Non-Linearly Quantized Moment Shadow Maps

Christoph Peters
University of Bonn

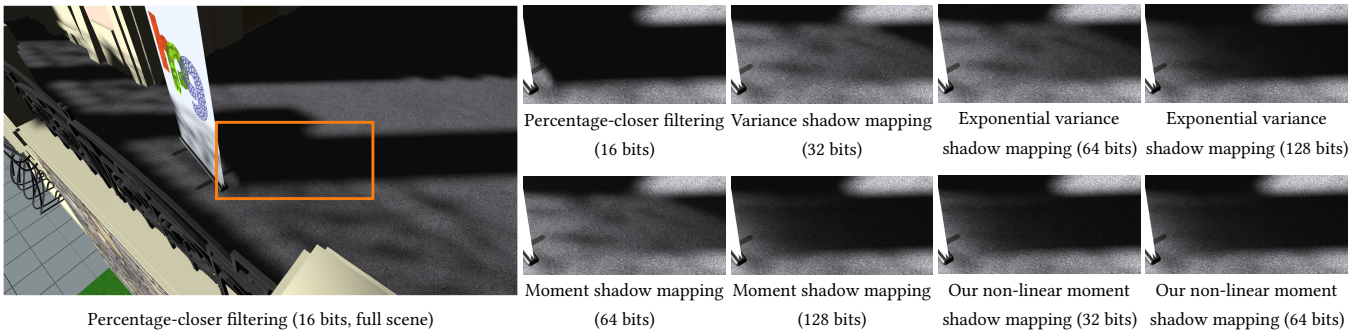


Figure 1: A challenging test case for filterable shadow maps. The railing casts a shadow onto the banner and the banner shadows the ground over a short range. Earlier filterable shadow maps require 128 bits per shadow map texel to avoid light leaking. Our non-linearly quantized moment shadow maps provide a similar quality, even at 32 bits per texel. Exposure has been increased by a factor of five for magnified insets.

ABSTRACT

Moment shadow maps enable direct filtering to accomplish proper antialiasing of dynamic hard shadows. For each texel, the moment shadow map stores four powers of the depth in either 64 or 128 bits. After filtering, this information enables a heuristic reconstruction. However, the rounding errors introduced at 64 bits per texel necessitate a bias that strengthens light leaking artifacts noticeably. In this paper, we propose a non-linear transform which maps the four moments to four quantities describing the depth distribution more directly. These quantities can then be quantized to a total of 32 or 64 bits. At 64 bits, the results are virtually indistinguishable from moment shadow mapping at 128 bits per texel. Even at 32 bits, there is hardly any additional light leaking but banding artifacts may occur. At the same time, the computational overhead for the reconstruction is reduced. As a prerequisite for the use of these quantization schemes, we propose a compute shader that applies a resolve for a multisampled shadow map and a 9^2 two-pass Gaussian filter in shared memory. The quantized moments are written back to device memory only once at the very end. This approach makes our technique roughly as fast as variance shadow mapping with 32 bits per texel. Since hardware-accelerated bilinear filtering is incompatible with non-linear quantization, we employ blue noise dithering as inexpensive alternative to manual bilinear filtering.

HPG '17, July 28-30, 2017, Los Angeles, CA, USA

CCS CONCEPTS

• **Computing methodologies** → *Rasterization*; *Visibility*; *Shared memory algorithms*;

KEYWORDS

Moment shadow mapping, non-linear quantization, 32-bit quantization, on-chip filtering, compute shader, fast Gaussian blur, shadows, real-time rendering

ACM Reference format:
Christoph Peters. 2017. Non-Linearly Quantized Moment Shadow Maps. In *Proceedings of High Performance Graphics conference, Los Angeles, CA, USA, July 28-30, 2017 (HPG '17)*, 11 pages.
https://doi.org/10.1145/3105762.3105775

1 INTRODUCTION

Few problems in real-time rendering call for aggressive optimization as much as dynamic, filtered hard shadows. Most modern applications spend a substantial fraction of their frame time on this effect. In spite of extensive research, percentage-closer filtering [Reeves et al. 1987] with relatively small filter sizes is still the most widely used technique. It provides robust results at an acceptable cost per fragment but can only partially hide shadow map aliasing with practical filter sizes.

Several works [Annen et al. 2007, 2008; Donnelly and Lauritzen 2006; Lauritzen and McCool 2008; Salvi 2008], the most recent one being moment shadow mapping [Peters and Klein 2015], reduce the cost of large filters by enabling direct filtering of the shadow map. To this end, they do not store the depth but vectors computed from the depth in filterable shadow maps with multiple channels. A single filtered sample then allows a heuristic reconstruction of the filtered hard shadow. Light leaking artifacts are a concern but the cost per shaded fragment is reduced substantially and proper antialiasing becomes feasible.

On the other hand, the cost per texel of the shadow map is increased due to higher bandwidth requirements. The stored vectors typically take more memory than a single depth value and common filtering operations such as a resolve for multisampled shadow maps, a two-pass Gaussian blur or mipmapping require multiple passes with reads and writes from and to device memory.

To diminish these costs, we develop a new quantization scheme for moment shadow mapping. While the original technique uses an affine transform [Peters and Klein 2015; Peters et al. 2017], we propose the use of a non-linear quantization transform described in Section 3. This allows us to better maintain the important information and thus reduces light leaking. At 64 bits per texel, the non-linear quantization does not introduce noticeable errors compared to linear quantization at 128 bits per texel and even at 32 bits per texel, there is no significant increase in light leaking but banding artifacts may occur. Furthermore, fewer arithmetic operations are required for shading.

However, the use of a non-linear quantization transform means that we can no longer rely on hardware-accelerated filtering. As inexpensive alternative to manual bilinear filtering, we propose blue noise dithering in Section 4.2. If the moments are filtered beforehand, gradients in the filtered hard shadows are smooth and the dithering patterns are relatively unobjectionable. To enable such filtering, Section 4.1 describes a compute shader that performs a resolve for a multisampled shadow map and applies a two-pass Gaussian blur of size 9^2 to the resulting moments. All of this is done block-wise in shared memory to minimize bandwidth requirements. The filtered and non-linearly quantized moments are written back to device memory only once.

Our evaluation in Section 5 and Figure 1 demonstrates that this approach yields as little light leaking as moment shadow mapping with 128 bits per texel at a cost comparable to variance shadow mapping with 32 bits per texel.

2 RELATED WORK

The most widely used approaches for real-time shadows rely on shadow mapping [Williams 1978] as it has more beneficial performance traits than shadow volumes [Crow 1977] or ray tracing. This image-based approach rasterizes the scene from the point of view of a point light or directional light and stores depth values to characterize the shadowed surfaces. The sampling on screen and in the shadow map can be brought into rough agreement using a special projection transform [Wimmer et al. 2004] or by fitting multiple shadow maps to parts of the view frustum [Lauritzen et al. 2011; Zhang et al. 2006]. The remaining sample offsets necessitate a depth bias which may be computed adaptively [Dou et al. 2014]. A perfect sample distribution is viable with irregular z-buffers [Wyman et al. 2015].

Depths stored in the shadow map may lie anywhere within the first occluder. This enables heavy compression through voxel-based approaches with octrees [Kämpe et al. 2016] or texture-based approaches with quadtrees [Scandolo et al. 2016]. The compressed data structures are designed for fast random access but long compression times limit these techniques to static shadow maps.

Since point lights and directional lights lead to perfectly hard shadow boundaries, the shadow signal contains arbitrarily high frequencies. In consequence, no sampling rate is high enough to avoid aliasing entirely. This problem is made worse by the fact that sampling happens twice, first in light space and then in screen space. A practical way to diminish this aliasing is to filter the shadow signal in light space. These filtered hard shadows are our main concern and we now introduce a stochastic interpretation [Donnelly and Lauritzen 2006; Peters and Klein 2015] to discuss the related work and our approach.

