Layered Variance Shadow Maps

Andrew Lauritzen[*] Michael McCool[†]

Computer Graphics Lab, David R. Cheriton School of Computer Science, University of Waterloo

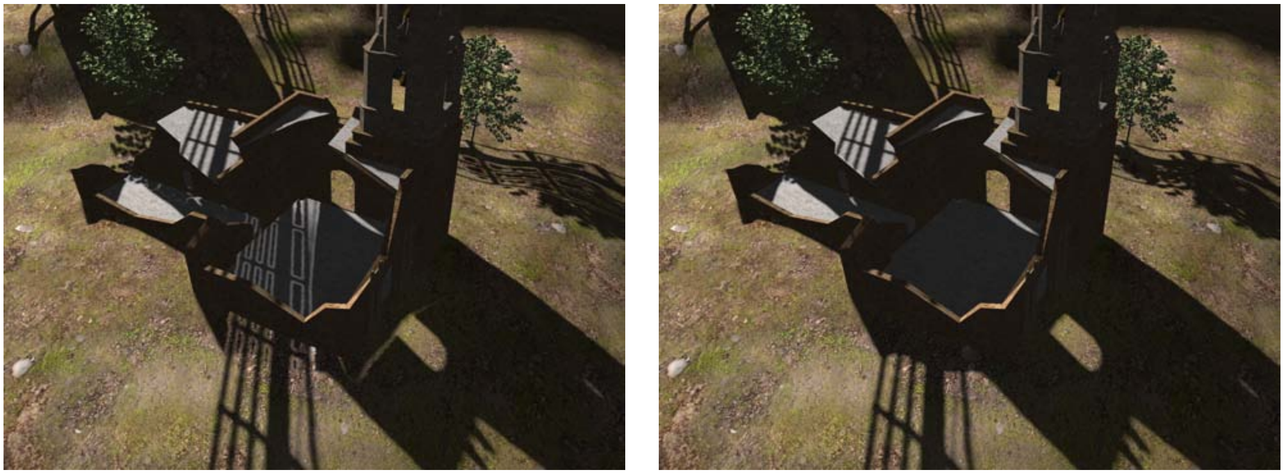


Figure 1: Comparison of regular variance shadow maps (left) with our technique (right). The scene demonstrates a difficult case for shadow filtering algorithms. Our algorithm removes the light bleeding artifacts (seen in the left image where multiple shadows overlap) by partitioning the light's depth range into multiple layers.

Abstract

Shadow maps are commonly used in real-time rendering, but they cannot be filtered linearly like standard color, resulting in severe aliasing. Variance shadow maps resolve this problem by representing the depth distribution using moments, which *can* be linearly filtered. However, variance shadow maps suffer from "light bleeding" artifacts and require high-precision texture filtering hardware.

We introduce *layered variance shadow maps*, which provide simultaneous solutions to both of these limitations. By partitioning the shadow map depth range into multiple layers, we eliminate all light bleeding between different layers. Using more layers increases the quality of the shadows at the expense of additional storage. Because each of these layers covers a reduced depth range, they can be stored in lower precision than would be required with typical variance shadow maps, enabling their use on a much wider range of graphics hardware. We also describe an iterative optimization algorithm to automatically position layers so as to maximize the utility of each.

Our algorithm is easy to implement on current graphics hardware and provides an efficient, scalable solution to the problem of shadow map filtering.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1 Introduction

Shadow maps [12] are the most commonly used shadowing technique in real-time applications, providing several advantages over shadow volumes [3]. Most notably, shadow maps can be queried at arbitrary locations, they work with anything that can be rasterized including alpha tested geometry, and they are much less sensitive to geometric complexity than shadow volumes. Like most textures, however, they suffer from aliasing if not properly filtered. Unfortunately, common linear filtering algorithms such as mipmapping and anisotropic filtering are inapplicable to ordinary shadow maps, which require a non-linear depth comparison per sample.

