Exponential Shadow Maps

Thomas Annen*
MPI Informatik
Germany

Tom Mertens†
Hasselt University — EDM
transnationale Universiteit
Limburg, Belgium

Hans-Peter Seidel‡
MPI Informatik
Germany

Eddy Flerackers§
Hasselt University — EDM
transnationale Universiteit
Limburg, Belgium

Jan Kautz¶
University College
London, UK

Figure 1: A backyard scene rendered with a $2k \times 2k$ shadow map and 5×5 Gauss filtering using (from left to right, statistics include mip-map memory): CSMs (22 FPS, 170 MB), VSMs (84 FPS, 42 MB), and ESMs (94 FPS, 21 MB). ESMs require $8\times$ less memory than CSMs and have less light leaking at contact points while their filtering quality is almost indistinguishable. Like CSMs, ESMs also avoid the high frequency light leaking artifacts seen with VSMs.

ABSTRACT

Rendering high-quality shadows in real-time is a challenging problem. Shadow mapping has proved to be an efficient solution, as it scales well for complex scenes. However, it suffers from aliasing problems. Filtering the shadow map alleviates aliasing, but unfortunately, native hardware-accelerated filtering cannot be applied, as the shadow test has to take place beforehand.

We introduce a simple approach to shadow map filtering, by approximating the shadow test using an exponential function. This enables us to pre-filter the shadow map, which in turn allows for high quality hardware-accelerated filtering. Compared to previous filtering techniques, our technique is faster, consumes less memory and produces less artifacts.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismColor, Shading, Shadowing and Texture; I.3.1 [Computer Graphics]: Hardware ArchitectureGraphics processors;

1 INTRODUCTION

Ever since computer graphics has emerged, great efforts have been made to provide innovative and efficient solutions to visibility and shadow computation. Unfortunately, high quality shadows still demand a lot of computational resources, which forces us to sacrifice quality in order to attain real-time performance. Sampling-based methods like Shadow Mapping [12] scale well with an increasing scene complexity, and have become the de facto standard shadow algorithm in many rendering engines. The major problem of shadow mapping is discretization artifacts due to insufficient shadow map resolution, which causes aliasing and temporal incoherence. Compared to the vast amount of papers dedicated to increase the effective shadow map resolution [3, 6, 11, 13], only very few deal with shadow map filtering to provide effective screen-space anti-aliasing.

High quality texture filtering is commonplace in today's graphics hardware. Shadow maps, which are basically depth textures, cannot be filtered in the same manner, as the shadow test has to be carried out before the filtering takes place. *Percentage Closer Filtering* (PCF) [9] respects the order of filtering and testing, and became available on graphics hardware, albeit with limited quality (bilinear filtering only). The lack of decent filtering does not only degrade image quality, but also reduces temporal coherence. One can increase the quality of PCF by taking into account more samples, (e.g., in a hardware shader) but this reduces performance dramatically. Efficient filtering of regular textures relies on *pre-filtering* the image a priori. This is done by pre-computing an image pyramid, commonly known as a mip-map. However, the shadow test cannot be carried out at that point, since the distance to the light source of the to-be-shaded point has to be taken into account as well but is not known in advance.

Recently, Variance Shadow Maps (VSMs) [5] and Convolution Shadow Maps (CSMs) [2] have been introduced. They make it possible to pre-filter the shadow map, and also support additional convolutions, and increase temporal coherence. However, VSMs are plagued by severe high frequency light leaking artifacts, and CSMs suffer from lightness problems at contact shadows but render correct shadow boundaries. In addition, the large memory footprint and high filter cost of CSMs hinder their applicability.

We present *Exponential Shadow Maps* (ESMs), a new shadow mapping method that allows for efficient (pre-)filtering. ESMs are inspired by CSMs, but use a single-term approximation, whereas CSMs use much more (typically 16) terms. This approximation assumes that the support of a filter kernel does not contain surface

*e-mail: tannen@mpi-inf.mpg.de
†e-mail:tom.mertens@uhasselt.be
‡e-mail:hpseidel@mpi-inf.mpg.de
§e-mail:eddy.flerackers@uhasselt.be
¶e-mail:j.kautz@cs.ucl.ac.uk

samples (i.e., z values) that lie beyond the distance from each screen pixel's world-space position to the light source. This approximation holds for many cases, e.g., for rendering a shadow on a large receiver, like a floor. When the assumption does not hold, we fall back to PCF, which typically only happens for a small fraction of pixels on the screen.

ESMs offer higher frame rates compared to CSMs, and a reduced memory footprint, because they use a much simpler approximation (1 term only). In addition, ESMs do not suffer from the light leaking artifacts as much as VSMs and CSMs. Finally, compared to PCF, ESMs can exploit high quality texture filtering modes, such as anisotropic filtering, which are commonly found on today's graphics hardware.

2 Related Work

A complete review of existing shadow algorithms is beyond the scope of this article and we refer the reader to Woo et al. [14] and Hasenfratz et al. [7] for excellent overviews on shadow methods. This section's focus is on rendering and anti-aliasing of hard shadows and also on current hardware shadow texture filtering.