Suppose we apply a filter with $n \in \mathbb{N}$ samples to shadows computed from a shadow map. Let its weights be $w_0, \dots, w_{n-1} \geq 0$ where $\sum_{l=0}^{n-1} w_l = 1$ and let the corresponding sampled depths from the shadow map be $z_0, \dots, z_{n-1} \in [-1, 1]$. These quantities define a depth distribution that we write using Dirac-δ distributions:

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

In this notation, the filtered shadow intensity for a fragment at depth z_f is the probability that a sample from the filter region in the shadow map is less than z_f:

$$Z(\mathbf{z} < z_f) := \sum_{l=0,\ z_l < z_f}^{n-1} w_l \quad \text{where } \mathbf{z}(z) := z.$$

Percentage-closer filtering [Reeves et al. 1987] evaluates this sum directly. Note that it requires n samples from the shadow map per shaded fragment and that n grows quadratically with the radius of the filter.

Compact, filterable representations of the depth distribution Z are obtained through random variables mapping each depth $z \in [-1, 1]$ to a vector $\mathbf{b}(z) \in \mathbb{R}^{m+1}$ where $\mathbf{b}_0(z) := 1$ and $m \in \mathbb{N}$ is the number of channels stored in the filterable shadow map. Then the expectation of this random variable

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

provides information about Z in the form of so-called general moments [Peters and Klein 2015]. These general moments are constructed by mapping each depth z in the shadow map to $\mathbf{b}(z)$ and filtering the m channels of the resulting filterable shadow map directly.

This approach goes back to variance shadow mapping [Donnelly and Lauritzen 2006] where $\mathbf{b}(z) = (1, z, z^2)^\mathsf{T}$. The resulting moments provide mean and variance of Z and enable computation of a lower bound to the shadow intensity through Cantelli's inequality. It serves as approximation. This systematic underestimation avoids wrong self shadowing but leads to light leaking when the variance is large (Figure 1).

Exponential shadow mapping [Annen et al. 2008; Salvi 2008] uses $\mathbf{b}(z) = (1, \exp(c \cdot z))^\mathsf{T}$ where $c \gg 1$ and reconstructs through Markov's inequality. Convolution shadow maps [Annen et al. 2007] set the component functions of $\mathbf{b}$ to Fourier basis functions and reconstruct by means of a truncated Fourier series. Layered variance shadow maps [Lauritzen and McCool 2008] use multiple variance shadow maps for different depth ranges. Exponential variance shadow maps [Lauritzen and McCool 2008] fix parameters

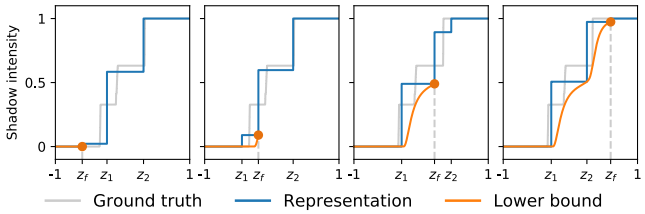


Figure 2: Computation of lower bounds in moment shadow mapping. All representations have the same moments b as the ground truth but only three points of support z_f, z_1, z_2.

Algorithm 1 Reconstruction for moment shadow mapping [Peters and Klein 2015].
Input: Moments $b = \mathcal{E}_Z(\mathbf{b}) \in \mathbb{R}^5$ and fragment depth $z_f \in \mathbb{R}$.
Output: A sharp lower bound to $Z(\mathbf{z} < z_f)$.

(1) Set $B(b) := \begin{pmatrix} b_0 & b_1 & b_2 \\ b_1 & b_2 & b_3 \\ b_2 & b_3 & b_4 \end{pmatrix} \in \mathbb{R}^{3\times 3}$.
(2) Solve $B(b) \cdot q = (1, z_f, z_f^2)^\mathsf{T}$ for $q \in \mathbb{R}^3$.
(3) Solve the quadratic equation $q_2 \cdot z^2 + q_1 \cdot z + q_0 = 0$ for roots $z_1, z_2 \in \mathbb{R}$.
(4) Set $z_0 := z_f$ and solve $\begin{pmatrix} 1 & 1 & 1 \\ z_0 & z_1 & z_2 \\ z_0^2 & z_1^2 & z_2^2 \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}$.
(5) Return $\sum_{l=0,\, z_l < z_f}^{2} w_l$.

$c^+, c^- > 0$ and apply variance shadow mapping to two exponentially warped depths, i.e.

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

Our work is an extension of moment shadow mapping [Peters and Klein 2015]. Therefore, we now provide a detailed explanation of this technique. Moment shadow mapping uses $m = 4$ channels and

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

Thus, $b \in \mathbb{R}^5$ stores five moments of Z where $b_0 = 1$ and does not need to be stored. Based on this information, the technique computes the sharpest possible lower bound to the shadow intensity:

$$Z(\mathbf{z} < z_f) \geq \inf\{S(\mathbf{z} < z_f) \mid S \text{ depth distribution on } \mathbb{R},\ \mathcal{E}_S(\mathbf{b}) = b\}$$

In doing so, it exploits that the minimizing depth distribution S always has a very specific structure shown in Figure 2. It has exactly three points of support, one of which is $z_0 := z_f$. Furthermore, it is proven that the two unknown points $z_1, z_2 \in \mathbb{R}$ are the roots of a particular quadratic polynomial. Once they are available, the weights w_0, w_1, w_2 are the solution of a system of linear equations. Putting these pieces together leads to Algorithm 1 [Peters and Klein 2015]. When the moments are stored in 16-bit fixed-point numbers, biasing and an affine quantization transform help to compensate rounding errors.

Exponential variance shadow mapping and moment shadow mapping both provide practical solutions for generic applications. At 64 bits per texel, moment shadow mapping produces less light leaking in most cases. At 128 bits, moment shadow mapping still leaks light onto surfaces receiving shadow from casters at three or more different depths (Figure 9) while exponential variance shadow mapping leaks slightly more light over short range (Figure 1) and misrepresents shadows at object boundaries [Peters et al. 2017]. The other filterable shadow maps discussed here have more drastic light leaking artifacts or unreasonable bandwidth requirements and are only suited for special cases where these traits are acceptable. Like some of its predecessors, moment shadow mapping is also suited for rendering shadows of transparent occluders, soft shadows or single scattering efficiently by precomputing relevant integrals [Peters et al. 2017].

The preprocessing of filterable shadow maps for hard shadows typically involves a resolve for a multisampled depth buffer, a Gaussian blur or a similar separable filter and generation of mipmaps. Fast separable filters on GPUs are a well-studied problem [Podlozhnyuk 2012; Stewart 2016]. Still, all publicly available solutions that we found write results back to device memory between the horizontal and the vertical pass. By exploiting domain-specific knowledge about the number of channels and desirable filter sizes, we are able to do the resolve, both passes of the separable filter and the non-linear quantization while storing all intermediate results in fast on-chip memory. Thus, we avoid the introduction of quantization errors and minimize bandwidth requirements.

3 NON-LINEAR QUANTIZATION

Further improvements of moment shadow mapping should reduce light leaking and the bandwidth requirements to arrive at a faster and more robust technique. In the following, we demonstrate how to accomplish both by introducing a non-linear quantization transform which maps the moments b to four quantities $\Phi(b) \in \mathbb{R}^4$. These quantities describe the shape of the lower bound in an intuitive manner and thus make it easy to maintain the relevant information.

