2010] but with more artifacts. All techniques skip filtering if the result of the blocker search allows it. For additional details on the implementation of moment soft shadow mapping we refer to Appendix D.4.

4.6.1. Qualitative Evaluation

Figure 8 shows two examples where moment soft shadow mapping produces plausible soft shadows. It works well for character models (Figure 8a) but also for complex models with many fine details (Figure 8b). As expected, shadows harden at contact-points. Note that short-range shadows exhibit slight light leaking. Since precision in the summed-area table is high, light leaking is only slightly stronger than for single-precision moment shadow maps (Figure 4f). Using depths in $[-1, 1]$ is beneficial here.

Figure 9 compares all implemented techniques for soft shadows. Percentage-closer soft shadowing generates a good result but to get an acceptable run time the search region has to be limited to 15^2 and therefore long-range shadows are too hard (Figure 9a). The other techniques support large search regions efficiently. Our naïve implementation of variance soft shadow mapping produces objectionable light leaking (Figure 9b). Note that kernel subdivision would fix this but at an increased cost. Moment soft shadow mapping without interpolation produces visible stripe patterns at hard shadow boundaries (Figure 9c). Interpolation eliminates this artifact (Figure 9d).

A failure case is shown in Figure 10. Like all techniques based on the framework of percentage-closer soft shadows, moment soft shadow mapping does not fuse oc-



(a) *Percentage-closer soft shadows,* 15^2 *search region*

(b) *Naïve variance soft shadow mapping,* 27^2 *search region, interpolated*

(c) *Moment soft shadow mapping,* 27^2 *search region, not interpolated*

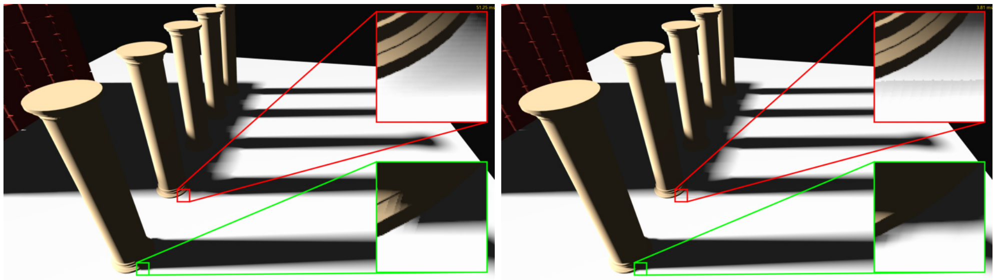
(d) *Moment soft shadow mapping,* 27^2 *search region, interpolated*

Figure 9. Various techniques for soft shadows in a scene where shadows are cast over a long range.

cluders at different depths correctly. It rather replaces them by a single occluder at the average occluder depth. Therefore, the short-range shadow of a pillar becomes soft due to the shadow of the more distant brick wall. This artifact occurs whenever the search region contains more than two occluding surfaces. Thus, it coincides with increased light leaking making the artifact more noticeable for moment soft shadow mapping. A stronger depth bias strengthens this light leaking.

The effect of an insufficient depth bias is shown in Figure 11. Lighting that is nearly parallel to a surface, coupled with large filter regions, is likely to cause wrong self-shadowing. Moment soft shadow mapping is less susceptible to this artifact than percentage-closer soft shadows but when using large search regions, it is an issue. In our implementation the depth bias is proportional to the filter size but not adaptive with respect to the surface. We believe that an adaptive depth bias will offer robust results without parameter tweaking, even for large search regions [Dou et al. 2014].

Overall, we found moment soft shadow mapping to be more robust than percentage-closer soft shadows. With percentage-closer soft shadows missing contact shadows due to a strong depth bias are hard to avoid (Figure 10a bottom). While moment soft shadow mapping does not solve this problem entirely, it does diminish it by using



(a) *Percentage-closer soft shadows* **(b)** *Moment soft shadow mapping*

Figure 10. An example of wrong occluder fusion. The hardness of the contact shadows in the two magnified insets should be the same. Both shown techniques exhibit this artifact but light leaking of moment soft shadow mapping strengthens it.

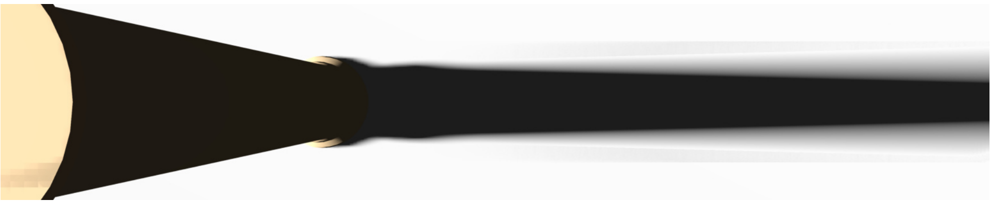


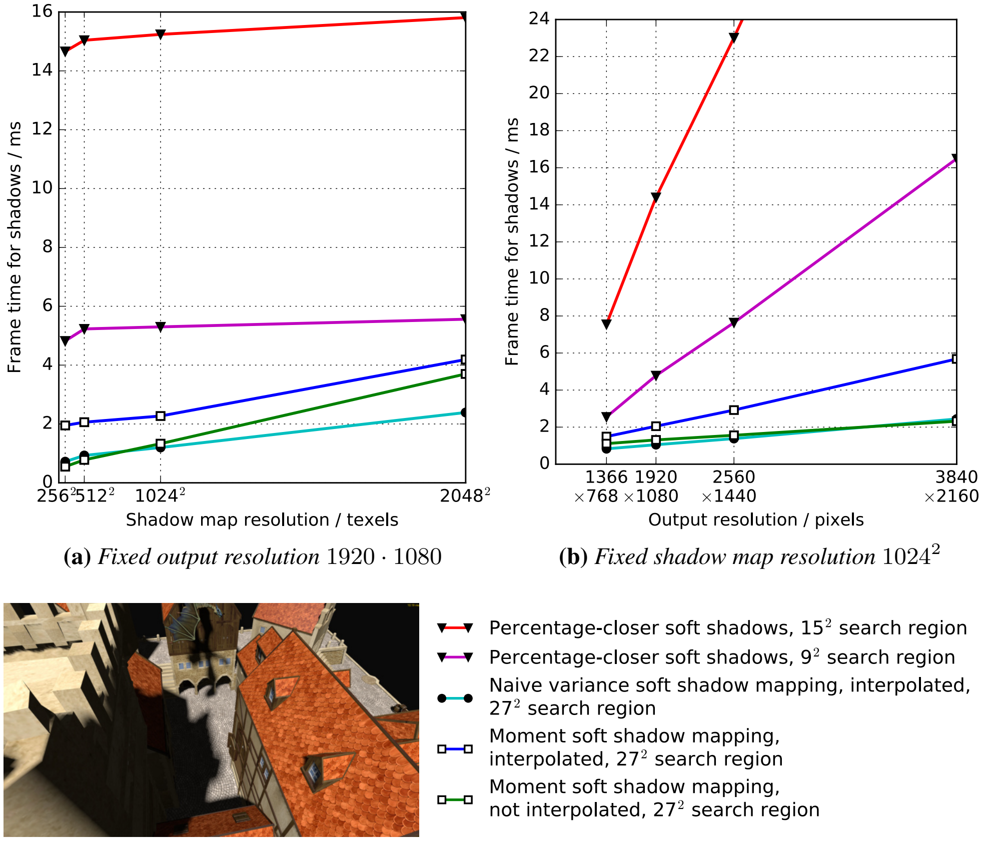
Figure 11. The shadow of a column rendered with moment soft shadow mapping using a 27^2 search region. The ground is lit at an angle of incidence of 80°. Fragments that are just outside the penumbra still use a large filter size and wrong self-shadowing occurs due to an insufficiently biased fragment depth.

lower bounds. Light leaking, which is not present in percentage-closer soft shadows, is weak thanks to the high precision in the summed-area table.

4.6.2. Run Time

Figure 12 shows run time measurements recorded in our Direct3D 11 implementation running on an NVIDIA GeForce GTX 970. The cost per texel of the shadow map increases with the memory per texel (Figure 12a). However, it is less crucial here due to the high cost per shaded fragment. Even for a 2048^2 shadow map, all techniques using filterable shadow maps are faster than percentage-closer soft shadows with a small 9^2 search region. We provide more insights on the time it takes to generate the summed-area table in Appendix C.

The cost per shaded fragment is immense for percentage-closer soft shadows, especially when using a 15^2 search region (Figure 12b). Note that such a search region is still too small to generate sufficiently soft long-range shadows (Figure 9a). Without interpolation during filtering, moment soft shadow mapping has a slightly lower cost per fragment than naïve variance soft shadow mapping with interpolation. Enabling interpolation increases this cost significantly, but moment soft shadow mapping is still



(a) *Fixed output resolution* $1920 \cdot 1080$

(b) *Fixed shadow map resolution* 1024^2

(c) *Scene*

(d) *Legend*

Figure 12. Frame times for rendering soft shadows with various techniques in the shown scene. The frame time for rendering without shadows has been subtracted. All techniques with filterable shadow maps use $4\times$ multisample antialiasing for the shadow map. The output always uses $4\times$ multisample antialiasing.

consistently faster than percentage-closer soft shadows with a 9^2 search region.

4.6.3. Conclusions

Percentage-closer soft shadows is easily the most widely used real-time technique for dynamic soft shadows of moderately large area lights. Moment soft shadow mapping is substantially faster, scales better to large search regions and large output resolutions and is less susceptible to wrong self-shadowing, which plagues percentage-closer soft shadows. The only newly introduced artifact is light leaking, which is weak due to the high precision in the summed-area table.

Therefore, we believe that it may become the new technique of choice for affordable soft shadows. A notable limitation is that the summed-area table supports exclusively rectangular light sources aligned with the shadow map. This may not be ideal, e.g., for shadows of sun light, but the smooth penumbra regions are still plausible. None the less, given a performance target to render soft shadows in 2.1 ms at 1920×1080 with a 1024^2 shadow map, this technique requires fewer compromises in quality than other methods. More natural kernels may be constructed from multiple rectangles at an increased cost.