Shadows. *Shadow Volume* rendering [4] is an object-space algorithm that constructs semi-finite volumes from object silhouettes with respect to the current light view, to determine whether a point is in shadow or not. Even though it renders accurate shadows, its performance strongly depends on the geometric complexity and the resulting rasterization overhead, and has robustness issues.

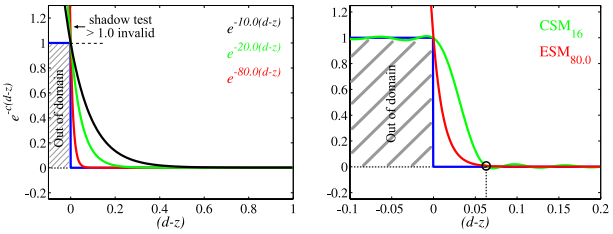
Shadow Mapping [12] is an image-based method which discretizes objects into the depth buffer, viewed from the light source. This depth buffer is then used by the shadow test in the second pass, and is typically implemented using projective texture mapping. Shadow mapping scales very well with increased scene geometry, but it is also hampered by aliasing due to limited shadow map resolution.

Previous work tackles this resolution problem, and tries to alleviate the shadow map's perspective distortion. Solutions range from generating the shadow map in post perspective space [11, 13], over constructing hierarchical and adaptive shadow maps to increase the resolution where needed [6], to irregular sampling when rendering the shadow map [1].

Anti-Aliasing. *Percentage Closer Filtering* [9] determines the coverage of a camera pixel in light space and applies the shadow test to a number of samples distributed over this region to get a filtered results. Unfortunately, the shadow test depends on the distance to the point to shade. This distance is only available at run-time and prevents pre-filtering (i.e., mip-mapping).

Variance Shadow Maps [5] is a probabilistic approach that supports pre-filtering, and additional convolutions. When the shadow map is rasterized, the z and z^2 values are stored and used during rendering to estimate the probability whether a point is in shadow or not. Their estimate only gives an upper bound of the result and produces noticeable high frequency light leaking artifacts for scenes with a high depth complexity. Recently, a variant of VSMs using summed-area tables has been published, which reduces light leaking [8]. However, the authors show that it cannot be removed completely.

Convolution Shadow Maps [2] achieve anti-aliased shadows by approximating the shadow test by a Fourier series expansion. Depending on the truncation order, z-values are converted into several basis textures. In the final rendering, pre-filtered texture samples are fetched to reconstruct a smoother shadow. CSMs have the same desirable properties as VSMs, but do not exhibit such severe light leaking artifacts. However, a reliable shadow test requires a high truncation order, which in turn increases memory consumption and filtering as well as reconstruction effort. This makes CSMs less attractive for practical applications.



(a) Shadow test domain (b) CSM_{16} vs ESM_{80}

Figure 2: ESMs assume that the domain of the shadow test is always positive $[(d-z) \geq 0]$ (a). As a result the shadow test can be approximated by an exponential decay. A larger factor c yields a better approximation. (b) shows that an ESM_{80} achieves better quality than a CSM_{16} (with an offset of -0.032). (The abscissa in (b) has been scaled to emphasize the difference.)

3 Exponential Shadow Maps

In this section, we will outline the theory behind ESMs, and discuss a practical implementation. ESMs are conceptually similar to CSMs [2], in that they also try to approximate the shadow test using an expansion.

We denote the world-space position of a camera pixel as $\mathbf{x} \in \mathbb{R}^3$ and point $\mathbf{p} \in \mathbb{R}^2$ is the same position but in shadow map space. $d(\mathbf{x})$ is the distance from $\mathbf{x}$ to the light source and $z(\mathbf{p})$ is the distance to the closed blocker seen from the light source along the direction from $\mathbf{x}$ to the light source. Boldface characters indicate positions and other variables are scalar values. CSMs [2] define the shadow test as a function of $\mathbf{x}$ using the following form:

$$s(\mathbf{x}) := f\big(d(\mathbf{x}), z(\mathbf{p})\big) \tag{1}$$

where $f(d,z)$ yields a binary result: 0 if $d > z$ and 1 otherwise. The shadow test can then be "linearized" by approximating $f(d,z)$ as a sum of separable terms:

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

where B_i are so called basis images with respect to the z-values. One of the key insights in using this expansion, is that the terms are factored into two functions which only depend on d and z, respectively. This makes it possible to pre-filter the shadow map.