3.1 A Non-Linear Representation of Moments

Our non-linear quantization transform is based on a previously used descendant of Algorithm 1 [Peters et al. 2017, Algorithm 2], which generates a representation for the first three moments only. The underlying idea is to let the fragment depth z_f go to infinity. As this happens, w_0 approaches zero and we discard it. Let the remaining two depths be y_1, y_2 and let the weights be $v_1, v_2 \in [0, 1]$. Since we discarded w_0, the corresponding depth distribution

$$S := v_1 \cdot \delta_{y_1} + v_2 \cdot \delta_{y_2}$$

no longer has the correct fourth moment b_4 but still matches the other moments $b_0, \ldots, b_3$ [Peters et al. 2017]. Note that $v_1 = 1 - v_2$. Thus, we describe the three moments b_1, b_2, b_3 completely through the quantities y_1, y_2, v_2. In the supplementary we prove that $y_1, y_2 \in [-1, 1]$ if Z is a depth distribution on $[-1, 1]$.

Of course, b_4 must not be discarded entirely. Rather than using it directly, we consider the offset of the fourth moment with respect to $\mathcal{E}_S(\mathbf{b}_4)$:

$$\xi_4 := b_4 - \mathcal{E}_S(\mathbf{b}_4) = b_4 - v_1 \cdot y_1^4 - v_2 \cdot y_2^4 \in \mathbb{R} \qquad (1)$$

Interestingly, it is always non-negative. To see this, we consider the Hankel matrix $B(\mathcal{E}_S(\mathbf{b}))$ as defined in Algorithm 1. Being the

Algorithm 2 Non-linear representation of moments.
Input: Moments $b = \mathcal{E}_Z(\mathbf{b}) \in \mathbb{R}^5$.
Output: A non-linear representation $(y_1, y_2, v_2, \xi_4) = \Phi(b)$.

(1) Set $\sigma^2 := q_2 := b_2 - b_1^2$.
(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 $y_1, y_2 \in \mathbb{R}$.
(5) Set $v_2 := \frac{b_1 - y_1}{y_2 - y_1}$.
(6) Set $L_{2,1} := \frac{b_3 - b_1 \cdot b_2}{\sigma^2}$.
(7) Set $\xi_4 := b_4 - b_2^2 - L_{2,1}^2 \cdot \sigma^2$.
(8) Return (y_1, y_2, v_2, ξ_4).

Hankel matrix for a distribution S with exactly two points of support, it is known to be positive semi-definite but singular [Peters and Klein 2015, Proposition 4]. The determinant of $B(b)$ is monotonically increasing with b_4 because the variance $\sigma^2 := b_2 - b_1^2$ is non-negative and

$$\det \begin{pmatrix} b_0 & b_1 & b_2 \\ b_1 & b_2 & b_3 \\ b_2 & b_3 & b_4 \end{pmatrix} = \det \begin{pmatrix} 1 & b_1 & 0 \\ b_1 & b_2 & 0 \\ b_2 & b_3 & b_4 \end{pmatrix} + \det \begin{pmatrix} 1 & b_1 & b_2 \\ b_1 & b_2 & b_3 \\ b_2 & b_3 & 0 \end{pmatrix}$$
$$= \sigma^2 \cdot b_4 + B((1, b_1, b_2, b_3, 0)^\mathsf{T}). \quad (2)$$

Thus, $\xi_4 < 0$ implies $\det B(b) < 0$ which contradicts our knowledge that the Hankel matrix is positive semi-definite [Peters and Klein 2015, Proposition 4].

Equation (2) also offers a useful way to compute ξ_4 when $\sigma^2 \neq 0$. By Equations (1) and (2), both ξ_4 and $\det B(b)$ depend linearly on b_4. Furthermore, $\xi_4 = 0$ if and only if $\det B(b) = 0$. Taking the gradient in Equation (2) into account yields

$$\xi_4 = \frac{\det B(b)}{\sigma^2}. \quad (3)$$

If we construct the Cholesky decomposition $B(b) = L \cdot D \cdot L^\mathsf{T}$ with diagonal $D \in \mathbb{R}^{3\times3}$ and lower unitriangular $L \in \mathbb{R}^{3\times3}$, the first two diagonal entries of D are 1 and σ^2. Since $\det B(b) = \det D$, the third diagonal entry has to be ξ_4 by Equation (3). Computing ξ_4 in this manner takes slightly fewer instructions than Equation (1) and more importantly the computation does not depend on y_1, y_2 or v_2 which is beneficial to instruction-level parallelism.

This derivation leads to Algorithm 2 which includes the previously proposed algorithm [Peters et al. 2017, Algorithm 2] but additionally computes ξ_4. It works robustly in single-precision arithmetic. Considering that the input moments are provided in single precision as well, the usual biasing for single-precision moments should be applied to the input (see Section 3.2). This also ensures $\sigma^2 > 0$ and $y_1 \neq y_2$.

3.2 The Cause of Light Leaking

The offset of the fourth moment ξ_4 may be viewed as a measure of light leaking. In the ideal case, $\xi_4 = 0$, we know that $Z = S$ because no other depth distribution can reproduce these moments [Peters and Klein 2015, Proposition 4]. This case occurs whenever the filter region in the shadow map contains exactly two depth values, which is approximately true when the filter region covers the silhouette of an occluder. Compared to the distance between receiver and occluder, the variation of depths on the two surfaces is small.

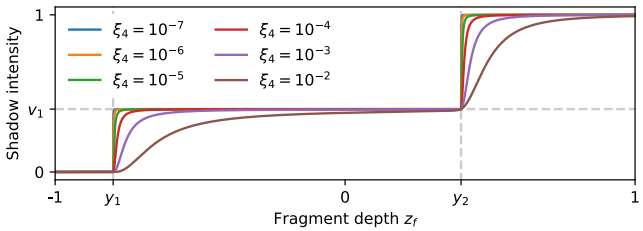


Figure 3: Lower bounds computed using Algorithm 1 for $y_1 = -0.8$, $y_2 = 0.4$, $v_2 = 1 - v_1 = 0.6$ and various values of ξ_4.

For the ideal case with $\xi_4 = 0$, the shadow intensity is reconstructed perfectly at all depths. The reconstruction has two steps of height $v_1 = 1 - v_2$ and v_2 at depths y_1 and y_2. As ξ_4 grows, these steps become less steep and the lower bound decreases everywhere as shown in Figure 3. What is remarkable is how rapidly this happens. At a value of $\xi_4 = 10^{-4}$ we should already expect noticeable light leaking in short-range shadows. Such moderate values of ξ_4 may occur naturally when the depth distribution Z cannot be approximated well by two depth values but in practice these cases are rare (see Figure 4).

The other major reason for larger values of ξ_4 is biasing. Rounding errors frequently turn valid vectors of moments into invalid vectors where $\xi_4 < 0$. To pull the vector of moments b back into the valid domain, the biasing applies a simple linear interpolation. It replaces b by the biased vector

$$b' := (1 - \alpha) \cdot b + \alpha \cdot b^\star \in \mathbb{R}^5$$

where $b^\star \in \mathbb{R}^5$ and $\alpha \in [0, 1]$ depend on the quantization scheme. When using the affine quantization transform with 64 bits per texel, the recommended values are $b^\star = (1, 0, 0.628, 0, 0.628)^\mathsf{T}$ and $\alpha = 6 \cdot 10^{-5}$ [Peters et al. 2017]. For moments stored in single-precision numbers, the values $b^\star = (1, 0, 0.375, 0, 0.375)^\mathsf{T}$ and $\alpha = 3 \cdot 10^{-7}$ should be used [Peters et al. 2017] and are used for the input of Algorithm 2.