5. Prefiltered Single Scattering with Moments

Another common simplification in rendering is to assume that all relevant light interactions happen at surfaces. This neglects volumetric scattering occurring in participating media such as smoke, dusty air, moist air and so forth. Light can be reflected in midair towards the camera. When this effect is coupled with proper computation of shadows, it leads to visible shafts of light which provide great artistic value. This effect is widely used but costly. Scattering occurs everywhere within the volume. At the same time, the visibility of the light source can change arbitrarily along a view ray. This visibility term makes the integration expensive.

In this section, we combine moment shadow mapping with prefiltered single scattering [Klehm et al. 2014]. The resulting technique evaluates single scattering using only one lookup per pixel in a prefiltered moment shadow map. We use either four (Section 5.3) or six moments (Section 5.4) and adaptively balance between lower and upper bounds (Section 5.3.1). In addition, we demonstrate how to apply filtering during the necessary resampling step (Section 5.2).

5.1. Related Work

To keep the cost reasonable, most approaches for real-time scattering disregard global illumination effects and instead focus on single scattering. Light coming directly from a light source is scattered into the view ray. Ray marching in a pixel shader [Tóth and Umenhoffer 2009] is the most immediate way to compute it. Per ray sample, a common shadow map is sampled to determine visibility. Voxelized shadow volumes [Wyman 2011] store the shadow map as boolean volume in a different coordinate system to enable 128 queries at once. A 1D min-max-mipmap may be used to traverse ray segments that are fully lit or fully shadowed in a single step [Chen et al. 2011].

5.1.1. Prefiltered Single Scattering

Prefiltered single scattering [Klehm et al. 2014] introduces the concept of filterable shadow maps to single scattering. The single scattering results for all light rays are precomputed into a convolution shadow map. While this method is fast, independent of scene complexity, the Fourier series used in convolution shadow maps introduces ringing and memory requirements are high (e.g., 256 bits per texel). Our work is an extension of prefiltered single scattering and in the following we provide enough details on this technique to make our discussion self-contained.

Besides the restriction to single scattering, prefiltered single scattering makes a few additional simplifying assumptions that we inherit. The participating media has to be homogeneous, i.e., its physical properties must be the same everywhere. Namely, these properties are the extinction coefficient σ_t defining transmittance, the phase function $f(\omega_l, \omega_p)$ which is the volumetric analog to a BRDF, and the scattering albedo ρ. We also assume homogeneous lighting from a single directional light with direction ω_l and maximal irradiance E_l. Multiple directional lights can be handled by superposition.

Now consider a surface element at distance s from the camera with outgoing radiance L_s towards the camera. The camera lies in direction ω_p at position p. Let $V(q)$ be a visibility function for the light, mapping a position in 3D-space to one if it is lit and to zero if it is shadowed. Then the radiance received at the camera is

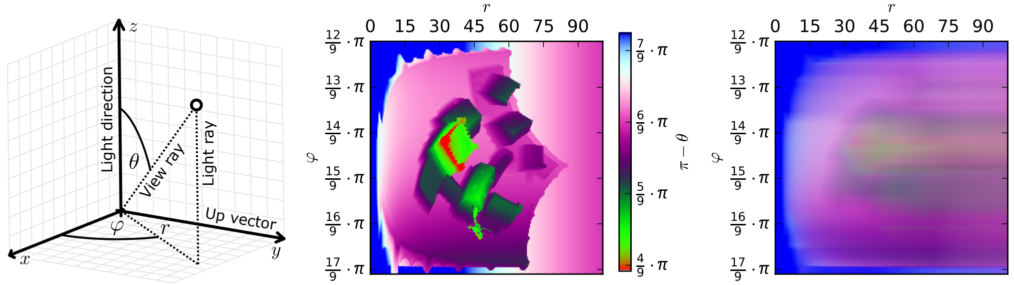
$$\exp(-\sigma_t \cdot s) \cdot L_s + f(\omega_l, \omega_p) \cdot E_l \cdot \int_0^s \exp(-\sigma_t \cdot t) \cdot V(p - t \cdot \omega_p) \, \mathrm{d}t.$$

The first summand is the radiance from the surface that remains after absorption and out-scattering. The second summand models single scattering. At each lit segment along the view ray a differential radiance of $f(\omega_l, \omega_p) \cdot E_l$ is added but only $\exp(-\sigma_t \cdot t)$ of it is transmitted to the camera.

The cost of computing single scattering lies in the integration, which includes the visibility term. When we view it as one-dimensional function along a light ray, the visibility function is a simple Heaviside step function. It is one, up to the first occluder, and then it is zero. The filterable shadow maps described in Section 2.2 provide means to store such functions in a way that enables the application of filters. We utilize them to turn integration of single scattering into an integration over rows of a shadow map.

To this end, the parameterization of the shadow map has to meet two requirements. View rays have to map to rows in the shadow map and the depth of view rays within the shadow map has to be constant. In most cases such a parameterization can be constructed as simple perspective transform [Klehm et al. 2014]. We have implemented this linear rectification but for reasons given in Section 5.2 we opted for the other proposed solution; a non-linear rectification transform applied by means of resampling [Baran et al. 2010].

To convert coordinates from world space to rectified coordinates, we first convert to light view-space and move the origin of the coordinate system into the camera location. In this space the light direction corresponds to the z-axis and the other axes are chosen arbitrarily but orthogonal as shown in Figure 13. Then the horizontal texture coordinate in the shadow map corresponds to the distance to the origin after projecting to the x-y-plane, r. This agrees with the distance between light ray and camera. For the other two coordinates, we convert to spherical coordinates. The vertical texture coordinate corresponds to the azimuth $\varphi \in (0, 2 \cdot \pi]$. Depth stored in



(a) *Rectified coordinates r, φ and θ*

(b) *Rectified moment shadow map (first three channels)*

(c) *Rectified and prefiltered moment shadow map*

Figure 13. Prefiltered single scattering resamples the shadow map into a coordinate system where each view ray maps to a row. Computation of weighted prefix sums over these rows effectively precomputes the single scattering result for every possible view ray.

the shadow map corresponds to the flipped inclination $\pi - \theta \in [0, \pi]$. This way, depth values in the new coordinate system grow monotonically with original depth values.

Since φ and θ are independent of the distance to the camera, view rays map to shadow map rows and have constant depth as required. At the same time the parameterization is valid for a shadow map because each light ray has constant r and φ and thus maps to a single texel. In terms of epipolar geometry φ indexes epipolar slices containing the light direction and going through the camera. Single scattering computations for different epipolar slices are independent. To fit the shadow map tightly, bounds for r, φ and θ are computed such that the entire view frustum is covered (see Appendix A). Along the dimension of θ we add a guard band to avoid light leaking. The length of the interval of values for θ is stretched by 10%.

We generate a filterable shadow map $b(u, v)$ in this coordinate system, indexed with integer texel indices u, v. Each texel stores a representation of the visibility function along a light ray (e.g., Fourier coefficients [Klehm et al. 2014] or moments) and filtering a row corresponds to filtering along all view rays in the corresponding epipolar slice. To precompute the relevant integrals, we need to know the world-space distance $\Delta(v)$ between successive view ray samples per slice. Since this quantity depends upon the inclination θ, a heuristic is required. Sophisticated heuristics have been proposed [Klehm et al. 2014] but we simply compute the distance for the arithmetic mean of the minimal and maximal values of θ. Then transmittance-weighted prefix sums are computed as

$$b_\Sigma(u,v) := \frac{\sum_{j=0}^{u} w_{j,v} \cdot b(j,v)}{\sum_{j=0}^{u} w_{j,v}}$$

$$w_{u,v} := \left[-\frac{1}{\sigma_t} \cdot \exp(-\sigma_t \cdot t)\right]_{\left(u-\frac{1}{2}\right)\cdot\Delta(v)}^{\left(u+\frac{1}{2}\right)\cdot\Delta(v)}.$$

To compute the scattering for a view ray ending at some location $q \in \mathbb{R}^3$, we sample the prefiltered shadow map b_Σ at the appropriate location, reconstruct a shadow intensity between zero and one as one would for filtered hard shadows, subtract it from one to get visibility and then multiply by the maximal possible scattering

$$f(\omega_p, \omega_l) \cdot E_l \cdot \left[-\frac{1}{\sigma_t} \cdot \exp(-\sigma_t \cdot t)\right]_0^{\|q-p\|_2}.$$

This procedure only requires a single lookup in the prefiltered shadow map b_Σ per pixel on screen. Thus, the run time of the technique is independent of the scene complexity.

5.2. Rectification with Filtering

The linear rectification proposed by [Klehm et al. 2014] tends to allocate major parts of the shadow map for geometry near the camera while farther geometry is compressed. This can be alleviated by moving away the near clipping plane or by using split shadow maps but neither solution is quite satisfactory. Besides, non-linear rectification still has to be implemented for the case where an epipole is near the field of view.

On the other hand, the non-linear rectification described above requires resampling of shadow maps to be implemented efficiently with rasterization hardware. Since common shadow maps cannot be filtered during resampling, this introduces considerable aliasing artifacts. Straight silhouettes exhibit staircase artifacts that lead to visible stripes in the crepuscular rays (Figure 17a). These stripes are not stable with regard to movements or rotations of the camera which makes them quite noticeable.

Ideally, the shadow map could be filtered during resampling. We have accomplished this using moment shadow maps. Instead of taking a sample without filtering from a common shadow map, we take a filtered sample from a moment shadow map. We then turn the obtained moments back into a depth distribution because we need to distort depth in a non-linear fashion. It is appropriate to expect simple distributions because we are working with small filter regions. In most relevant cases the filter region will not cover more than two different surfaces.