3.1 Exponential Approximation

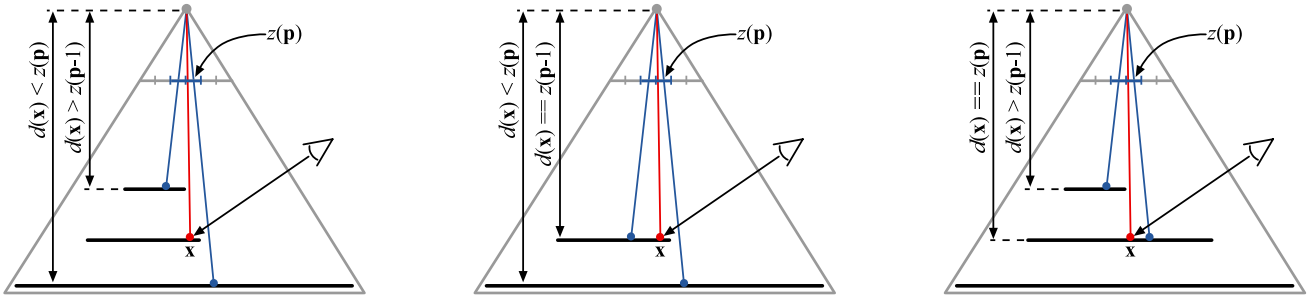
ESMs are based on a simple observation related to the domain of the shadow test, in other words, the d and z parameters in $f(d,z)$. Consider a point $\mathbf{x}$ seen by the camera, we know that the distance to the light source must be larger than or equal to the corresponding z-value read from the shadow map, because the shadow map always stores the closest surface to the light source and therefore, $d(\mathbf{x}) - z(\mathbf{p}) \geq 0$ holds. However, in practice, this is not always true. Before discussing when this happens, we will first outline how we can exploit this assumption in order to simplify the shadow test. Finally, we discuss how to deal with cases that violate the assumption.

Assuming that $d \geq z$, we can define the shadow test $f(d,z)$ as:

$$f(d,z) = \lim_{\alpha \to \infty} e^{-\alpha(d-z)}$$

which can be approximated by filling in a large positive constant c for α. This exponential function can be separated into factors depending on d and z:

$$\begin{aligned} f(d,z) &= e^{-c(d-z)} \\ &= e^{-cd} e^{cz}. \end{aligned} \tag{3}$$



(a) Failure case 1 (b) Failure case 2 (c) Correct filtering

Figure 3: ESM filtering. Red dots denote a camera sample and blue a shadow map sample. (a) shows a failure case where $\mathbf{x}$ should be darkened by 50% but its intensity is incorrectly increased because $z(\mathbf{p}) > d(\mathbf{x})$. This violates our assumption. (b) illustrates a similar case but here $\mathbf{x}$ is lit anyway and therefore doesn't cause artifacts. In both cases however a failure is detected and we enable PCF. (c) depicts a setup where our assumption holds and correct filtering is applied.

We now continue by filtering the shadow function s, yielding a value s_f. We represent the filtering operation as a convolution [2], and we fill in the exponential approximation:

$$\begin{aligned} s_f(\mathbf{x}) &= \left[w * f\left(d(\mathbf{x}), z\right)\right](\mathbf{p}) \\ &= \left[w * \left(e^{-cd(\mathbf{x})} e^{cz}\right)\right](\mathbf{p}) \\ &= e^{-cd(\mathbf{x})}\left[w * e^{cz}\right](\mathbf{p}) \end{aligned} \tag{4}$$

We see that shadow filtering now has become equivalent to applying a filter directly to the exponent-transformed depth values, which can be done beforehand.

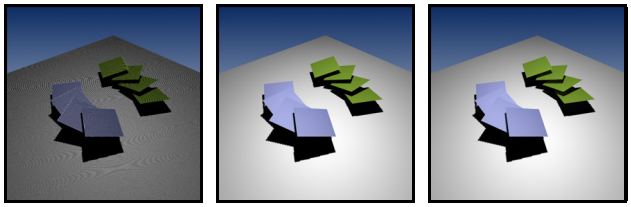
Choice of c. A higher value c results in a steeper fall-off, and thus a better approximation of the shadow test; see Figure 2(a). If c is not high enough, we will observe light leaking artifacts, similar to those reported by Annen et al. [2]. However, there is an upper bound for c, depending on the precision of the floating point representation. We empirically determined an optimal value of $c = 80$ for 32-bit floating point numbers, which is unaffected by precision issues. It even gives a better approximation than CSMs with 16 basis functions (Figure 2(b)). We abbreviate the reconstruction order M of CSMs and parameter c of ESMs as lower script values (e.g. CSM_{16} and ESM_{80}).

3.2 Violation of Assumption

Let $\Delta_{\mathbf{x}} = d(\mathbf{x}) - z(\mathbf{p})$. In the previous section, we assumed that $\Delta_{\mathbf{x}} \geq 0$. If not, the shadow test returns an arbitrarily large number as the new expansion does not converge to 1.0 but grows exponentially. We will discuss how this affects the results in the following two cases.

Without Filtering. We first analyze the case when the shadow map is not filtered (nearest neighbor sampling). In unshadowed areas, $d(\mathbf{x})$ should ideally be equal to $z(\mathbf{p})$. However, the precision of the shadow map is finite due to the limited spatial and numerical resolution. Consequently, $d(\mathbf{x})$ will only be approximately equal to $z(\mathbf{p})$, especially for slanted surfaces. In standard shadow mapping, this leads to the well-known "shadow acne" problem. In our case, the reduced precision may incur negative $\Delta_{\mathbf{x}}$ values, yielding an overflow of the shadow function (i.e., a value larger than one). To overcome this problem, we can simply clamp the exponential to one.