The effect of the 64-bit biasing on y_1, y_2 and v_2 is completely negligible. The same cannot be said for ξ_4. In the most extreme case, which occurs for $Z = \frac{1}{2} \cdot \delta_{0.915} + \frac{1}{2} \cdot \delta_1$, ξ_4 is zero before biasing and $2.8 \cdot 10^{-4}$ afterwards. More typical increases are in the same magnitude as α. Such an increase is sufficient to lead to light leaking in short-range shadows.

It seems tempting to bias ξ_4 in a more explicit manner that preserves small values. Doing so does reduce light leaking but inevitably leads to overestimation of the shadow intensity. This happens rarely but results in obvious artifacts. Once the four moments are stored in 16-bit fixed-point numbers, information is lost and cannot be recovered. Therefore, the quantization itself has to change.

3.3 Non-Linear Quantization at 64 Bits

In the following, we assume that we are given a filtered vector of moments b in single precision. This assumption is justified through Section 4.1. As last step during generation of the moment shadow

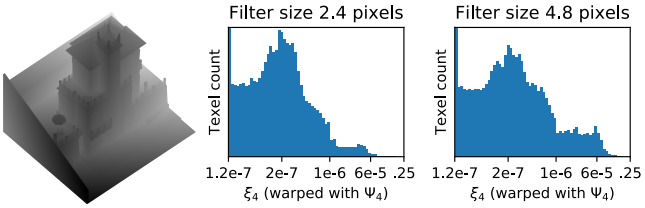


Figure 4: Left: A 2048^2 shadow map with 4× MSAA. Right: Histograms for the offset of the fourth moment ξ_4 after filtering with a Gaussian filter of specified standard deviation. The warp Ψ_4 makes them relatively uniform. Note the prevalence of small values. The left-most bar is cut off.

map, we wish to store it at 64 bits per texel while minimizing loss of information.

For simplicity, we use a texture format with four unsigned 16-bit fixed-point numbers. However, we do not store the moments directly. Instead we store the quantities generated by Algorithm 2. Storing the depths $y_1, y_2 \in [-1, 1]$ to the first two channels directly works well. The weight $v_2 \in [0, 1]$ can also be stored directly.

For the offset of the fourth moment ξ_4, it would be ill-advised to store it directly. Most of the time, its value is extremely small but loss of information has to be minimized. We need to find a monotonic function that warps ξ_4 to make its distribution more uniform. This function and its inverse have to be efficiently evaluable in shaders.

In the supplementary we prove $\xi_4 \leq 0.25$. If we clamp ξ_4 to a minimal value of $1.2 \cdot 10^{-7}$, this does not change any computed shadow intensities significantly. Thus, we are looking for a warping function $\Psi_4 : \left[1.2 \cdot 10^{-7}, 0.25\right] \to [0, 1]$.

To find it, we generate a moment shadow map for the 2048^2 shadow map in Figure 4, filter it, apply biasing for single-precision moments and compute ξ_4 per texel. Then we try to construct an efficiently evaluable function Ψ_4 that equalizes the histogram of these values. Some experimentation led to

$$\ln(\ln(\xi_4 \cdot 10^7)).$$

For $\xi_4 \searrow 10^{-7}$ it approaches $-\infty$ which is why we need to clamp at $1.2 \cdot 10^{-7}$. Additionally, we need to normalize the outputs. Then the complete function is

$$\Psi_4(\xi_4) := \frac{\ln(\ln(\max\{1.2 \cdot 10^{-7}, \xi_4\} \cdot 10^7)) - \ln(\ln(1.2))}{\ln(\ln(0.25 \cdot 10^7)) - \ln(\ln(1.2))}.$$

Note that the normalization is only a single multiply-add operation.

Figure 4 shows the resulting histograms for different filter sizes. These histograms are not perfectly uniform and differ considerably. Still, this quantization is drastically better than uniform quantization and we found this to be true for a variety of scenes. It is remarkable that the majority of values is less than 10^{-6} and only few values are larger than $6 \cdot 10^{-5}$. The original 64-bit quantization would make most values surpass $6 \cdot 10^{-5}$. Storing $\Psi_4(\xi_4)$ in the fourth channel completes our non-linear quantization scheme at 64 bits per texel.

3.4 Efficient Shading

Once we have retrieved y_1, y_2, v_2 and ξ_4 from a non-linearly quantized moment shadow map, we can shade a fragment at depth z_f. The obvious approach is to reconstruct the moments via

$$b = (1 - v_2) \cdot \mathbf{b}(y_1) + v_2 \cdot \mathbf{b}(y_2) + (0, 0, 0, 0, \xi_4)^\mathsf{T} \in \mathbb{R}^5 \qquad (4)$$

and to use them as input to Algorithm 1. However, there is a far more efficient way.

Equation (4) leads to a factorization of the Hankel matrix $B(b)$:

$$\begin{aligned} B(b) &= (1 - v_2) \cdot \begin{pmatrix} 1 \\ y_1 \\ y_1^2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ y_1 \\ y_1^2 \end{pmatrix}^\mathsf{T} + v_2 \cdot \begin{pmatrix} 1 \\ y_2 \\ y_2^2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ y_2 \\ y_2^2 \end{pmatrix}^\mathsf{T} + \xi_4 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^\mathsf{T} \\ &= \begin{pmatrix} 1 & 1 & 0 \\ y_1 & y_2 & 0 \\ y_1^2 & y_2^2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 - v_2 & 0 & 0 \\ 0 & v_2 & 0 \\ 0 & 0 & \xi_4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 0 \\ y_1 & y_2 & 0 \\ y_1^2 & y_2^2 & 1 \end{pmatrix}^\mathsf{T} \end{aligned} \qquad (5)$$

To simplify this factorization further, we scale and shift the domain of depth values to ensure $y_1 = 0$ and $y_2 = 1$. Since the only other depth value is z_f, this is achieved by setting

$$z_f' := \frac{z_f - y_1}{y_2 - y_1}.$$

The supplementary proves that the correct transform for ξ_4 is

$$\xi_4' := \frac{\xi_4}{(y_2 - y_1)^4}.$$

The last line of Algorithm 1 outputs different sums of weights dependent on the relation between z_f and z_1, z_2 (see Figure 2). In the supplementary, we prove that y_1, y_2 conveniently characterize the different cases. Assuming $y_1 < y_2$, the shadow intensity is zero if $z_f \leq y_1$, w_1 if $y_1 < z_f \leq y_2$ and $w_1 + w_2 = 1 - w_0$ if $y_2 < z_f$. Thus, the method can terminate immediately when it finds that $z_f \leq y_1$.

For the other two cases, we employ an alternative method to compute w_0 or w_1. For $l \in \{0, 1, 2\}$, we know [Peters and Klein 2015, Proposition 10]

$$w_l = \frac{1}{(1, z_l, z_l^2) \cdot B^{-1}(b) \cdot (1, z_l, z_l^2)^\mathsf{T}}.$$

Since $z_0 = z_f$ is known, it is possible to compute w_0 immediately without computing z_1 or z_2. It can be accelerated further by exploiting $y_1 = 0$, $y_2 = 1$ and the decomposition in Equation (5). This way, the Hankel matrix $B(b)$ is just the product of a constant matrix, a diagonal matrix and the transposed constant matrix. Computation of w_1 works in the same manner except that z_1 has to be computed first. This is done with the quadratic formula as in Algorithm 1 but utilizing the decomposition in Equation (5).

Algorithm 3 handles all three cases. For readability some common subexpressions have not been eliminated. Note that there are no conditionals where both branches are expensive, i.e., branch divergence is only a minor concern. Occasionally the rounded inputs to this algorithm will fulfill $y_1 = y_2$, $v_2 = 0$ or $v_2 = 1$. All of these cases correspond to zero variance and lead to undefined behavior. Dedicating a branch to these cases works well and is easily accomplished but it is more efficient to clamp away from the problematic values with a small tolerance. We use 10^{-6} such that the introduced error is less than the rounding errors at 64 bits per texel.