In Section 2.2.1 we showed how to reconstruct a depth distribution with three depth values z_0, z_1, z_2 from four moments where $z_0 = z_f$ is prescribed. This leaves us with the question how to choose z_0. To avoid an arbitrary choice and to obtain a more efficient solution we let z_0 go to infinity. As this happens, w_0 approaches zero and we can discard the depth value z_0. The remaining distribution $w_1 \cdot \delta_{z_1} + w_2 \cdot \delta_{z_2}$ is still compatible with the moments b_0, b_1, b_2, b_3. Only the fourth moment b_4 does not match. Under the assumption of two or fewer surfaces at nearly constant depth, we can be certain that the reconstruction is adequate. Otherwise it provides a reasonable approximation that is certainly better than a single shadow map sample.

Algorithm 2 Construction of a depth distribution with two depth values from three moments.
Input: A vector of moments $b \in \mathbb{R}^4$ with $b_0 = 1$ and $b_2 - b_1^2 > 0$.
Output: A distribution Z on $\mathbb{R}$ such that $\mathcal{E}_Z\left(\mathbf{z}^j\right) = b_j$ for all $j \in \{0, 1, 2, 3\}$.

```
1. Set $q_2 := b_2 - b_1^2$.
2. Set $q_1 := b_1 \cdot b_2 - b_3$.
3. Set $q_0 := -b_1 \cdot q_1 - b_2 \cdot q_2$.
4. Solve $q_2 \cdot z^2 + q_1 \cdot z + q_0 = 0$ to get solutions $z_1, z_2 \in \mathbb{R}$.
5. Set $w_2 := \frac{b_1 - z_1}{z_2 - z_1}$ and $w_1 := 1 - w_2$.
6. Return $w_1 \cdot \delta_{z_1} + w_2 \cdot \delta_{z_2}$.
```

The described distribution is constructed by Algorithm 2. It fails for inputs with non-positive variance $\sigma^2 := q_2 = b_2 - b_1^2$ but this case is adequately handled by simply returning δ_{b_1}. We provide a correctness proof in Appendix B.

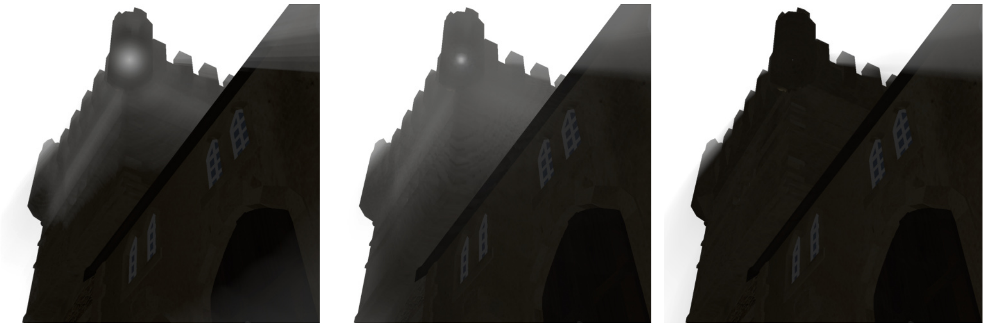
Once we have obtained the distribution, we convert its depths z_1, z_2 to inclinations θ_1, θ_2 as described in Section 5.1.1 and normalize to the interval $[-1, 1]$ clamping out-of-range values. Then we convert both values to vectors of moments via $\Theta_m^\star \circ \mathbf{b}$ and linearly combine them using the weights w_1, w_2. The result is stored in $b(u, v)$. At this point we can also generate more than four moments or other filterable shadow maps.

Using this scheme is entirely optional. Our implementation creates the rectified, filterable shadow map $b(u, v)$ using a pixel shader. When filtering is enabled, the pixel shader takes a filtered sample from a moment shadow map, otherwise it just loads a texel from a common shadow map. The sample from the moment shadow map is only slightly more expensive (see Section 5.5.2).

5.3. Prefiltered Single Scattering with Four Moments

Exchanging convolution shadow maps for moment shadow maps in prefiltered single scattering is straightforward. Instead of storing values of the Fourier basis in the shadow map, we store four moments with the usual quantization transform (see Section 2.4). When it comes to the computation of the shadow intensity during evaluation of single scattering, we can proceed as for hard shadows using Algorithm 1.

However, some assumptions made for surface shadows are inadequate for single scattering. For surface shadows we always underestimate the shadow intensity to avoid surface acne. For single scattering this is generally not necessary but we may



(a) *Convolution shadow map,* $32 \cdot 8$ *bits [Klehm et al. 2014]* **(b)** *Moment shadow map,* $4 \cdot 16$ *bits,* $\beta = \frac{1}{2}$ **(c)** *Moment shadow map,* $4 \cdot 16$ *bits,* $\beta = 0$

Figure 14. A view of the pinnacles of a tower. The directional light is hidden behind it but when single scattering is not underestimated, the light is clearly visible due to leaking artifacts for large θ.

want to avoid other artifacts. Our solution is to take a weighted combination of the sharp lower and upper bound. The upper bound is obtained by adding w_0 from Algorithm 1 to the lower bound as shown in Figure 2. Thus, the overhead for computing both bounds is small.

5.3.1. Adaptive Overestimation

Having sharp upper and lower bounds, we need a weight $\beta \in [0, 1]$ to interpolate between the two. For $\beta = 0$ single scattering is underestimated, for $\beta = 1$ it is overestimated. A simple approach would set $\beta = \frac{1}{2}$ such that the worst-case error is minimal. However, the weight can be set arbitrarily per pixel and a more sophisticated choice avoids artifacts.

Figure 14a shows an artifact of prefiltered single scattering [Klehm et al. 2014]. Light leaking is strongly increased at the epipole (i.e., when looking into the light). The inclination of the corresponding view ray is $\theta = \pi$ which corresponds to a minimal depth in the rectified shadow map. Thus, no occluder can have a smaller depth. Near the epipole, inclinations are still large and the leaking only falls off slowly.

If we use moment shadow maps with $\beta = \frac{1}{2}$, this problem persists (Figure 14b) but if we underestimate the single scattering it vanishes as expected (Figure 14c). On the other hand, a constant choice of $\beta = 0$ degrades the approximation quality elsewhere. In particular, at the antipodal point of the epipole, the inclination reaches $\theta = 0$ and no depth values in the rectified shadow map can be greater than the fragment depth. Thus, underestimation of the single scattering leads to no single scattering, which is likely incorrect.

To avoid both artifacts while maintaining the best approximation quality in intermediate cases, we set β dependent on the direction of the view ray ω_p. The weight β

is supposed to be zero for $\omega_p = -\omega_l$, one for $\omega_p = \omega_l$ and near the epipoles it should vary slowly. In our experiments, we found that the following choice reliably removes the artifacts discussed above while providing a smooth and plausible result:

$$\beta = \frac{1 + \omega_l^{\mathsf{T}} \cdot \omega_p}{2}.$$

5.4. Prefiltered Single Scattering with Six Moments

Four moment shadow mapping works well for surface shadows, because individual points rarely receive shadow from more than two different surfaces. Prefiltered single scattering on the other hand computes the average shadow received by an entire view ray. Such a view ray may be shadowed by many different surfaces at different depths. Overall, depth distributions are significantly more complex. Using only four moments for their representation is often insufficient.

Fortunately, Algorithm 1 is formulated for an arbitrary even number of moments. As a reasonable tradeoff between computational complexity and quality, we investigate the use of six moments. The robust implementation of this method is non-trivial and we now discuss the necessary steps to avoid numerical noise.

5.4.1. Computation of Roots

Solving the 4×4 linear system $B(b) \cdot q = (z^0, \ldots, z^3)^{\mathsf{T}}$ using a Cholesky decomposition still works well. Next we need to solve the cubic equation $\sum_{j=0}^{3} q_j \cdot z^j = 0$ for z. After experimenting with various iterative and closed-form solutions, we settled for a variation of a closed-form solution proposed by Blinn [2007].

The algorithm presented in the article uses two different branches for computation of the roots of minimal and maximal magnitude to avoid cancellation. In our application, we found that this overhead is unnecessary. Using one of the two branches to compute all three roots yields results that are free of artifacts. Other closed-form solutions suffered from artifacts for $|q_3| \ll 1$ and iterative solutions had a high computational overhead. Among all attempted solutions, the one based on Blinn's work is the fastest.

5.4.2. Computation of Bounds

The final step is to approximate the average visibility, which is proportional to the strength of single scattering. It is a linear combination of the weights $w_0, \ldots, w_3$, which are subject to the moment constraints

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ z_0 & z_1 & z_2 & z_3 \\ z_0^2 & z_1^2 & z_2^2 & z_3^2 \\ z_0^3 & z_1^3 & z_2^3 & z_3^3 \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

The weights in the linear combination are

$$v_0 := \beta \quad \text{and} \quad \forall l \in \{1,2,3\} : v_l := \begin{cases} 0 & \text{if } z_f > z_l, \\ 1 & \text{if } z_f \leq z_l. \end{cases}$$

Written as a dot product, the linear combination is

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}^{\mathsf{T}} \cdot \underbrace{\begin{pmatrix} 1 & z_0 & z_0^2 & z_0^3 \\ 1 & z_1 & z_1^2 & z_1^3 \\ 1 & z_2 & z_2^2 & z_2^3 \\ 1 & z_3 & z_3^2 & z_3^3 \end{pmatrix}^{-1} \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}}_{=:u}.$$

Since the matrix in this last expression is a Vandermonde matrix, the vector $u \in \mathbb{R}^4$ holds the coefficients of the interpolation polynomial $\sum_{j=0}^{3} u_j \cdot z^j$ taking value v_l for $z = z_l$ where $l \in \{0,1,2,3\}$. We construct its Newton form using divided differences and then transform back to the canonical basis of polynomials [Greenbaum and Chartier 2012, p. 181 ff.]. This works efficiently and sufficiently robust for our purposes.

5.4.3. Quantization and Biasing

In presence of complex depth distributions, even perfect accuracy in all computations cannot yield a perfect reconstruction. Therefore, we expect the effect of strong biasing to be less drastic for single scattering and using moment shadow maps with low precision is attractive.