With Filtering. Similar to CSMs and VSMs, ESMs can be filtered prior to using it for rendering the actual shadows. However, z-values under the support of the filter will not necessarily be smaller than a given $d(\mathbf{x})$. For instance, this happens at slanted surfaces, or possibly at depth discontinuities. Consequently, we will get an erroneous filter response due to an overflow of the exponential.



(a) PCF (F=0, U=0) (b) PCF (F=2, U=1) (c) ESM (F=0, U=0)

Figure 4: Shadow acne and polygon offset (F=factor, U=units in OpenGL format). (a) without polygon offset numerical imprecision generates incorrect self-shadowing. (b) ESMs are less prone to numerical inaccuracies because the exponential decay is not steep enough over such small distances.

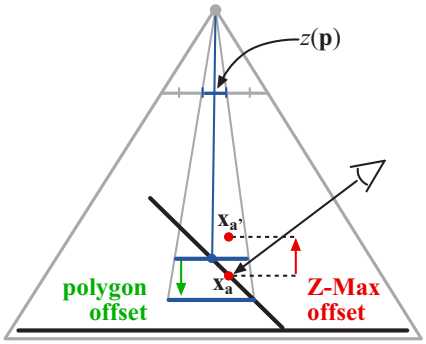
Clamping the values for each individual sample, would require us to resort to a PCF-style method, which defeats the purpose of prefiltering. Figure 3(a,b) illustrates two common failure cases when $\Delta_{\mathbf{x}}$ becomes negative.

Frequency of Violation. In most cases, the filter support contains z-values that are smaller than $d(\mathbf{x})$; see Figure 3(c). When sampling points that are far away from shadow borders, the z-values are either all blockers (fully in shadow), or represent the sampled surface itself (fully illuminated) and our assumption holds. This occurs quite often, as most of the pixels are either fully in shadow or fully illuminated. Furthermore, the assumption holds for large receivers (e.g., a floor), in which case all blockers lie in front of the receiver, w.r.t. the light source.

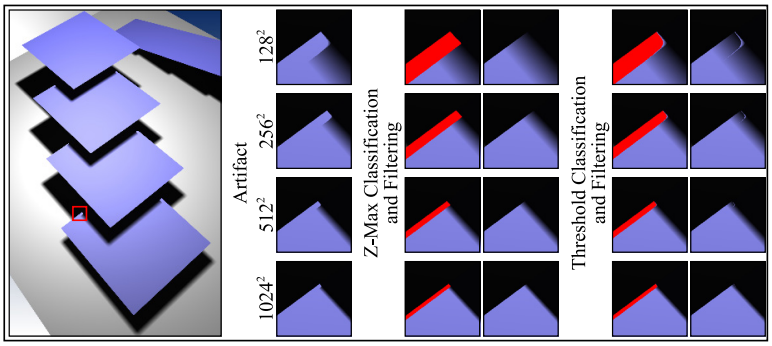
Even if the assumption is violated, the effect may not be visible. For example, unoccluded slanted surfaces (see Fig. 5(a)), may be sampled above the stored z-value (denoted by x_a in the figure), and therefore overflow. However, this overflow can be easily clamped to one (i.e., fully visible), not introducing artifacts. Since this surface is actually supposed to be fully visible, the violation goes unnoticed.

In Section 3.3, we introduce two methods to classify pixels where the assumption is violated. For these pixels, we (can) fall back to a custom filtering solution in order to avoid artifacts.

Polygon Offset. Regular shadow mapping suffers from the so-called "shadow acne" artifact, which refers to erroneous self-shadowing due to precision issues and is illustrated in Fig. 4(a) (Note that we describe the polygon offset in OpenGL terms where an offset o is computed by $o = m \cdot factor + r \cdot units$, where m is the maximum polygon depth slope and r is the smallest value that ensures a resolvable offset). This can be solved by slightly offsetting the z-values away from the light source (see Fig. 3(d) and 4(b)). In



(a) Polygon and Z-Max Offset



(b) Classification

Figure 5: (a) describes the difference between polygon offset and the Z-Max offset which is important during failure classification. (b) ESM failure classification and fall back to PCF. We illustrate the artifacts and the difference in Z-Max and Threshold classification, and their overall quality for various shadow map resolutions. Timings for both classifications are listed in Table 2.

practice, polygon offsetting is not required for ESMs, because the exponential does not decay fast enough over such small distances. However, we still employ an additional offsetting but for another reason, namely for failure classification, which we will be detailed in the next section.