Algorithm 3 Optimized version of moment shadow mapping (Algorithm 1) for non-linearly quantized moments.
Input: Output of Algorithm 2 (y_1, y_2, v_2, ξ_4), fragment depth z_f.
Output: A sharp lower bound to $Z(\mathbf{z} < z_f)$.

- If $z_f \leq y_1$: Return 0.
- Set $z'_f := \frac{z_f - y_1}{y_2 - y_1}$ and $\xi'_4 := \frac{\xi_4}{(y_2 - y_1)^4}$.
- Set $z_l := z'_f$.
- If $z_f \leq y_2$:
  - Set $q_0 := \frac{-\xi'_4}{(1-v_2)\cdot z'_f}$ and $q_1 := -q_0 \cdot \frac{v_2 - z'_f}{v_2 - v_2 \cdot z'_f} - 1$.
  - Set $z_l := -\frac{q_1}{2} - \sqrt{\frac{q_1^2}{4} - q_0}$.
- Set $w_l := \frac{1}{(1-v_2)^{-1}\cdot(1-z_l)^2 + v_2^{-1}\cdot z_l^2 + \xi'^{-1}_4 \cdot (1-z_l)^2 \cdot z_l^2}$.
- Return w_l if $z_f \leq y_2$ and $1 - w_l$ otherwise.

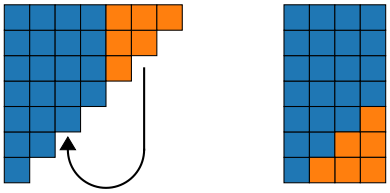


Figure 5: Left: Since $\tilde{y}_1 \leq \tilde{y}_2$, all pairs lie in a triangle. Right: Rotating one part by 180° halves the extent of the domain and saves one bit. Note that this example uses $2 \cdot 3$ bits.

3.5 Non-Linear Quantization at 32 Bits

To further reduce memory requirements, we now demonstrate how to pack the four non-linearly quantized moments into a single 32-bit integer. The warped offset of the fourth moment $\Psi_4(\xi_4) \in [0, 1]$ introduced in Section 3.3 is simply stored as fixed-point number in six bits. During loading, we ensure that it is rounded up to guarantee a lower bound to the shadow intensity.

For the weight $v_2 = 1 - v_1$, we observe that it is often directly proportional to the brightness contributed by a light source (see Figure 3). Thus, smaller values should be stored at higher precision to account for human perception. For efficiency reasons, we simply store $\sqrt{v_2} \in [0, 1]$ as fixed-point number in seven bits.

We are left with 19 bits to store y_1 and y_2. Without loss of generality, these two depths fulfill $y_1 < y_2$. Exploiting this inequality as shown in Figure 5 allows us to save one bit such that both depths are effectively stored as 10-bit fixed-point numbers. We first map both depths to integers in the range from 0 to 1022 via $\tilde{y}_l = \lfloor 511.5 \cdot (y_l + 1) \rfloor$. Then if $\tilde{y}_1 \geq 512$, the integers $\tilde{y}_1$ and $\tilde{y}_2$ are replaced by $1023 - \tilde{y}_1$ and $1022 - \tilde{y}_2$.

The bit counts stated here are the result of experimentation. Decreasing the precision of either v_2 or ξ_4 further leads to noticeable banding. Figure 8 shows an example of insufficient depth precision.

4 ON-CHIP FILTERING

In Section 3.3 we start from the assumption that we are given a readily filtered vector of moments in four single-precision floats. Originally, moment shadow mapping does the filtering in multiple passes [Peters and Klein 2015; Peters et al. 2017]. In the first pass, each sample from a multisampled shadow map is turned into a vector of moments and the samples are averaged per texel for the resolve. The result is written back to a texture and then two more passes apply a horizontal and vertical Gaussian filter. This method would lead to bandwidth requirements that defeat the goal of our work entirely, especially when using single-precision floats.

4.1 Filtering in Shared Memory

To avoid these bandwidth requirements, we utilize a compute shader that generates the filtered moments one block at a time. A complete block from the multisampled shadow map, including a guard band to account for the size of the Gaussian filter, is loaded and moments are stored into shared memory. Then the horizontal and vertical Gaussian filters operate on the floats in shared memory. Only at the very end, the non-linear quantization is applied and the results are written back to device memory.

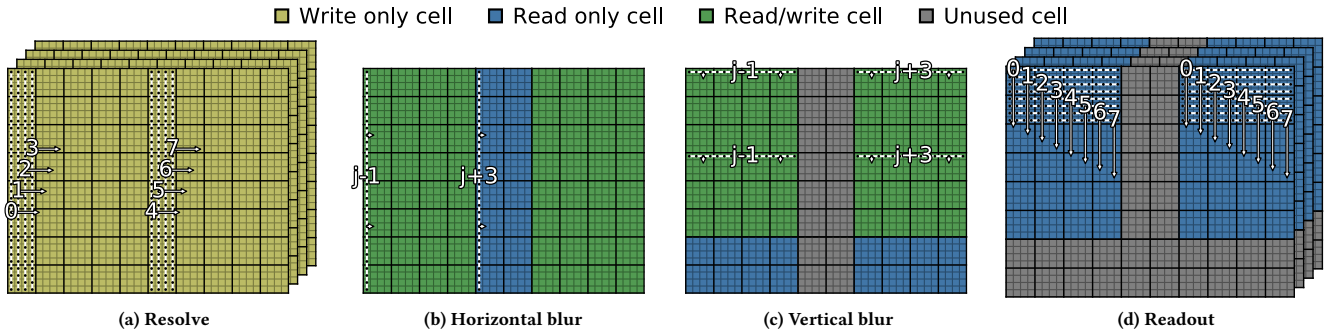
Use of a compute shader is particularly attractive on platforms that support asynchronous compute. Typically, generation of multisampled shadow maps only uses vertex shaders and thus compute units are poorly utilized while the rasterizer does most of the work. When multiple shadow maps are needed for multiple light sources or a single directional light [Lauritzen et al. 2011; Zhang et al. 2006], the compute shader can process some of them while the rasterizer produces others. In this manner, the cost of the compute shader is hidden. Although this optimization is very promising on compatible platforms, we have not tested it in our implementation.

We develop and profile this shader on an NVIDIA GeForce GTX 970 GPU. To some extent our optimizations are specific to this hardware. In particular, we exploit that the 32 threads in one warp execute concurrently. This is true for all current NVIDIA hardware and on AMD hardware 64 threads execute concurrently. It has implications for thread synchronization and for the optimal use of the memory system.

Resolve. To get adequately sized workloads for each warp in each step, we generate output blocks of size 32×24. The Gaussian filter has size 9^2 and a standard deviation of 2.4 texels. Therefore, the input blocks with the guard band have size 40×32. The thread group consists of 256 threads or eight warps. In the first phase each thread loads all samples for one texel, generates averaged moments and writes them to shared memory. This is repeated five times until the whole input block is loaded. Each warp operates on one column at a time (Figure 6a). Given the size of the block in shared memory, our target hardware supports four concurrent thread groups per streaming multiprocessor or 1024 threads. This results in 50% occupancy which is sufficient for latency hiding, thanks to strong instruction-level parallelism.

Two-pass blur. For the blur, we give each of the eight warps an independent workload. Firstly, the four moments can be filtered independently. Secondly, we halve the input block vertically into two blocks of size 20×32. The eight columns in the middle are read only, such that race conditions between warps are impossible (Figures 6b, 6c).