Again, rounding errors should be diminished by means of an affine transform that is applied to the moments before storing them in the moment shadow map. As in Section 2.4 we use numerical optimization to determine it. We have experimented with general transforms and with transforms that operate on odd and even moments separately. The best found transform belongs to the latter category. It increases the entropy of the stored data by 12.5 bits per texel and is given by

$$\Theta_6^\star \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 2.5 & -1.87499864 & 1.26583039 \\ -10 & 4.20757543 & -1.47644883 \\ 8 & -1.83257679 & 0.71061660 \end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix} b_1 \\ b_3 \\ b_5 \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \\ \begin{pmatrix} 4 & 9 & -0.57759806 \\ -4 & -24 & 4.61936648 \\ 0 & 16 & -3.07953907 \end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix} b_2 \\ b_4 \\ b_6 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.018888946 \end{pmatrix} \end{pmatrix}.$$

An optimal biasing is determined as for the case with four moments (see Section 2.5). The vector of biasing moments is

$$b^\star := (1,\ 0,\ 0.5566,\ 0,\ 0.489,\ 0,\ 0.47869382)^{\mathsf{T}}.$$

For storage of the moments in our Direct3D 11 implementation we use two textures with either three channels storing 10-bit fixed-point numbers (64 bits per texel, $\alpha_b = 4 \cdot 10^{-3}$), two and four channels storing 16-bit fixed-point numbers (96 bits per texel, $\alpha_b = 8 \cdot 10^{-5}$) or three channels storing single-precision floats (192 bits per texel, $\alpha_b = 5 \cdot 10^{-6}$).

5.5. Results and Discussion

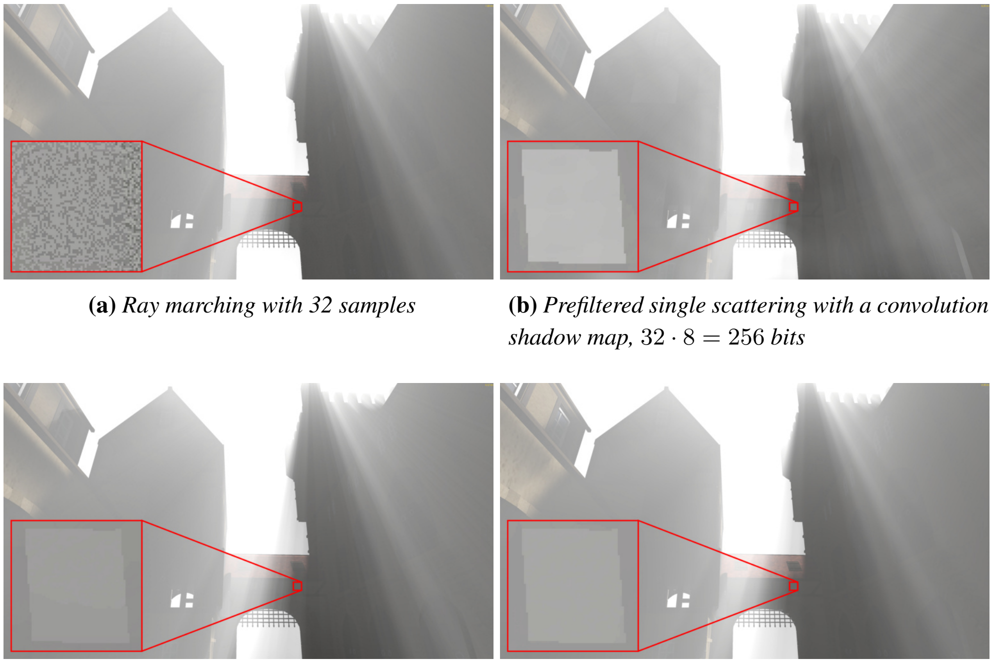
We apply all single scattering techniques in a deferred rendering pass with the depth buffer as input. Multisample antialiasing is disabled. Note that prefiltered single scattering is fast enough to be applied during forward rendering thus avoiding problems with multisampling and transparencies but we have not tested this. The details of our implementation are described in Appendix D.5. Images in this section use a shadow map resolution of 1024^2. The rectified shadow map is generated from the shadow map for surface shadows and has the same resolution.

5.5.1. Qualitative Evaluation

For comparison we have implemented ray marching with equidistant, jittered samples and prefiltered single scattering using convolution shadow maps with 32 real coefficients [Klehm et al. 2014]. The compute shader generating the transmittance-weighted prefix sums is described in detail in Appendix C. For common shadow maps we use 16-bit textures and for convolution shadow maps we use 8 bits per channel.

Figure 15 shows a comparison of these techniques. The scene mostly consists of occluders with a large area thus providing a simple case for ray marching. Therefore, noise is acceptable using 32 ray samples (Figure 15a). Techniques based on prefiltered single scattering do not produce noise but more systematic errors. When using prefiltered single scattering with convolution shadow maps (Figure 15b), ringing due to the truncation of the Fourier series is strong. Techniques based on moment shadow mapping do not exhibit ringing (Figs. 15c, 15d). An artifact that is shared by all techniques with prefiltering is magnified (Figs. 15b, 15c, 15d). A window allows a view onto the interior of the building, which is entirely shadowed. Thus, there should be no additional single scattering but approximation errors let the window appear brighter anyway. This artifact is strongest with convolution shadow maps (Figure 15b). Note that Figures 15a, 15b and 15c exhibit some surface acne from percentage-closer filtering which is not present in Figure 15d because the available moment shadow map is used for shadowing.

Figure 16 shows a more challenging test case where all techniques exhibit some characteristic artifacts. In spite of the increased number of samples, ray marching still produces strong noise (Figure 16a). Ringing in prefiltered single scattering with convolution shadow maps leads to overly dark concentric circles that change with camera movements (Figure 16b). A characteristic artifact of prefiltered single scatter-



(a) *Ray marching with 32 samples*

(b) *Prefiltered single scattering with a convolution shadow map,* $32 \cdot 8 = 256$ *bits*

(c) *Prefiltered single scattering with four moments and without filtering, 64 bits*

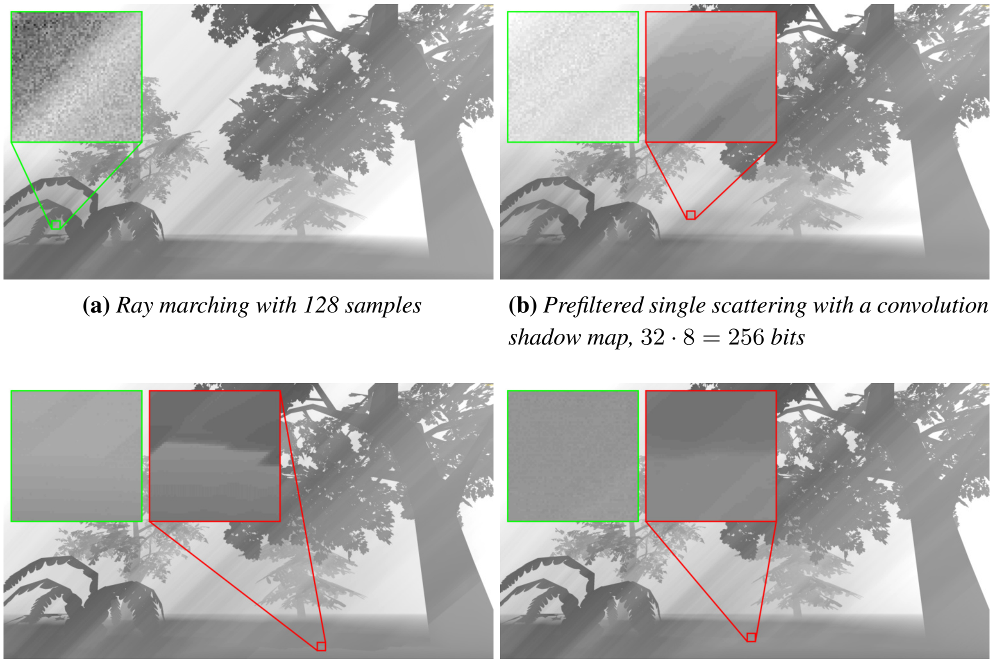
(d) *Prefiltered single scattering with six moments and filtering, 64 bits*

Figure 15. A comparison of various single scattering techniques. Note the ringing produced by convolution shadow maps and the window that appears brighter than the surrounding wall (magnified). Contrasts in the magnified insets are stretched by a factor of four.

ing with moment shadow maps are excessively sharp boundaries of shadow volumes (Figs. 16c, 16d). This occurs because the approximation quality can change quickly as depth distributions become more complex. Using six moments reduces this artifact heavily (Figure 16d). It is also slightly diminished by a greater moment bias α_b.

Figure 17 demonstrates the positive effect of filtering during resampling in an extreme case. Without filtering, crepuscular rays exhibit fine structures, which change rapidly as the camera moves or rotates (Figure 17a). Especially for slowly moving cameras this aliasing can be a very distracting artifact. Applying bilinear filtering to a moment shadow map during resampling makes the shadows lose details but aliasing is reduced to a point where it is unproblematic (Figure 17b).

Figure 18 demonstrates a case where approximation errors can be problematic. As a dragon enters the view, the single scattering is not only attenuated below but also above it. Especially for moving objects this can be quite noticeable. However, this artifact is not associated with particular locations in the scene but rather with particular view rays. If the moving object were seen in a close up the artifact would likely be much weaker because the distribution of depth values along the view ray



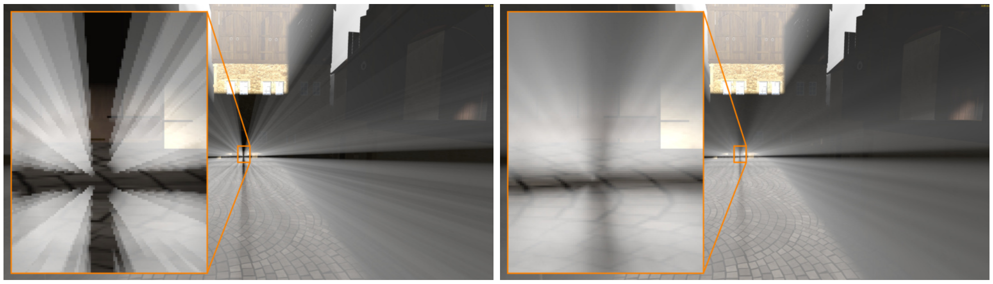
(a) *Ray marching with 128 samples*

(b) *Prefiltered single scattering with a convolution shadow map,* $32 \cdot 8 = 256$ *bits*

(c) *Prefiltered single scattering with four moments and without filtering, 64 bits*

(d) *Prefiltered single scattering with six moments and without filtering, 64 bits*

Figure 16. A challenging scenario for single scattering techniques involving shadows of trees. The scene itself is shaded black. Various artifacts are shown in magnified insets (red) next to the ray marching result (green). Contrasts in the magnified insets are stretched by a factor of four.