3.3 Failure Classification and Fall Back Solution

The previous section explained in which situations our initial assumption of $d(\mathbf{x}) - z(\mathbf{p}) \geq 0$ will be violated. This section presents two methods to check for such failure cases, and how to fix them. If a given pixel is classified as invalid, we fall back to a customized filtering which we refer to as *custom filtering* or *custom PCF*. For performance reasons we opt to only use a 2×2 filter kernel similar to the bilinear version of PCF implemented in hardware [9] which we cannot use as we don't want to use the shadow map in addition to an 32-bit ESM (this would increase the memory consumption by 24- or even 32-bit times the shadow map resolution). Fortunately, we can simulate a filtered shadow test simply by evaluating the ESM at the 4 nearest neighbors followed by a bilinear interpolation on the clamped results.

Z-Max Classification. This approach relies on an additional texture in which we maintain the maximum z-values in a given neighborhood for the current shadow map. When we convert the z-values into the exponential basis, we simultaneously copy the z-values into the base level of the Z-Max texture. A max-filter is then used to build a mip-map structure, effectively storing maximum z-values for mip-mapped neighborhoods.

Classification of the pixel $\mathbf{x}$ works as follows. First $d(\mathbf{x})$ is computed and the z_{max} for the current filter kernel is fetched from the mip-mapped Z-Max texture (an appropriate LOD is selected to match the filter kernel). Checking if $d(\mathbf{x}) < z_{max}$ gives a conservative answer whether the assumption is violated for $d(\mathbf{x})$ or not.

To avoid misclassification of fully lit surfaces we have to add a small offset to $d(\mathbf{x})$. Note that this is problem is similar to the original polygon offsetting but it works in the exact opposite direction (see Fig. 5(a). We want z_{max} to be slightly smaller so that a lit surface is not incorrectly flagged. The effect of the offset can be seen in Fig. 6(a).

Threshold Classification. A second option to check if our assumption holds for a given pixel is to first evaluate the ESM result and then check if it exceeds $1+\varepsilon$ where ε is a given threshold. This essentially checks if a large ESM value contributed to the result indicating that the assumption is violated (then large values occur to exponential growth depicted in Fig. 2). The effect of Thresholding compared to the Z-Max method is depicted in Fig. 6(b).

(a) Offset: 0.001 (b) Threshold: 0.020

Figure 6: (a) The influence of the offset, which is added to $d(\mathbf{x})$ for Z-Max classification. (b) The threshold on intensities shown in (b). (a) shows that the offset is underestimated and needs to be larger.

3.4 Discussion

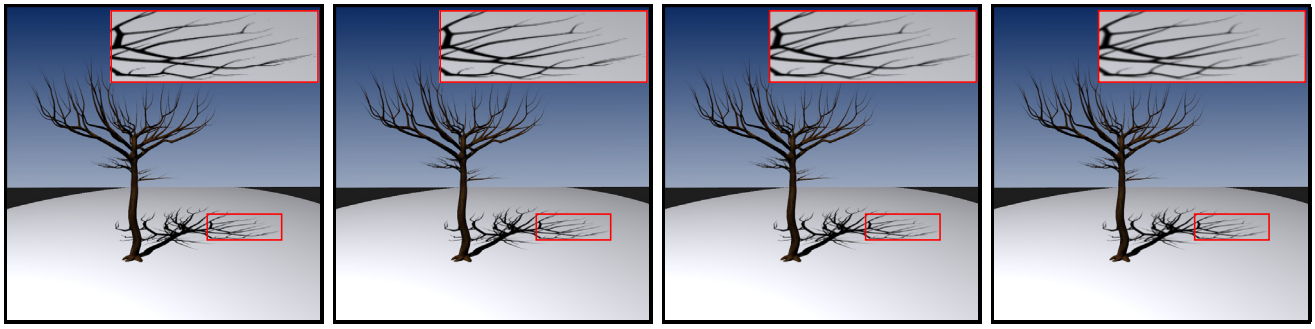
This section discusses the difference between our classification schemes and addresses issues regarding the overall temporal coherence of ESMs.

Z-Max Classification is a conservative method and achieves more accurate results but requires an additional texture map, which needs to be down-sampled using a max-filter. In case where additional convolutions are applied to the ESM, the Z-Max texture also requires a max-filter of the same size and therefore reduces the overall frame rate.

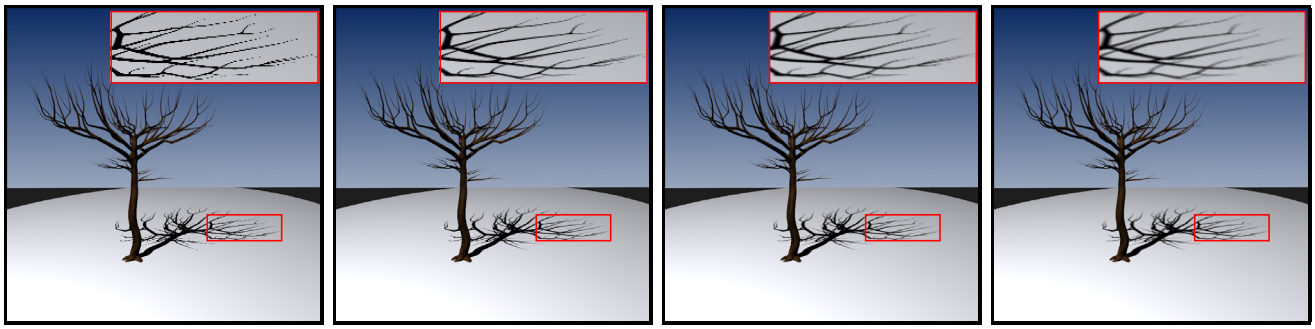
Threshold Classification does not require any additional resources and renders efficiently, but may suffer from small artifacts due to *false negative* classification errors. This can occur because thresholding is not a safe method to determine if the initial assumption is valid for all pixels within the filter kernel, as only the resulting filtered ESM value is checked. The visual quality and classification result is shown in Figure 5(b).