To blur horizontally, each thread loads 24 consecutive moments from one row into registers. Each warp reads one complete column



(a) Resolve (b) Horizontal blur (c) Vertical blur (d) Readout

Figure 6: An overview of our compute shader. The resolve writes moments to shared memory, they are blurred in two passes and then read out, quantized and written to device memory. The cells correspond to locations within the 40×32 block. White lines indicate warps annotated with the warp index. Each black dot is a thread, arrows indicate the stride in loops. The index $j \in \{1, 2, 3, 4\}$ refers to the index of a moment b_j.

at a time (Figure 6b). Then the blurring is done through brute force convolution in registers. The 16 filtered moments are then written back to the part of the row that is not read only.

The vertical blur now has to operate on a strip of size 16×32. The first half of threads within a warp processes the upper half of this strip. The second half processes the lower half. Thus, each warp processes two rows at a time (Figure 6c). Each thread loads 20 consecutive moments from a column into registers. After blurring them in registers, the result is written back to the source strip.

Readout and quantization. Finally, each thread reads three complete vectors of moments, applies non-linear quantization to each of them and writes them to the non-linearly quantized moment shadow map. In this step, each warp processes one row at a time (Figure 6d).

Note that all 256 threads are active throughout the entire algorithm. Conditionals are only needed to avoid out-of-range writes when the output block size is not aligned with the shadow map size. The implementation exploits that the guard band size eight is a divisor of the warp size 32. Therefore, larger filters would be hard to implement and smaller filters should use the same procedure with some zero filter weights.

Thread synchronization. If the warp size is known to be a multiple of 32, we only need a barrier synchronizing all threads in a group after the resolve and another one before the readout. Otherwise, additional barriers have to synchronize between the horizontal and vertical blur and between reading and writing in the vertical blur.

Memory layout. Additional effort went into avoidance of bank conflicts. For each access to shared memory, all indices in one warp should be distinct modulo 32. We found a storage pattern that accomplishes this throughout the algorithm with minimal index arithmetic and padding entries. Each row of 40 moments is stored in 40 consecutive array entries. However, the left 20 moments use all even indices and the right 20 moments use all odd indices. After four consecutive rows, there is one empty array entry to swap odd and even indices. The blocks for the four moments are stored consecutively. Thus, the moment b_j where $j \in \{1, 2, 3, 4\}$ coming out of the resolve at location $x \in \{0, \dots, 39\}$, $y \in \{0, \dots, 31\}$ is stored at index

$$(40 \cdot 32 + 7) \cdot (j - 1) + 40 \cdot y + \left\lfloor \frac{y}{4} \right\rfloor + \begin{cases} 2 \cdot x & \text{if } x < 20, \\ 2 \cdot x - 39 & \text{otherwise.} \end{cases}$$

The alternation between odd and even indices helps to avoid bank conflicts when accessing a complete column (Figures 6a, 6b) and when accessing two rows that are 12 cells apart (Figure 6c). To validate that this scheme avoids all bank conflicts, we have simulated the memory accesses of the warps offline. Note that we do not require expensive index arithmetic per access. Only a few indices are computed and then the remaining indices are obtained by adding constants that the compiler precomputes for us.

4.2 Blue Noise Dithering as Alternative to Bilinear Interpolation

A convenient aspect of moment shadow mapping is that bilinear interpolation can be applied to the moment shadow map. Unfortunately, the non-linear quantization destroys this property. To get true bilinear interpolation, we would have to load four texels, apply Algorithm 3 four times and interpolate the shadow intensities. This would be costly, especially in terms of arithmetic operations. On the other hand, nearest neighbor interpolation leads to an unpleasant blocky appearance.

To get a better result without a significant increase in cost, we use blue-noise dithering [Ulichney 1988]. Given texture coordinates for the shadow map using texels as unit, we retrieve two uniform noise values in $[0, 1]$, add them to the texture coordinates and then round down to integer coordinates to fetch a single texel. The noise values are blue noise, i.e. the low-frequency components are very weak and the Fourier transform is approximately radially symmetric [Ulichney 1988]. Even slight blurring, such as it may occur when the human visual system cannot fully resolve each pixel on screen, turns such high-frequent patterns into nearly uniform areas. The results are far more pleasant than with common white noise as shown in Figure 7. The low contrasts between neighboring texels, which result from the Gaussian blur described in Section 4.1, make the dithering patterns still harder to perceive.

We use the void-and-cluster method [Ulichney 1993] to precompute such blue noise into textures[1]. The Gaussian filter used within this technique is set to a standard deviation of 1.9 pixels. The resulting textures are tileable and thanks to the weak low-frequency content repetitions are hard to notice, even for small tiles. We use tiles of size 64^2 with two channels stored at 8 bits per channel. One such texture has a size of 8 kB and can fit into L1 cache entirely. The textures are always accessed using the screen space pixel index because resampling would destroy the blue noise properties.

To avoid fixed screen space patterns, we generate 64 of these 64^2 textures. All of them are loaded into a texture array. In each frame, a constant with random bits provides the index of the used blue noise texture and random horizontal and vertical offsets. This randomization should be particularly helpful in combination with temporal antialiasing but we have not tried this. Even without such smoothing, it makes the dither patterns much less noticeable.

5 RESULTS

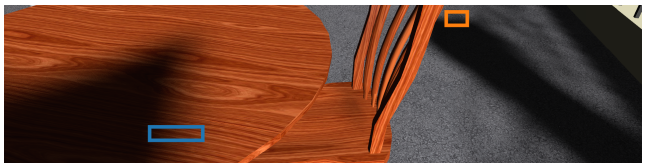
We have implemented all of our techniques in a Direct3D 11 based forward renderer and ran on an NVIDIA GeForce GTX 970. Our implementations of filterable shadow maps from the related work do not use the compute shader described in Section 4, although the ones with four channels could benefit from it. Instead they use three pixel shader passes for the resolve, the horizontal and the vertical blur. The blurs use bilinear interpolation to halve the number of samples. Our implementation of percentage-closer filtering relies on hardware-acceleration to process four samples at once. Experiments use the scene shown in Figure 10d with a single directional light. The 2048^2 shadow map with 4× multisample antialiasing (MSAA) is simply fitted to the convex hull, independent of the view frustum. All screenshots are captured at 1920 × 1080 with 4× MSAA.

5.1 Qualitative Evaluation

Figure 1 compares all considered techniques in a setting that provokes light leaking in short-range shadows. Percentage-closer filtering handles it well except for a missing contact shadow due to a high depth bias. Variance shadow mapping [Donnelly and Lauritzen 2006] suffers from strong light leaking and so does exponential variance shadow mapping [Lauritzen and McCool 2008] when using the exponents supported at 64 bits per texel ($c^+ = c^- = 5.5$) and moment shadow mapping [Peters and Klein 2015] at 64 bits per texel ($\alpha = 6 \cdot 10^{-5}$). At 128 bits per texel exponential variance shadow mapping ($c^+ = 40$, $c^- = 5.5$) and moment shadow mapping ($\alpha = 3 \cdot 10^{-7}$) both provide results with very little light leaking. In spite of the lower memory overhead, the results of our proposed non-linear moment shadow mapping at 64 bits per texel are practically identical to those of common moment shadow mapping at 128 bits per texel. Even at 32 bits per texel, the results are only slightly worse.

We have been unable to find any cases where non-linear quantization at 64 bits per texel has a significant negative impact on the reconstruction quality. For example, when we render the image in Figure 10d using moment shadow mapping at 128 bits with nearest neighbor interpolation and compare to non-linear quantization at 64 bits, we find that only 16 pixels have an absolute difference in the shadow intensity above 3%. The highest measured difference is below 10%. These rare differences are perceivable in direct comparison but cannot be considered objectionable artifacts. Only if the depth range is stretched heavily, they may become problematic.