(a) *Not filtered*

(b) *Filtered*

Figure 17. A view into the shadow of a gate rendered using prefiltered single scattering with six moments (64 bit). Resampling a common shadow map without filtering yields heavy aliasing that is not temporally stable. Resampling a four moment shadow map with filtering (Section 5.2) produces a much smoother result.

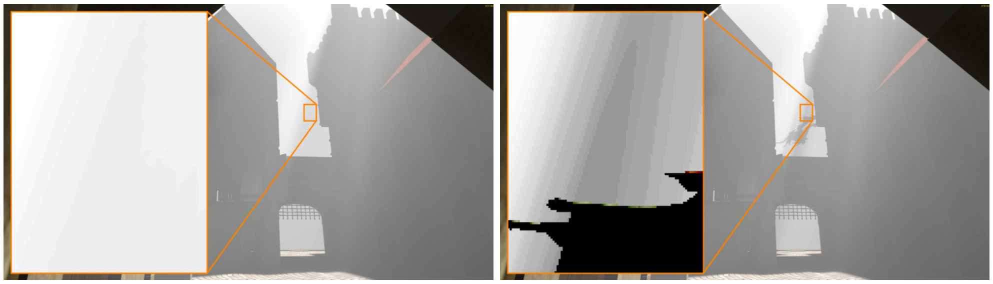
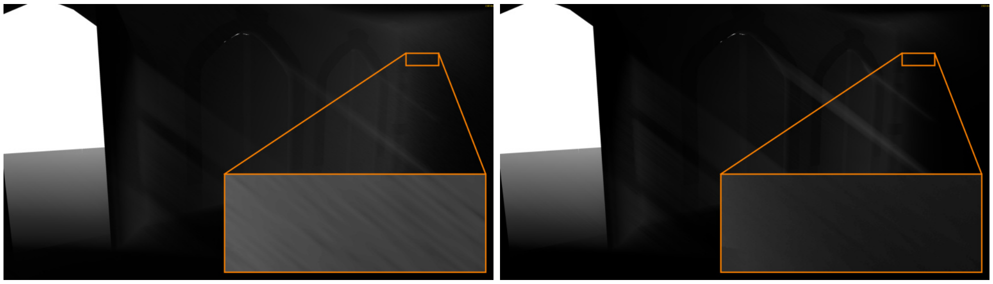


Figure 18. A scene rendered using prefiltered single scattering with six moments. As a dragon enters, crepuscular rays are not only darkened below but also above it. Magnified insets stretch contrasts by a factor of four.



(a) *Six moments in* $6 \cdot 10$ *bits,* $\alpha_b = 4 \cdot 10^{-3}$ **(b)** *Six moments in* $6 \cdot 16$ *bits,* $\alpha_b = 8 \cdot 10^{-5}$

Figure 19. An indoor scene where light leaks through walls. The scene is shaded black. Note that the use of 10 bits per moment leads to quantization noise and that the increased moment bias α_b strengthens light leaking.

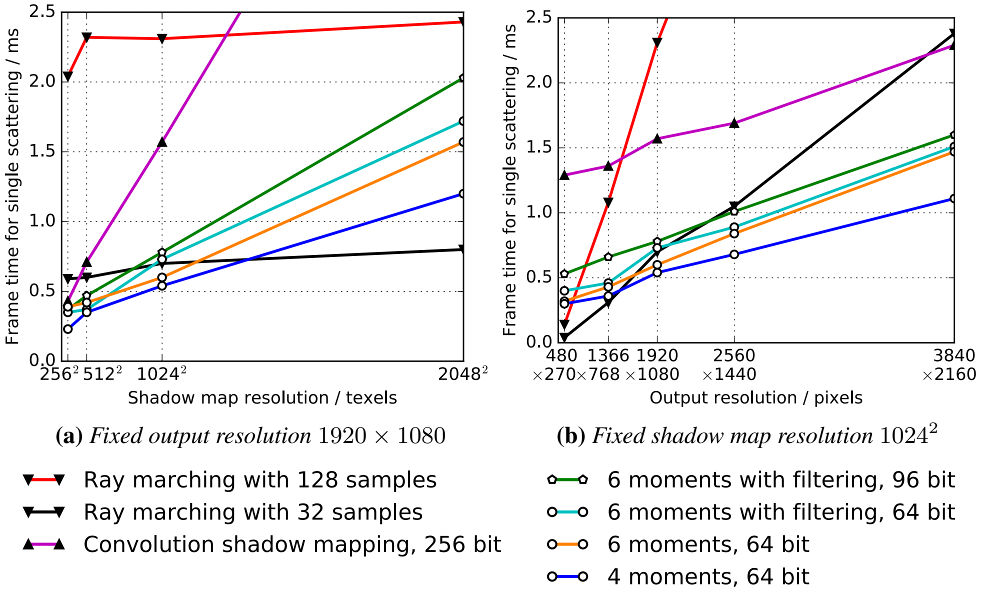
would be less complex.

Figure 19 demonstrates the slight quality improvement obtained by using 96 rather than 60 bits (plus four unused bits) for storage of six moments. The high moment bias required with 60 bits increases light leaking, which is problematic for indoor environments with thin walls. Besides, quantization errors manifest as splotches in dark regions. In some scenarios, the higher quality resulting from 96 bits may be required but in most situations the lower contrast of single scattering and the surface shading will cover up the artifacts.

5.5.2. Run Time

We measured frame times on an NVIDIA GeForce GTX 970 and show the results in Figure 20. The frame times for rendering without single scattering have been subtracted. Since the single scattering is applied in a deferred pass, this means that the timings shown are almost entirely scene-independent.

As expected, the run time of ray marching only depends weakly on the shadow map resolution because no post-processing is applied to the common shadow map



(a) *Fixed output resolution* 1920 × 1080

(b) *Fixed shadow map resolution* 1024^2

Figure 20. The contribution to the frame time due to single scattering techniques as function of shadow map resolution and output resolution. Note that shadow map generation is not included in these timings because the same (moment) shadow map is used for single scattering and surface shadows.

(Figure 20a). On the other hand, the cost of shadow map sampling increases rapidly with the output resolution, especially when using 128 samples (Figure 20b). At this number of samples, ray marching can only compete with prefiltered single scattering with four or six moments at very low resolutions such as 480 × 270. Note that these resolutions may be sufficient when a bilateral upscaling is used but undersampling artifacts have to be expected.

Prefiltered single scattering with convolution shadow maps gets more expensive rapidly with growing shadow map resolution because it stores 256 bits per texel. When using moment shadow maps, the cost per texel is much lower (Figure 20a). Using six moments stored in 64 bits is slightly more costly than using four moments stored in 64 bits, which is likely due to the use of two textures for six moments in our implementation. Filtering during resampling adds to this cost slightly and so does the use of 96 bits per texel.

Looking at the cost per pixel in the output, we observe that it is similar for all techniques using six moments but lower for prefiltered single scattering with four moments (Figure 20b). This is a strong indication that arithmetic operations are the bottleneck for techniques using six moments. Still, the cost is moderate and even at a resolution of 3840 × 2160 the techniques finish in about 1.5 ms.

5.5.3. Conclusions

Prefiltered single scattering with moment shadow maps outperforms the approach with convolution shadow maps clearly. It is faster, does not suffer from ringing and enables an explicit control over leaking artifacts by interpolating between lower and upper bounds. Compared to ray marching, the high performance at large output resolutions means that no upscaling is needed. These traits make our techniques highly attractive for performance-sensitive real-time applications.

In most cases, the variants with six moments stored in 64 bits should provide the best tradeoff between quality and run time. When single scattering is only used as a subtle effect, four moments may provide sufficient quality. If a moment shadow map for the scene is already available, the overhead for filtering is small and it should be used. Otherwise, it depends upon the scene whether the reduced aliasing justifies the cost for creation of a moment shadow map.

6. Conclusions

All presented techniques are designed to work robustly without separate branches for special cases. This is made possible by the high approximation quality of moment shadow mapping. Results may not be physically accurate but they are smooth and plausible in most situations. Moment shadow mapping itself is already being used in production by Ready at Dawn Studios and we are confident that the techniques presented here will also find their way into production.

Rendering in 4k and virtual reality requires high output resolutions. In these settings the cost of creating and filtering a moment shadow map is easily amortized and our techniques are much faster than techniques based on common shadow maps. This will make them highly attractive over the next years.

Beyond that, we are hoping to establish the theory of moments as a standard tool in graphics research and practice. They offer a generic and powerful way to store compact, filterable representations of distributions. While we have shown a variety of useful applications here, there is great potential for future work in all sorts of graphics applications. For example, there could be benefits for order-independent transparency and volumetric obscurance [Loos and Sloan 2010]. Outside rendering, recent work has applied the theory of moments to transient imaging [Peters et al. 2015].

Acknowledgments

We would like to thank Jonathan Klein, Zdravko Velinov and Jon Greenberg for their feedback and Ralf Sarlette for hardware support. We are also grateful to the anonymous reviewers for their valuable input and to the corresponding editor Carsten Dachsbacher and the editor-in-chief Marc Olano for their service to the community.

The models in Figure 8 are courtesy of the Blender foundation (see sintel.org and tearsofsteel.org). The models in Figure 12c are courtesy of Enrico Steffen and Zoltan Miklosi, the scene in Figure 16 is courtesy of Eugene Kiver.