Variance shadow maps (VSMs) [4, 7] provide a solution to this problem by representing a probability distribution of depths at each shadow map texel. To achieve constant space usage, the distributions are approximated using their first two moments. The visibility function over an arbitrary filter region can be estimated using Chebyshev's Inequality, which yields an upper bound on the amount of light reaching the fragment being shaded. In the case of a single planar occluder and a single planar receiver the reconstruction is exact [4], motivating its direct application to shading.

However, because only two moments are stored, complex visibility functions cannot be reconstructed perfectly. This leads to regions that should be in shadow being lit, an artifact known as "light bleeding". Storing more moments could eliminate the problem, but estimating the visibility function becomes very computationally intensive as more moments are used. Furthermore higher-order moments are numerically unstable, making them difficult to use in practice. Even the second moment requires fairly high precision storage and filtering to avoid numeric problems with large light ranges. Graphics hardware that supports 32-bit floating point filter-

[*] e-mail: atlaurit@cgl.uwaterloo.ca

[†] e-mail: mmccool@cgl.uwaterloo.ca

ing is sufficent, but hardware with less precision (16-bit for example) often is not. This limits the applicability of VSMs to high-end hardware.

Rather than using more moments to improve the quality of VSMs, we can instead alter the depth metric. The key observation is that the visibility test is a function of the depth *ordering* in the scene, not the specific depth *values*. We can thus choose any monotonic warp of the depth metric and still obtain an upper bound on the visibility using Chebyshev's Inequality as with standard VSMs.

Moreover we can use any number of uniquely chosen warps. Each warp will produce a different upper bound on the visibility function, some tighter than others. We use this fact to partition the light's depth range into different *layers*, each of which has a unique warping function. We choose these warps so that for a given depth d, we know in advance which layer will provide the best approximation. With this approach, only a single texture sample from one layer is required for each fragment while shading, allowing us to scale to a large number of layers.

A remaining problem is where to place the layer boundaries to eliminate the most light bleeding. While uniformly or manually placed layer boundaries may be sufficient for many applications, it is desirable to have an automated method available. We describe an iterative image-space optimization algorithm based on Lloyd relaxation [5] in Section 4.

In summary, layered variance shadow maps (LVSMs) generalize variance shadow maps and provide a scalable way to increase shadow quality beyond what is possible with standard VSMs. Additionally, LVSMs can be used with a wider range of graphics hardware than VSMs since they do not require high-precision texture filtering.

2 Related Work

Shadow maps [12] are an efficient method of computing shadows in general scenes. Unfortunately, they suffer from several forms of aliasing. There are two orthogonal bodies of research that address these aliasing problems: projection optimization and shadow map filtering. Lloyd [8] gives an excellent survey and analysis of various warping an partitioning algorithms that aim to improve the shadow projection. We address the filtering problem, but projection optimization algorithms can be used in conjunction with our approach to obtain high quality shadows.

Percentage-closer filtering (PCF) [10] provides a way to filter shadow maps by observing that the *results* of many depth comparisons over a filter region should be averaged, rather than the depth values themselves. This method does not support pre-filtering, however, and so for large filter sizes it is very expensive. Additionally, PCF does not consider the receiver geometry within the filter region, which causes significant biasing problems.

Deep shadow maps [9] store a distribution of depths for each shadow map texel and allow pre-filtering. Unfortunately general deep shadow maps require a variable amount of storage per pixel, making them unsuitable for implementation on current graphics hardware. Furthermore averaging two distributions is non-trivial, which complicates filtering.

Opacity shadow maps [6] and convolution shadow maps [1] represent the visibility function with respect to a basis to allow linear filtering. These algorithms are limited in their ability to represent discontinuities in the visibility function, and thus suffer from light bleeding near all occluders. Properly rendering any reasonably sized scene requires a huge number of layers (in the case of opacity shadow maps) or coefficients (in the case of convolution shadow maps), which makes these solutions impractical for real-time rendering of complex scenes. Furthermore, convolution shadow maps need to sample *all* of the basis coefficients for a given shadow map texel, which scales poorly as the number of coefficients increases. In contrast, layered variance shadow maps require only a single texture sample per shaded pixel, just like variance shadow maps.