Temporal Coherence for ESMs is, independent of the classification, superior to regular shadow mapping methods. However, due to the assumption we make and the resulting need for custom filtering, ESMs exhibit slightly less temporal coherence in a small neighborhood of pixels as CSMs can achieve. As this usually only happens for a very small amount of screen space pixels we did not recognize noticeable differences between ESMs and CSMs.

Both quality and performance comparison show that the benefit of Z-Max decreases with texture size whereas its performance penalty increases at the same time. According to this observation



(a) ESM trilinear (248 FPS) (b) ESM 3x3 (182 FPS) (c) ESM 5x5 (155 FPS) (d) ESM 7x7 (136 FPS)



(e) PCF (409 FPS) (f) PCF 4x4 (224 FPS) (g) PCF 6x6 (142 FPS) (h) PCF 8x8 (95 FPS)

Figure 7: Quality and performance comparison between regular PCF with multiple samples and ESMs with additional convolutions. We compare bilinear, 4×4, 6×6, and 8×8 PCF versus trilinear ESMs, and trilinear ESMs with additional convolutions applied. At least 36 samples are necessary for PCF to match regular trilinear ESMs and 64 samples are required to achieve similar quality to ESMs with an additional $5x5$ Gauss filter.

we opted to use thresholding for all results.

It should be noted that our current custom PCF cannot remove artifacts that occur when additional convolutions are used, as the custom filter kernel size would be too large to be applicable in real-time applications.

4 Implementation

Integrating ESMs into an existing rendering pipeline is straightforward. To generate exponential basis images we use a 32-bit floating point depth texture available through the *NV_depth_buffer_float* OpenGL extension by writing $exp(cz)$ instead of regular z values. In our implementation we also use a linear depth buffer. Any additional convolution is applied to the exponential basis image in the same manner as in [2]. Rendering shadows with ESMs is now trivial. Instead of performing an explicit shadow test against d and z we simply evaluate Equation 3. Failure cases are detected by either one of the methods described in Section 3.3 and should incorporate the current polygon offset for shadow map generation for faithful detection.

5 Results

This section demonstrates the quality and efficiency of Exponential Shadow Maps. All examples have been implemented in OpenGL 2.0 and rendered on a Dual-Core AMD Opteron PC with 2.6GHz and 2.75GB RAM equipped with an NVIDIA GeForce 8800 GTX graphics card. We have used the Thresholding approach as failure classifier. Rendering performance for various shadow algorithms are compared in Table 1. All memory statistics already contain the mip-map overhead (a factor of 1.3).

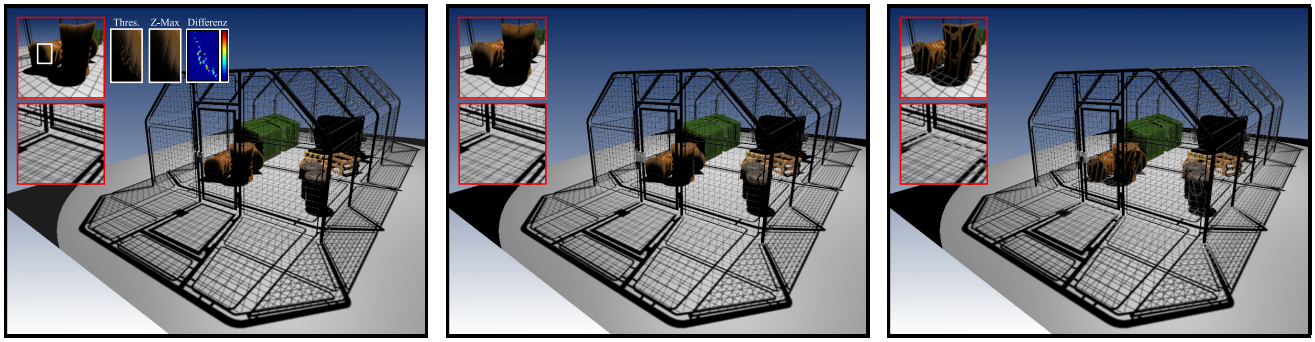
Figure 1 compares ESMs, VSMs, and CSMs in terms of quality and memory consumption. The closeups show that ESMs perform better at contact shadows than CSMs while avoiding light leaking of VSMs. The latest variant of VSMs, Summed-Area VSMs [8] also reduce light leaking but cannot completely avoid it and still have higher memory cost.