References

Annen, T., Mertens, T., Bekaert, P., Seidel, H.-P., and Kautz, J. 2007. Convolution shadow maps. In *EGSR'07: 18th Eurographics Symposium on Rendering*, Eurographics Association, 51–60. URL: http://dx.doi.org/10.2312/EGWR/EGSR07/051-060. 20, 25, 27

Annen, T., Dong, Z., Mertens, T., Bekaert, P., Seidel, H.-P., and Kautz, J. 2008. Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph. (Proc. SIGGRAPH 2008) 27*, 3 (Aug.), 34:1–34:8. URL: http://dx.doi.org/10.1145/1360612.1360633. 18, 31

Annen, T., Mertens, T., Seidel, H.-P., Flerackers, E., and Kautz, J. 2008. Exponential shadow maps. In *GI'08: Proceedings of Graphics Interface 2008*, Canadian Information Processing Society, 155–161. URL: http://dx.doi.org/10.20380/GI2008.20. 20

Baran, I., Chen, J., Ragan-Kelley, J., Durand, F., and Lehtinen, J. 2010. A hierarchical volumetric shadow algorithm for single scattering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2010) 29*, 6 (Dec.), 178:1–178:10. URL: http://dx.doi.org/10.1145/1882261.1866200. 40

Blinn, J. F. 2007. How to solve a cubic equation, part 5: Back to numerics. *IEEE Computer Graphics and Applications 27*, 3 (May), 78–89. URL: http://dx.doi.org/10.1109/MCG.2007.60. 45

Chen, J., Baran, I., Durand, F., and Jarosz, W. 2011. Real-time volumetric shadows using 1D min-max mipmaps. In *I3D'11: Proceedings of the 15th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 39–46. URL: http://dx.doi.org/10.1145/1944745.1944752. 39

Crow, F. C. 1984. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '84, 207–212. URL: http://dx.doi.org/10.1145/800031.808600. 18, 31

Delalandre, C., Gautron, P., Marvie, J.-E., and François, G. 2011. Transmittance function mapping. In *I3D'11: Proceedings of the 15th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 31–38. URL: http://dx.doi.org/10.1145/1944745.1944751. 18, 27, 28

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *I3D'06: Proceedings of the 2006 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 161–165. URL: `http://dx.doi.org/10.1145/1111411.1111440`. 17, 19

DOU, H., YAN, Y., KERZNER, E., DAI, Z., AND WYMAN, C. 2014. Adaptive depth bias for shadow maps. *Journal of Computer Graphics Techniques (JCGT) 3*, 4 (December), 146–162. URL: `http://jcgt.org/published/0003/04/08/`. 34, 36

ENDERTON, E., SINTORN, E., SHIRLEY, P., AND LUEBKE, D. 2010. Stochastic transparency. In *I3D'10: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 157–164. URL: `http://dx.doi.org/10.1145/1730804.1730830`. 27, 30

FERNANDO, R. 2005. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, ACM. URL: `http://dx.doi.org/10.1145/1187112.1187153`. 18, 31, 33

GREENBAUM, A., AND CHARTIER, T. P. 2012. *Numerical methods – Design, analysis and computer implementation of algorithms*. Princeton University Press. 46

JANSEN, J., AND BAVOIL, L. 2010. Fourier opacity mapping. In *I3D'10: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 165–172. URL: `http://dx.doi.org/10.1145/1730804.1730831`. 27

KLEHM, O., SEIDEL, H.-P., AND EISEMANN, E. 2014. Filter-based real-time single scattering using rectified shadow maps. *Journal of Computer Graphics Techniques (JCGT) 3*, 3, 7–34. URL: `http://jcgt.org/published/0003/03/02/`. 18, 31, 39, 40, 41, 42, 44, 47, 60

KREĬN, M. G., AND NUDEL'MAN, A. A. 1977. *The Markov Moment Problem and Extremal Problems*, vol. 50 of *Translations of Mathematical Monographs*. American Mathematical Society. 20

LAURITZEN, A., AND MCCOOL, M. 2008. Layered variance shadow maps. In *GI'08: Proceedings of Graphics Interface 2008*, Canadian Information Processing Society, 139–146. URL: `http://dx.doi.org/10.20380/GI2008.18`. 20

LAURITZEN, A., SALVI, M., AND LEFOHN, A. 2011. Sample distribution shadow maps. In *I3D'11: Proceedings of the 15th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 97–102. URL: `http://dx.doi.org/10.1145/1944745.1944761`. 61

LAURITZEN, A. 2007. *GPU Gems 3*. Addison-Wesley, ch. Summed-Area Variance Shadow Maps, 157–182. URL: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch08.html. 18, 31, 32

LOOS, B. J., AND SLOAN, P.-P. 2010. Volumetric obscurance. In *I3D'10: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 151–156. URL: http://dx.doi.org/10.1145/1730804.1730829. 52

MCGUIRE, M., AND ENDERTON, E. 2011. Colored stochastic shadow maps. In *I3D'11: Proceedings of the 15th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 89–96. URL: http://dx.doi.org/10.1145/1944745.1944760. 27, 62

MCGUIRE, M., AND MARA, M. 2016. A phenomenological scattering model for order-independent transparency. In *I3D'16: Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 149–158. URL: http://dx.doi.org/10.1145/2856400.2856418. 18, 27, 28, 62

PETERS, C., AND KLEIN, R. 2015. Moment shadow mapping. In *I3D'15: Proceedings of the 19th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 7–14. URL: http://dx.doi.org/10.1145/2699276.2699277. 17, 18, 19, 20, 22, 23, 24, 25, 32, 58, 59

PETERS, C., KLEIN, J., HULLIN, M. B., AND KLEIN, R. 2015. Solving trigonometric moment problems for fast transient imaging. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2015) 34*, 6 (Nov.). URL: http://dx.doi.org/10.1145/2816795.2818103. 52

PETERS, C., MÜNSTERMANN, C., WETZSTEIN, N., AND KLEIN, R. 2016. Beyond hard shadows: Moment shadow maps for single scattering, soft shadows and translucent occluders. In *I3D'16: Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 159–170. URL: http://dx.doi.org/10.1145/2856400.2856402. 18, 25

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '87, 283–291. URL: http://dx.doi.org/10.1145/37401.37435. 19

SALVI, M., VIDIMČE, K., LAURITZEN, A., AND LEFOHN, A. 2010. Adaptive volumetric shadow maps. *Computer Graphics Forum 29*, 4, 1289–1296. URL: http://dx.doi.org/10.1111/j.1467-8659.2010.01724.x. 27

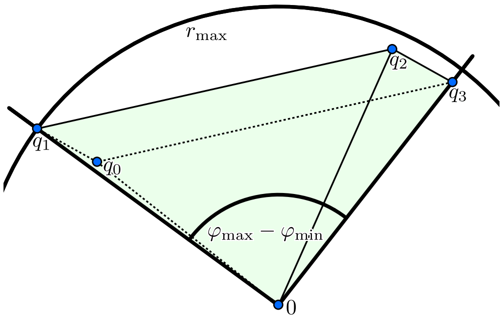
SALVI, M. 2008. *ShaderX^6*. Cengage Learning Inc., March, ch. 4.3. Rendering filtered shadows with exponential shadow maps, 257–274. 20

SHEN, L., FENG, J., AND YANG, B. 2013. Exponential soft shadow mapping. *Computer Graphics Forum 32*, 4. URL: `http://dx.doi.org/10.1111/cgf.12156`. 18, 31

TÓTH, B., AND UMENHOFFER, T. 2009. Real-time volumetric lighting in participating media. In *Eurographics 2009 - Short Papers*, The Eurographics Association. URL: `http://dx.doi.org/10.2312/egs.20091048`. 39

WYMAN, C. 2011. Voxelized shadow volumes. In *HPG'11: Proceedings of the 3rd Conference on High-Performance Graphics*, ACM, 33–40. URL: `http://dx.doi.org/10.1145/2018323.2018329`. 39

YANG, B., DONG, Z., FENG, J., SEIDEL, H.-P., AND KAUTZ, J. 2010. Variance soft shadow mapping. *Computer Graphics Forum 29*, 7. URL: `http://dx.doi.org/10.1111/j.1467-8659.2010.01800.x`. 18, 31, 35


$r_{\max}$
q_2
q_3
q_1
q_0
$\varphi_{\max} - \varphi_{\min}$
0
$r_{\max}$
q_1
q_0
$\varphi_{\max} - \varphi_{\min}$
0
q_2
q_3

(a) *(Flipped) light direction outside frustum* **(b)** *Flipped light direction in frustum*

Figure 21. The view frustum seen from the perspective of a directional light. By considering the far plane vertices q_0, q_1, q_2, q_3, we can compute sharp bounds for r and φ. Computation of bounds for θ is not shown.

A. Bounds for Rectified Coordinates

In Section 5.1.1 we state that we compute sharp bounds for r, φ and θ such that the entire view frustum is covered. This appendix provides details on this procedure. A reference implementation is provided in the supplemental materials.

Single scattering should be accumulated over the entire view ray. Thus, we do not consider the near clipping plane. Let $q_0, q_1, q_2, q_3 \in \mathbb{R}^3$ be the coordinates of the four vertices of the far clipping plane of the camera used for main scene rendering in light view space. Without loss of generality let the camera position be in the origin of the coordinate system as shown in Figure 21. Then the maximal value for r is given by

$$r_{\max} := \max_{j \in \{0,\ldots,3\}} \sqrt{(q_j)_0^2 + (q_j)_1^2}.$$

Note that $(q_j)_k$ denotes the k-th entry of the vector q_j for $k \in \{0, 1, 2\}$. Since we ignore the near plane, $r_{\min} := 0$.

To determine $\varphi_{\min}$ and $\varphi_{\max}$, we compute the maximal pairwise angle enclosed by the vectors $((q_j)_0, (q_j)_1)^\mathsf{T} \in \mathbb{R}^2$ for $j \in \{0, 1, 2, 3\}$ (Figure 21a). The azimuth of one of the two involved vectors provides $\varphi_{\min}$, the other provides $\varphi_{\max}$. A special case arises if the light direction or flipped light direction lies within the view frustum. This can be checked using the four side clipping planes. In both cases we have to set $\varphi_{\min} = 0$ and $\varphi_{\max} = 2 \cdot \pi$ to indicate that the boundary of the far clipping plane surrounds the camera position in the shadow map (Figure 21b).