Relevant differences are entirely due to the different approaches to interpolation. Figure 7 shows results of the blue-noise dithering that we use for non-linearly quantized moment shadow maps in comparison to alternatives such as true bilinear interpolation. Among the options with a single unfiltered sample per pixel, it is clearly the most pleasant result. Nonetheless, there is visible noise that is not present with bilinear interpolation. Whether this noise is acceptable in practice depends on various factors. A large filter for the moment shadow map (Section 4.1), lower contrasts in the shadows and more detailed surface textures make it less noticeable. Low pixel densities on screens and shorter viewing distances make it more obvious.



(a) Full scene with blue noise dithering

(b) Bilinear interpolation of moments

(c) Nearest neighbor interpolation

(d) White noise dithering

(e) Blue noise dithering

Figure 7: Non-linear moment shadow mapping at 64 bits uses blue-noise dithering instead of bilinear interpolation. When the moment shadow map is magnified, this results in stippling patterns, under minification it yields subtle noise. The quality is inferior to true bilinear interpolation but the cost is comparable to nearest neighbor interpolation.

[1]Our implementation and the used textures are available online [Peters 2016].

(a) Non-linear moment shadow mapping (32 bits)

(b) Non-linear moment shadow mapping (64 bits)

Figure 8: A protrusion in the facade casts a shadow over very short range. When quantizing the depths y_1, y_2 to 10 bits, this leads to visible banding whereas 16 bits are sufficient.

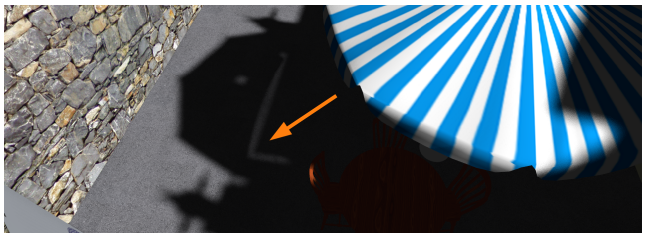
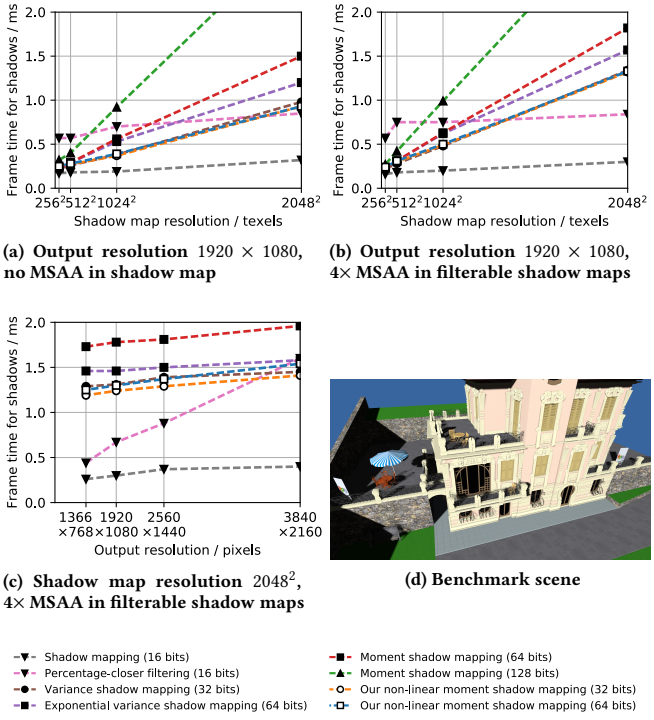


Figure 9: A failure case of non-linear moment shadow mapping at 64 bits. The line on the ground receives shadow from the umbrella, the roof and a railing. Together, these three occluders lead to light leaking.



(a) **Output resolution** 1920 × 1080, **no MSAA in shadow map**

(b) **Output resolution** 1920 × 1080, **4× MSAA in filterable shadow maps**

(c) **Shadow map resolution** 2048^2, **4× MSAA in filterable shadow maps**

(d) **Benchmark scene**

Figure 10: Frame times for rendering shadows in the shown scene. The full frame time for rendering without shadows and shading is subtracted from the frame time for rendering with shadows but without shading to get the values shown.

As demonstrated in the supplementary video, results are even better in an interactive setting because the noise changes each frame. This is particularly true at high refresh rates. We expect further improvements when the technique is coupled with temporal antialiasing. Note that direct bilinear interpolation of the moments enables use of a slightly lower depth bias. With non-linear quantization we use a constant depth bias of 10^{-3} depth units for all results.

The biggest problem for quantization at 32 bits per texel is that 10 bits are not always enough for the depth values y_1, y_2. This leads to distortions in shadows cast over very short range which manifest as banding artifacts (Figure 8). Especially when the light or the geometry is moving, these bands can be a rather distracting artifact. Except for this relatively rare artifact, the results are usually close to the results of non-linear quantization at 64 bits per texel.

Moment shadow maps with four channels can impossibly handle all situations correctly. When the filter region in the shadow map contains three or more occluders, light will leak onto fully shadowed surfaces. Non-linearly quantized moment shadow maps with 64 bits per texel inherit this artifact as shown in Figure 9.

5.2 Run Times

Figure 10 provides run time measurements for shadowing the scene in Figure 10d. The graphs in Figures 10a and 10b allow us to understand the cost per shadow map texel. It is very low for shadow mapping [Williams 1978] and percentage-closer filtering [Reeves et al. 1987]. Thanks to the optimized compute shader (Section 4.1), our novel non-linearly quantized moment shadow maps have a cost per texel similar to that of variance shadow mapping [Donnelly and Lauritzen 2006], which does not utilize our optimized compute shader. For reasons that we investigate below, there is no significant difference between the 32- and 64-bit variants. Filterable shadow maps with 64 or even 128 bits per texel have a substantially higher cost.

When we do not use MSAA for the filterable shadow maps (Figure 10a), non-linearly quantized moment shadow maps are about as fast as percentage-closer filtering with a 9^2 filter at a shadow map resolution of 2048^2 and much faster for smaller shadow maps. MSAA, which is not applicable for percentage-closer filtering, adds to this cost substantially (Figure 10b) but also has a tremendous positive effect on the quality of the shadows.

While percentage-closer filtering has a low cost per texel of the shadow map, its cost per shaded fragment is very high (Figure 10c). On the other hand, all filterable shadow maps have a very low cost per shaded fragment. The overhead at a resolution of 3840×2160 is around 0.3 ms for all of them. This includes the blue noise dithering for our non-linear moment shadow mapping. Again, there is no substantial difference between the 32- and 64-bit variants of non-linear moment shadow mapping.

We also captured GPU timings for the compute shader, discussed in Section 4.1, for generation of a non-linearly quantized moment shadow map with 64 bits per texel. For a shadow map without MSAA at a resolution of 2048^2 it takes 0.53 ms. This is a considerable improvement over the 1.25 ms required to generate a 2048^2 moment shadow map at 64 bits with classic three-pass filtering. The reduced run time of our novel technique is mostly due to this optimization. With 4× MSAA the run time for the compute shader does not grow but three-pass filtering now takes 1.42 ms.

When quantizing into 32 bits per texel, this run time does not change significantly. With 4× MSAA and 128 bits per texel, the run time rises to 0.71 ms. The reasons become evident through more detailed profiling. We have reimplemented the compute shader without MSAA as CUDA kernel and used NVIDIA Visual Profiler to analyze it. This analysis shows that the kernel is memory bound although compute utilization is not much lower. In the memory system, the shared memory bandwidth is the bottleneck. Therefore, reducing bandwidth requirements for device memory through quantization into 32 bits per texel is not beneficial. Only at 128 bits per texel, device memory requirements become the bottleneck.