In Figure 7 we evaluate how many samples an adaptive PCF filter would have to use to achieve anti-aliasing of similar quality as ESMs provide. To reach regular trilinear ESM filtering quality, PCF has to use at least 16 or up to 36 samples which reduces the framerate significantly compared to ESMs. To match ESMs with an additional $5x5$ Gauss convolution PCF needs at least 64 samples.

Figure 8 demonstrates that the filtering quality between ESMs and CSMs is virtually not distinguishable especially for scenes with

$C = no$	$ESM-T.$	$ESM-Z.$	VSM	CSM
$S:512^2$	145 fps	132 fps	152 fps	83 fps
$S:1024^2$	140 fps	123 fps	140 fps	68 fps
$S:2048^2$	119 fps	101 fps	106 fps	40 fps
$C = 3x3$	$ESM-T.$	$ESM-Z.$	VSM	CSM
$S:512^2$	141 fps	127 fps	149 fps	75 fps
$S:1024^2$	132 fps	113 fps	131 fps	56 fps
$S:2048^2$	102 fps	78 fps	87 fps	27 fps
$C = 7x7$	$ESM-T.$	$ESM-Z.$	VSM	CSM
$S:512^2$	138 fps	119 fps	146 fps	71 fps
$S:1024^2$	124 fps	100 fps	125 fps	46 fps
$S:2048^2$	86 fps	61 fps	78 fps	19 fps

Table 1: Frame rates for the backyard scene from Figure 1. We compare ESMs (Thresholding and Z-Max) against VSMs and CSMs. Measurements include varying shadow map resolution and additional convolution (Gauss) kernel sizes. *S* and *C* denote the shadow map and convolution size.



(a) ESM (66 FPS, 21 MB) (b) CSM (21 FPS, 170 MB) (c) VSM (60 FPS, 42 MB)

Figure 8: A complex fence scene (365k faces) rendered with a $2k \times 2k$ shadow map and an additional $5x5$ Gauss filter. ESMs achieve a quality which is visually equivalent to CSMs while performing better at contact shadows (see lower closeups). At the same time they don't suffer from light leaking as much as VSMs (top closeups). The ESM result was rendered with Thresholding (like all other results in this section). For a comparison between Thresholding and Z-Max classification see close-ups in (a).

high depth complexity owing to the surrounding fence. This examples illustrates the quality that ESMs achieve with $8\times$ less memory and a significantly better performance. We also compare ESMs against VSMs showing less light leaking and better performance.

Figure 10 visualizes the impact additional convolutions have on the failure classification. The larger the filter kernel the more often our assumption fails and we have to perform custom filtering for all pixels indicated in red. Table 3 lists the exact numbers (for this measurement anti-aliasing was turned off).

Table 2 gives the performance timings (for Figure 5(b)) regarding the failure detection and offers information for choices when one or the other classification approach is more applicable depending on the shadow map size.

$C = no$	$S:128^2$	$S:256^2$	$S:512^2$	$S:1024^2$
$Z-Max$	410 fps	393 fps	374 fps	347 fps
$Threshold$	569 fps	556 fps	527 fps	438 fps
$C = 3x3$	$S:128^2$	$S:256^2$	$S:512^2$	$S:1024^2$
$Z-Max$	369 fps	357 fps	345 fps	273 fps
$Threshold$	547 fps	536 fps	495 fps	374 fps
$C = 5x5$	$S:128^2$	$S:256^2$	$S:512^2$	$S:1024^2$
$Z-Max$	358 fps	351 fps	338 fps	237 fps
$Threshold$	544 fps	527 fps	474 fps	342 fps

Table 2: Failure detection performance for Z-Max and Thresholding ($800x800$ viewport) for the scene from Figure 5(b).

$Failure$	$No\ Conv.$	$3x3$	$5x5$	$7x7$
$Z-Max$	3.0%	6.1%	7.9%	9.3%
$Threshold$	2.8%	4.9%	6.1%	7.1%

Table 3: Failure classification for the backyard scene from Figure 1. Even for an additional $7x7$ convolution only 7.1% (or 9.3% for Z-Max test) of the screen-space pixels require special treatment.

A crucial situation for ESMs is minification where the filtering kernel can be very large and thus the probability increases that the z-values within the kernel are larger than the current $d(\mathbf{x})$. Figure 9 demonstrates how custom PCF avoids artifacts. We compare ESMs with regular trilinear filtering without custom PCF, ESMs with custom PCF, and ESMs with anisotropic filtering only and again no custom PCF.