The computation of $\theta_{\min}$ and $\theta_{\max}$ is more intricate because the extremal inclination may be realized at the vertices, on the edges, or within the area of the far clipping plane. It is convenient to exploit that inclinations depend monotonically on the z-coordinate of normalized vectors, i.e.,

$$\theta_j := \arccos \frac{(q_j)_2}{\|q_j\|_2}.$$

Taking the minimal and maximal values of $\theta_0, \dots, \theta_3$ yields the extrema at vertices. The inclination is extremal within the area of the far clipping plane if and only if the light direction or flipped light direction lie within the view frustum (see above). In this case an extremal inclination is $\theta_{\mathrm{min}} = 0$ or $\theta_{\mathrm{max}} = \pi$, respectively.

To find extrema on an edge of the far plane connecting corner points $j, k \in \{0, \dots, 3\}$, we take the derivative of the z-coordinate of normalized vectors on the edge to find critical points:

$$\frac{\partial}{\partial t} \frac{(q_j + t \cdot (q_k - q_j))_2}{\|q_j + t \cdot (q_k - q_j)\|_2} = 0.$$

This equation has the unique solution

$$t = \frac{(q_j)_2 \cdot q_j^{\mathsf{T}} \cdot (q_k - q_j) - (q_k - q_j)_2 \cdot \|q_j\|_2^2}{(q_k - q_j)_2 \cdot q_j^{\mathsf{T}} \cdot (q_k - q_j) - (q_j)_2 \cdot \|q_k - q_j\|_2^2}.$$

If $t \in [0, 1]$, we may need to adapt $[\theta_{\mathrm{min}}, \theta_{\mathrm{max}}]$ to include the inclination at this point on the ray.

Note that this whole algorithm only needs to be executed once per frame to get single scattering for one directional light.

B. Reconstruction from Three Moments

Our method for filtering during resampling relies on Algorithm 2 to turn three moments into a depth distribution with two depth values. We now provide a correctness proof for this algorithm.

Proposition 1. *Given a valid input, Algorithm 2 works correctly.*

Proof. The algorithm reconstructs a depth distribution using exactly two depth values. Such depth distributions are known to correspond to a singular Hankel matrix $B(b)$ as defined in Algorithm 1 [Peters and Klein 2015, Proposition 4]. Implicitly, the algorithm computes the missing fourth moment b_4 such that the Hankel matrix is singular.

For all $b_4 \in \mathbb{R}$ the multilinearity of the determinant implies (remembering that $b_0 = 1$)

$$\begin{aligned}\det B\begin{pmatrix} b \\ b_4 \end{pmatrix} &= \det \begin{pmatrix} b_0 & b_1 & b_2 \\ b_1 & b_2 & b_3 \\ b_2 & b_3 & b_4 \end{pmatrix} = b_4 \cdot \det \begin{pmatrix} b_0 & b_1 & 0 \\ b_1 & b_2 & 0 \\ b_2 & b_3 & 1 \end{pmatrix} + \det B \begin{pmatrix} b \\ 0 \end{pmatrix} \\ &= b_4 \cdot (b_2 - b_1^2) + \det B \begin{pmatrix} b \\ 0 \end{pmatrix}.\end{aligned}$$

Thus, we can choose the unique $b_4 \in \mathbb{R}$ that makes the Hankel matrix singular. Then there is a unique matching depth distribution using at most two depth values [Peters and Klein 2015, Proposition 4]. Furthermore, if $q \in \mathbb{R}^3$ is a vector in the kernel of $B\begin{pmatrix} b \\ b_4 \end{pmatrix}$, these depth values are the roots of the polynomial $\sum_{j=0}^{2} q_j \cdot z^j$ [Peters and Klein 2015, supplementary, p. 4].

To prove that $q := (q_0, q_1, q_2)^\mathsf{T}$ lies in the kernel, we observe that the first two rows of the Hankel matrix are linearly independent due to

$$\det \begin{pmatrix} b_0 & b_1 \\ b_1 & b_2 \end{pmatrix} = b_2 - b_1^2 > 0.$$

Thus, the third row is a linear combination of the first two rows and it suffices to show that q is orthogonal to the first two rows:

$$\begin{aligned} (1, b_1, b_2) \cdot q &= (-b_1 \cdot q_1 - b_2 \cdot q_2) + b_1 \cdot q_1 + b_2 \cdot q_2 = 0, \\ (b_1, b_2, b_3) \cdot q &= (b_2 - b_1^2) \cdot q_1 + (b_3 - b_1 \cdot b_2) \cdot q_2 = q_2 \cdot q_1 - q_1 \cdot q_2 = 0. \end{aligned}$$

Therefore, Algorithm 2 computes the two depth values correctly. Note that there must be two distinct roots. Otherwise there would be a matching depth distribution with a single depth value z_1 and the Hankel matrix would be the rank-one matrix

$$\begin{pmatrix} 1 & z_1 & z_1^2 \end{pmatrix}^\mathsf{T} \cdot \begin{pmatrix} 1 & z_1 & z_1^2 \end{pmatrix} \in \mathbb{R}^{3\times 3}.$$

The remainder of the algorithm solves the system of linear equations

$$\mathcal{E}_{w_1 \cdot \delta_{z_1} + w_2 \cdot \delta_{z_2}} \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} 1 & 1 \\ z_1 & z_2 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ b_1 \end{pmatrix}$$

which uniquely determines the correct weights. □

C. Generating Transmittance-Weighted Prefix Sums

In terms of arithmetic, generation of transmittance-weighted prefix sums is an inexpensive operation. We expect the corresponding compute shader to be bandwidth limited. We benchmark our implementation on an NVIDIA GeForce GTX 970 with a bandwidth of $196\ \mathrm{GB/s}$. Thus, computation of prefix sums over a 1024^2 texture with 64 bits per texel should ideally take

$$\frac{2 \cdot 1024^2 \cdot 8\,\mathrm{B}}{196\,\mathrm{GB/s}} = 85.6\,\mu\mathrm{s}.$$

For a texture with 128 bits per texel we expect it to take $171.2\,\mu$s.

We use GPU timings obtained with NVIDIA Nsight to compare this expectation to the actual run times. For the 128-bit textures used in moment soft shadow mapping, the simple scheme using one thread per row or per column [Klehm et al. 2014] is clearly bandwidth limited with a run time around $190\,\mu\mathrm{s}$. However, the same approach is less efficient for textures with 64 bits per texel. Our initial implementation for 64-bit shadow maps with six moments took $620\,\mu\mathrm{s}$.

Our optimized implementation can process a 64-bit moment shadow map with four moments in $110\,\mu\mathrm{s}$ which is close to the theoretical optimum of $85.6\,\mu\mathrm{s}$. A 64-bit moment shadow map with six moments takes $180\,\mu\mathrm{s}$ so it is not quite optimal but still only twice as expensive as the theoretical optimum and we were unable to optimize it further.

This is accomplished using thread groups of 8×8 threads. For a texture of height $n_y \in \mathbb{N}$ we spawn $\frac{n_y}{8}$ such thread groups, such that eight threads run per row. At any point in time each thread group operates on one 8×8 block in the texture, going through from left to right. Each thread will independently compute all prefix sums for its row. However, it will only write out the one prefix sum that corresponds to its location in the block.

This means that the overall amount of texture reads and additions is multiplied by a factor of eight but so is parallelism. Besides the writes to the output texture are coalesced. We tried caching the texture reads into thread-group-shared memory but the standard texture caches turned out to be more efficient. If the shadow map consists of multiple textures, they are processed in parallel by separate thread groups.

Our HLSL implementation of this approach for various texture formats is provided in the supplemental materials. Of course, other GPUs may exhibit different behavior and we do not recommend using our code in production without performing benchmarks on the targeted hardware.

D. Implementation Guide

The supplemental materials include documented shader code written in HLSL. The building blocks in this code are highly modular to enable the many combinations of techniques supported by the demo. This section provides details on how to combine the provided functions to implement the techniques described in this paper. We generally describe the fastest approach that we know. In many cases less intricate implementations are possible at an increased cost.

D.1. Generation of Moment Shadow Maps

In almost all discussed techniques, the first step is to generate a moment shadow map for the opaque surfaces in the scene.

Generation of a multisampled shadow map First, set up the view and projection matrices for the shadow map. The best way to do this depends strongly on the application, but whatever works well for common shadow maps works for moment shadow maps. Using multiple shadow maps for techniques such as sample distribution shadow maps [Lauritzen et al. 2011] is possible. For each shadow map, create a multisampled depth buffer. We recommend a 16-bit fixed-point format. Then render all shadow casters to this depth buffer. No render target is bound for this step, and usually, a pixel shader is not needed either.

Conversion to a moment shadow map Now, initialize a moment shadow map of matching resolution for each shadow map. The texture must have four channels, each using either unsigned, normalized 16-bit integers, or single-precision floats (see Figure 4). It is not multisampled and need not have mipmaps. The depth buffer is converted to a moment shadow map during a custom resolve in a pixel shader. For each texel of the moment shadow map, load each sample from the depth buffer. Then convert the read depth to a linear depth $z \in [-1, 1]$ and compute a vector of moments $(z, z^2, z^3, z^4)^\mathsf{T}$. The results for all samples are averaged. Other filters are possible as long as the weights are non-negative. For 64 bits per texel, apply the quantization transform in Equation (3). The transformed moments are the output of the pixel shader. Relevant shader functions in *Shaders/Shadow.fx* are:

- `ComputeFragmentDepth()`,
- `ComputeMomentVector4Moments_float4()`,
- `ComputeMomentVector4MomentsOptimized_float4()`.

D.2. Rendering Filtered Hard Shadows

In most cases a moment shadow map with 64 bits per texel is appropriate for filtered hard shadows.