6 CONCLUSIONS

In essence, non-linearly quantized moment shadow maps with 64 bits per texel offer the quality of moment shadow mapping with 128 bits per texel at the cost of variance shadow mapping with 32 bits per texel. Further speedups should be possible through asynchronous compute. Strong light leaking is seldom such that the technique offers a suitable replacement for percentage-closer filtering on general scenes. This way, shadow map aliasing can be diminished effectively. With small shadow maps or at large output resolutions, e.g., in 4k rendering or virtual reality, it is also faster than percentage-closer filtering.

The variant with 32 bits per texel works better than one might expect but on the tested hardware the reduction in run time is too insignificant to justify the strengthened artifacts. These findings may be different on other hardware.

Use of non-linearly quantized moment shadow maps is more intricate than use of common moment shadow maps. Especially the compute shader may necessitate platform-specific optimizations to get optimal performance. To ease adaptation, we provide full HLSL code for all novel techniques. The definitive version of the paper will also come with an executable demo. Note that our compute shader may also benefit other applications requiring a separable filter up to 9^2 for a texture with four channels.

Unlike common moment shadow maps, the non-linearly quantized counterpart is incompatible with hardware-accelerated filtering. When the shadow map is sufficiently filtered, blue noise dithering can be a viable alternative to bilinear interpolation and mipmapping or anisotropic filtering are barely needed. Another consequence from the non-linear quantization is that the technique is incompatible with summed-area tables and therefore it cannot be combined with moment soft shadow mapping [Peters et al. 2017]. It could be combined with prefiltered single scattering but since light leaking is less of an issue there, it is not opportune to do so [Peters et al. 2017]. On the other hand, the combination with stochastic shadow maps seems promising [Enderton et al. 2010; McGuire and Mara 2017; Peters et al. 2017].

ACKNOWLEDGMENTS

We would like to thank Luca Sassone for modeling the scene used for experiments throughout this paper. We have augmented this scene with additional furniture models by Doug Cummins, Jeffie, Mat Behr, Matt Mump and Przemek. All of these models have been made available on BlendSwap.com under a CC0 license. We are also grateful to Cedrick Münstermann for his constructive input and to Matt Pettineo who motivated this project through his feedback on moment shadow mapping.

REFERENCES

Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. 2007. Convolution Shadow Maps. In *EGSR07: 18th Eurographics Symposium on Rendering*. Eurographics Association, 51–60. https://doi.org/10.2312/EGWR/EGSR07/051-060

Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. 2008. Exponential Shadow Maps. In *GI '08: Proceedings of graphics interface 2008*. Canadian Information Processing Society, 155–161. https://doi.org/10.20380/GI2008.20

Franklin C. Crow. 1977. Shadow Algorithms for Computer Graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques (SIGGRAPH '77)*. ACM, 242–248. https://doi.org/10.1145/563858.563901

William Donnelly and Andrew Lauritzen. 2006. Variance Shadow Maps. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '06)*. ACM, 161–165. https://doi.org/10.1145/1111411.1111440

Hang Dou, Yajie Yan, Ethan Kerzner, Zeng Dai, and Chris Wyman. 2014. Adaptive Depth Bias for Shadow Maps. *Journal of Computer Graphics Techniques (JCGT)* 3, 4 (19 December 2014), 146–162. http://jcgt.org/published/0003/04/08/

Eric Enderton, Erik Sintorn, Peter Shirley, and David Luebke. 2010. Stochastic Transparency. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '10)*. ACM, 157–164. https://doi.org/10.1145/1730804.1730830

Viktor Kämpe, Erik Sintorn, Dan Dolonius, and Ulf Assarsson. 2016. Fast, Memory-Efficient Construction of Voxelized Shadows. *IEEE Transactions on Visualization and Computer Graphics* 22, 10 (Oct 2016), 2239–2248. https://doi.org/10.1109/TVCG.2016.2539955

Andrew Lauritzen and Michael McCool. 2008. Layered Variance Shadow Maps. In *Proceedings of graphics interface 2008*. Canadian Information Processing Society, 139–146. https://doi.org/10.20380/GI2008.18

Andrew Lauritzen, Marco Salvi, and Aaron Lefohn. 2011. Sample Distribution Shadow Maps. In *Proceedings of the 15th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '11)*. ACM, 97–102. https://doi.org/10.1145/1944745.1944761

Morgan McGuire and Michael Mara. 2017. Phenomenological Transparency. *IEEE Transactions on Visualization and Computer Graphics* 23, 5 (May 2017), 1465–1478. https://doi.org/10.1109/TVCG.2017.2656082

Christoph Peters. 2016. Free blue noise textures. http://momentsingraphics.de/?p=127. (December 2016). Blog post.

Christoph Peters and Reinhard Klein. 2015. Moment Shadow Mapping. In *Proceedings of the 19th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '15)*. ACM, 7–14. https://doi.org/10.1145/2699276.2699277

Christoph Peters, Cedrick Münstermann, Nico Wetzstein, and Reinhard Klein. 2017. Improved Moment Shadow Maps for Translucent Occluders, Soft Shadows and Single Scattering. *Journal of Computer Graphics Techniques (JCGT)* 6, 1 (2017), 17–67. http://jcgt.org/published/0006/01/03/

Victor Podlozhnyuk. 2012. Image Convolution with CUDA. (July 2012). http://developer.download.nvidia.com/compute/DevZone/C/html_x64/3_Imaging/convolutionSeparable/doc/convolutionSeparable.pdf NVIDIA whitepaper.

William T. Reeves, David H. Salesin, and Robert L. Cook. 1987. Rendering Antialiased Shadows with Depth Maps. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH '87)*. ACM, 283–291. https://doi.org/10.1145/37401.37435

Marco Salvi. 2008. *ShaderX*^{6}. Cengage Learning Inc., Chapter Rendering filtered shadows with exponential shadow maps, 257–274.

Leonardo Scandolo, Pablo Bauszat, and Elmar Eisemann. 2016. Merged Multiresolution Hierarchies for Shadow Map Compression. *Computer Graphics Forum (Proc. Pacific Graphics 2016)* 35 (2016). Issue 7. https://doi.org/10.1111/cgf.13035

Jason Stewart. 2016. SeparableFilter11. (January 2016). https://raw.githubusercontent.com/GPUOpen-LibrariesAndSDKs/SeparableFilter11/master/separablefilter11/doc/SeparableFilter11.pdf AMD sample.

Robert A. Ulichney. 1988. Dithering with blue noise. *Proc. IEEE* 76, 1 (Jan 1988), 56–79. https://doi.org/10.1109/5.3288

Robert A. Ulichney. 1993. Void-and-cluster method for dither array generation. *Proc. SPIE* 1913 (1993), 332–343. https://doi.org/10.1117/12.152707

Lance Williams. 1978. Casting Curved Shadows on Curved Surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques (SIGGRAPH '78)*. ACM, 270–274. https://doi.org/10.1145/800248.807402

Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. 2004. Light Space Perspective Shadow Maps. In *EGSR04: 15th Eurographics Symposium on Rendering*. Eurographics Association, 143–152. https://doi.org/10.2312/EGWR/EGSR04/143-151

Chris Wyman, Rama Hoetzlein, and Aaron Lefohn. 2015. Frustum-traced Raster Shadows: Revisiting Irregular Z-buffers. In *Proceedings of the 19th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D '15)*. ACM, 15–23. https://doi.org/10.1145/2699276.2699280

Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. 2006. Parallel-Split Shadow Maps for Large-Scale Virtual Environments. In *Proceedings of the 2006 ACM international conference on virtual reality continuum and its applications*. ACM, 311–318. https://doi.org/10.1145/1128923.1128975