It is interesting to note, that the fixed 2×2 PCF filter is sufficient to remove visible artifacts, which is most likely due to the fact that

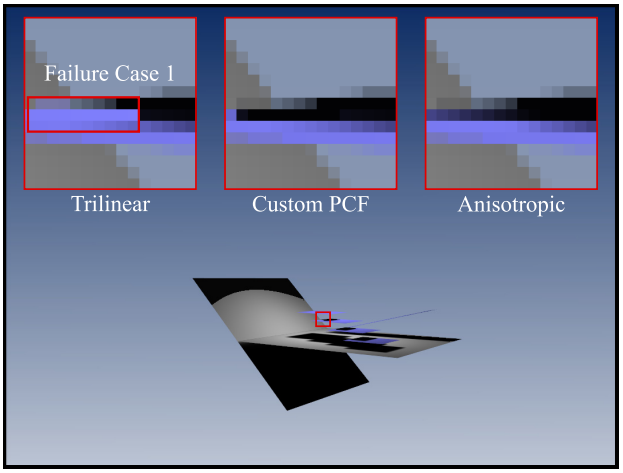


Figure 9: Anisotropic filtering. (a) without custom filtering camera pixels are incorrectly lit (red rectangle), (b) trilinear with custom PCF can prevent artifacts, and $10\times$ anisotropic filtering without custom filtering often handles such failure cases properly.

only very few pixels are filtered with PCF. Furthermore, the figure shows that when anisotropic filtering is turned on, the more expensive custom filtering is not really necessary. However, in situations where the filter kernel becomes very large, our custom PCF as well as anisotropic filtering may yield slightly less temporal coherence than the original CSM algorithm, which is due to our limiting the number of samples of our filter to 2×2 samples.

6 Conclusions

We have presented Exponential Shadow Maps, a new exponential expansion for shadow map filtering. We have introduced a simple assumption which simplifies the problem of shadow map filtering drastically and allows a single term expansion saving valuable resources in terms of memory and computational cost. Usually, this assumption holds for the vast majority of screen-space pixels. The pixels for which this assumption does not hold are easy to detect and we have presented two alternative solutions to such failure scenarios. We believe that ESMs are beneficial for real-time applications such as games where resources are limited. In the future, we would like to investigate other expansions and also use improved filtering to further increase the quality of ESMs.

(a) No conv. (3.0%) (b) 3x3 Gauss (6.1%) (c) 5x5 Gauss (7.9%) (d) 7x7 Gauss (9.3%)

(e) No conv. (2.8%) (f) 3x3 Gauss (4.9%) (g) 5x5 Gauss (6.1%) (h) 7x7 Gauss (7.1%)

Figure 10: Failure case classification. (a)–(d) uses Z-Max and (e)–(h) Thresholding. An increasing filter kernel size also increases the number of pixels for which ESMs cannot reconstruct a valid shadow test. For all red pixel we perform custom PCF filtering. The ratio of the total number of screen-space pixels (800×800) and failure cases is given in brackets.

ACKNOWLEDGEMENTS

We'd like to refer the reader to the following concurrent work "Rendering Filtered Shadows with Exponential Shadow Maps" by Salvi [10].

REFERENCES

[1] T. Aila and S. Laine. Alias-free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004*, pages 161–166. Eurographics Association, 2004.

[2] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz. Convolution shadow maps. In *Rendering Techniques 2007*, volume 18 of *Eurographics / ACM SIGGRAPH Symposium Proceedings*, pages 51–60. Eurographics, June 2007.

[3] S. Brabec, T. Annen, and H.-P. Seidel. Practical shadow mapping. *Journal of Graphics Tools*, 7(4):9–18, 2003.

[4] F. C. Crow. Shadow algorithms for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH '77)*, pages 242–248, 1977.

[5] W. Donnelly and A. Lauritzen. Variance shadow maps. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165, New York, NY, USA, 2006. ACM Press.

[6] R. Fernando, S. Fernandez, and K. Bala. Adaptive shadow maps. In *Proceedings of SIGGRAPH '01*, *Computer Graphics* Proceedings, Annual Conference Series, pages 387–390. ACM SIGGRAPH, 2001.

[7] J.-M. Hasenfratz, M. Lapierre, N. Holzschuh, and F. Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum (Proceedings of Eurographics '03)*, 22(3), 2003.

[8] A. Lauritzen. *Summed-Area Variance Shadow Maps*. Addison-Wesley, 2007.

[9] W. T. Reeves, D. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. *Computer Graphics (Proceedings of SIGGRAPH '87)*, pages 283–291, 1987.

[10] M. Salvi. Rendering filtered shadows with exponential shadow maps. In *ShaderX 6.0 – Advanced Rendering Techniques*. Charles River Media, 2008.

[11] M. Stamminger and G. Drettakis. Perspective shadow maps. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, pages 557–562, 2002.

[12] L. Williams. Casting curved shadows on curved surfaces. *Computer Graphics (Proceedings of SIGGRAPH '78)*, pages 270 – 274, 1978.

[13] M. Wimmer, D. Scherzer, and W. Purgathofer. Light space perspective shadow maps. In *Proceedings of the 2nd EG Symposium on Rendering*, Springer Computer Science. Eurographics, Eurographics Association, 2004.

[14] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Computer Graphics & Applications*, pages 13–32, 1990.