Filtering Some amount of filtering has already been done through the custom resolve, but more is probably necessary to diminish aliasing. We recommend application of a two-pass Gaussian filter to the moment shadow maps. Two pixel shader passes first blur horizontally and then vertically, though in some cases compute shaders may be faster. After filtering, generate a full mipmap hierarchy for the moment shadow maps, likely using built-in functionality of the graphics API. Relevant shader functions in *Shaders/Shadow.fx* are:

- `ApplyGaussianFilterHorizontal5_float4()`,
- `ApplyGaussianFilterVertical5_float4()`.

Retrieving moments At this point, everything is ready for shading fragments during forward or deferred rendering. First, compute the appropriate texture coordinate for a lookup in the moment shadow map. This works the same as for common shadow maps. At this point, also compute the fragment depth. Then take a single filtered sample from the appropriate moment shadow map. All hardware-accelerated texture filtering is applicable, e.g., bilinear filtering, mipmapping and anisotropic filtering. Revert any quantization transform used during generation of the moment shadow map, then apply the appropriate biasing scheme. Relevant shader functions in *Shaders/Shadow.fx* are:

- `ComputeShadowMapCoordinate()`,
- `Sample4MomentShadowMap()`,
- `Sample4MomentOptimizedShadowMap()`.

Computing the shadow Finally, use Algorithm 1 to evaluate the shadow intensity at the biased fragment depth. For over darkening against light leaking, divide by a constant like 98% and clamp back to $[0, 1]$. Subtract the result from one and multiply into the irradiance of the light source. Relevant shader functions in *Shaders/Shadow.fx* are:

- `Compute4MomentUnboundedShadowIntensity()`,
- `ScaleShadowIntensity()`.

D.3. Handling Translucent Occluders

For translucent occluders in the moment shadow map, still generate a moment shadow map for opaque occluders first. It may be necessary to split up the four channels across two textures with two channels to allow alpha blending.

Blending in translucent occluders There are various ways to achieve this because any method for order-independent transparency may be used. We use sorted geometry and the over operator but stochastic transparency looks promising as well [McGuire and Enderton 2011; McGuire and Mara 2016]. In any case take the depth of the translucent geometry and convert it to a vector of moments as in Section D.1. For alpha blending, use additive blending and multiply the source by α and the destination by $1 - \alpha$.

Moment shadow maps with translucent occluders are used in the same way as common moment shadow maps.

D.4. Moment Soft Shadow Mapping

The first step for moment soft shadow mapping is to generate a moment shadow map with 128 bits per texel using the optimized quantization transform. It is easiest to use single-precision floats at this point. Note that only axis-aligned rectangular filters are possible, and therefore it may be necessary to adapt the shadow map parameterization.

Generation of a summed-area table The summed-area table is a four-channel texture without multisampling or mipmaps storing 32-bit unsigned integers. First, determine the maximal number of texels in the search region n_t. Then use a compute shader with one thread per row to generate horizontal, integer prefix sums over the moment shadow map. The floating point moments are converted to integers by multiplying by $\frac{2^{32}-1}{n_t}$ and rounding. The second pass operates on the output of the first pass generating vertical prefix sums with one thread per column. Relevant shader functions in *Shaders/Shadow.fx* are:

- `ComputeFixedPrecision()`,
- `ApplyPrefixSumHorizontal_uint4()`,
- `ApplyPrefixSumVertical_uint4()`.

Blocker search Now everything is ready to shade a fragment. First, compute its coordinates in shadow map space as above. Then derive the texture coordinates bounding the search region, and query the summed-area table without interpolation to obtain the corresponding moments. The result is used to estimate the average blocker depth. If the blocker search reveals that the fragment is in the umbra, the shadow computation is complete at this point. Relevant shader functions in *Shaders/Shadow.fx* are:

- `ComputeShadowMapCoordinate()`,
- `GetBlockerSearchRectangle()`,
- `ComputeIntegerRectangleAverage_uint4()`,
- `Compute4MomentAverageBlockerDepth()`.

Penumbra estimation Penumbra estimation for moment soft shadow mapping and percentage-closer soft shadows works exactly the same. The result is used to compute the rectangular filter region for the final filtering of shadows. Relevant shader functions in *Shaders/Shadow.fx* are:

- `EstimatePenumbraSize()`,
- `GetShadowFilterRectangle()`.

Filtering Finally, query the summed-area table with interpolation to obtain four moments for the search region. At this stage, undo the quantization transform and apply weak biasing (e.g., moment bias $\alpha_b = 6 \cdot 10^{-7}$). Then use the moments to compute a shadow intensity as in Section D.2. Relevant shader functions in *Shaders/Shadow.fx* are:

- `ComputeRectangleAverage_uint4()`,
- `Convert4MomentOptimizedToCanonical()`,
- `Compute4MomentUnboundedShadowIntensity()`.

D.5. Prefiltered Single Scattering

Input to prefiltered single scattering may be a common shadow map or a moment shadow map dependent on whether filtering is desired during resampling. This can be the same (moment) shadow map used for surface shadows and can use arbitrary parameterizations.

We only discuss prefiltered single scattering with six moments here, but the other variants are implemented similarly.

Preparation of rectification transforms Prefiltered single scattering transforms the (moment) shadow map into a very specific coordinate system based on epipolar geometry. Some quantities for this coordinate conversion should be prepared on the CPU. HLSL reference implementations for the functions that need to be implemented on the CPU are provided in *Shaders/ParticipatingMedia.fx*:

- `ComputeRectificationToWorldSpaceDirectional()`,
- `GetRectifiedSpaceFrustumBounds()`.

Resampling Generate a six moment shadow map during resampling. It consists of two textures without mipmapping and multisampling using 10-bit unsigned, normalized integers for red, green and blue. The alpha channel is unused. This is created in a pixel shader pass using multiple render targets. When not using filtering, sample the common shadow map at the appropriate coordinate, convert the depth to moments, apply a quantization transform and output the result. When using filtering, take a filtered sample from the appropriate moment shadow map, represent the moments by a depth distribution with two depth values using Algorithm 2, convert both depths to moments, combine them and apply the quantization transform. Relevant shader functions in *Shaders/ParticipatingMedia.fx* are:

- `GetRectifiedDepth()`,
- `GetSparseRectifiedRepresentation()`,

- `GetMomentsFromDepth3_3()`,
- `GetMomentsFromSparseRectifiedRepresentation3_3()`.

Generation of prefix sums Next compute transmittance-weighted prefix sums for the output of the previous step. The output texture has the same format as the input texture except that it should have mipmaps. The prefix sums are generated through the compute shader discussed in Appendix C. Once they are created, generate the mipmap hierarchy. Relevant shader functions in *Shaders/ParticipatingMedia.fx* are:

- `ApplyTransmittanceWeightedPrefixSumNonLinearRectificationMSM3_3()`.

Single scattering pass Finally, apply the single scattering to the rendered scene during an additive deferred rendering pass. This pixel shader pass only requires the depth buffer and the prefiltered six moment shadow map as input. Sample the depth buffer and compute the world space position of the pixel, then convert to rectified coordinates and sample the six moment shadow map at the corresponding location. Use the moments to estimate a shadow intensity with adaptive overestimation. Finally, subtract it from one and multiply by

$$f(\omega_p, \omega_l) \cdot E_l \cdot \left[-\frac{1}{\sigma_t} \cdot \exp(-\sigma_t \cdot t) \right]_0^{\|q-p\|_2}$$

as defined in Section 5.1.1. Relevant shader functions in *Shaders/ParticipatingMedia.fx* are:

- `ComputePrefilteredSingleScatteringCoordinatesNonLinearRectification()`,
- `ComputeAdaptiveOverestimationWeight()`,
- `ComputePrefilteredSingleScattering3_3Moments()`,
- `ComputeSingleScatteringNoOcclusionDirectional()`,
- `ComputeSingleScatteringFactors()`,
- `ComputeSingleScatteringRadiance()`.

Index of Supplemental Materials

As supplemental material we provide an executable demo for Windows 7 SP1 x64 and above using Direct3D 11. While the demo itself is not open source, its documented

HLSL shader code is. It also supports HLSL shader debugging with tools such as Visual Studio or RenderDoc to ease reverse engineering.

The materials are in a single archive with a single directory. This directory contains the following important files and subdirectories:

ShadowDemo.exe The executable demo.

vc_redist.x64.exe If the demo is missing DLLs, you likely need to install the Visual Studio 2015 redistributable files by running this installer.

ShadowSettings.cfg Before running the demo you should set the appropriate resolution in this text file. It also allows you to enable support for shader debugging.

ReadMe.pdf Since the controls of the demo are not necessarily self-explanatory, this file provides a manual.

Shaders The demo constructs shaders as needed using functions from the HLSL code files in this directory. The files *Shadow.fx* and *ParticipatingMedia.fx* are of particular interest. The shaders used by the demo are stored in the directory *CreatedShaders*.

ShaderIncludes Some redundant functionality of the shader functions discussed above has been moved into includes, which you will find in this directory.

Documentation/index.html The shader code includes doxygen comments and this file provides the documentation that has been generated automatically from these comments. The documentation also includes hints on where to start reading.

Author Contact Information

Christoph Peters, Cedrick Münstermann, Nico Wetzstein, Reinhard Klein
Institute of Computer Science II, University of Bonn
Friedrich-Ebert-Allee 144
53111 Bonn
Germany
{peters,muenste,wetzstei,rk}@cs.uni-bonn.de
http://cg.cs.uni-bonn.de/

Peters, C., Münstermann, C., Wetzstein, N. and Klein, R., Improved Moment Shadow Maps, *Journal of Computer Graphics Techniques (JCGT)*, vol. 6, no. 1, 17–67, 2017
http://jcgt.org/published/0006/01/03/

Received:	2016-10-05		
Recommended:	2016-12-13	Corresponding Editor:	Carsten Dachsbacher
Published:	2017-03-30	Editor-in-Chief:	Marc Olano

This work is based on an earlier work: "Beyond Hard Shadows: Moment Shadow Maps for Single Scattering, Soft Shadows and Translucent Occluders", in *Proceedings of the 20th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games,* I3D '16, February 27–28, 2016, Redmond, WA © ACM, 2016.
http://dx.doi.org/10.1145/2856400.2856402