Our algorithm also uses a piecewise representation of the visibility function, but depth tests within each piece are resolved in the same way as with standard variance shadow maps (VSMs) [4]. Thus like VSMs our algorithm has no problem reconstructing the visibility function in the case of a single planar occluder and receiver. Unlike VSMs, however, we avoid multi-occluder light bleeding artifacts by partitioning the depth range into multiple pieces.

3 Algorithm Overview

The algorithm is built on top of variance shadow maps [4, 7], which we will review briefly here.

We begin by rendering both depth and squared depth into a two-component variance shadow map. The VSM is sampled while shading a fragment to produce the moments M_1 and M_2 of the depth distribution $F(x)$ over the texture filter region, defined as:

$$\begin{aligned} M_1 &= E[x] = \int_{-\infty}^{\infty} xF(x)dx \\ M_2 &= E[x^2] = \int_{-\infty}^{\infty} x^2F(x)dx \end{aligned} \quad (1)$$

From these we compute the mean and variance of the distribution:

$$\begin{aligned} \mu &= E[x] = M_1 \\ \sigma^2 &= E[x^2] - E[x]^2 = M_2 - M_1^2 \end{aligned} \quad (2)$$

We then apply the one-tailed version of Chebyshev's Inequality to estimate the percentage of light reaching a surface at depth t. In particular for the some random variable x drawn from a depth distribution recovered from the filtered variance shadow map, we have

$$P(x \geq t) \approx p(t) = \begin{cases} 1 & \text{if } t \leq \mu \\ \frac{\sigma^2}{\sigma^2+(t-\mu)^2} & \text{if } t > \mu \end{cases} \quad (3)$$

Finally, we use $p(t)$ to attenuate the amount of light reaching the fragment due to shadowing. Usually it is desirable to clamp the variance to some minimum value σ^2_{min} to avoid any numeric problems when $t \approx \mu$.

3.1 Light Bleeding

Variance shadow maps are simple and efficient, but they suffer from light bleeding. While Chebyshev's Inequality gives an upper bound on $P(x \geq t)$, there is no guarantee that the upper bound is a good approximation. As an example, consider the situation shown in Figure 2.

Let objects A, B and C be at depths a, b and c respectively. Note that only objects A and B will be represented in the filter region since object C is not visible from the light. Thus if we are shading a fragment at the center of the outlined region, we will recover (1) the moments

$$\begin{aligned} M_1 &= \frac{a+b}{2} \\ M_2 &= \frac{a^2+b^2}{2} \end{aligned}$$

Then from (2), we compute:

$$\begin{aligned} \mu &= \frac{b+a}{2} \\ \sigma^2 &= \frac{(b-a)^2}{4} \end{aligned}$$

Now consider shading the fragment on the surface of object C.

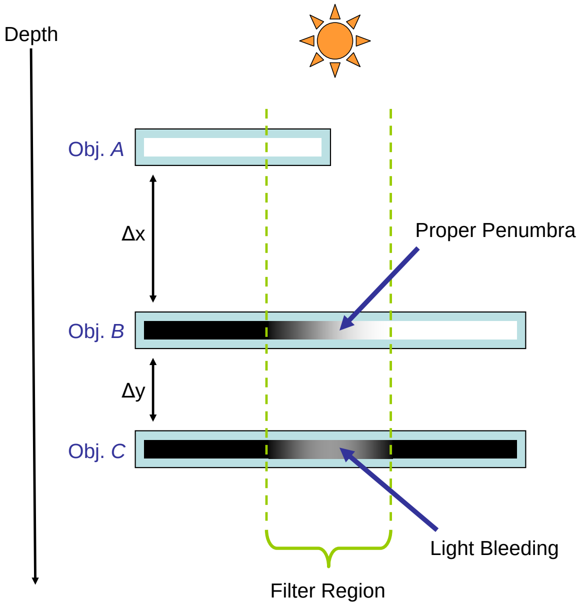


Figure 2: A simple situation in which light bleeding occurs. Object A casts a proper shadow onto object B. Object C should be fully occluded by object B, but some of the penumbra from object A's shadow "bleeds" through incorrectly.

In terms of $\Delta x = b - a$ and $\Delta y = c - b$ (shown in Figure 2), (3) yields the visibility function

$$\begin{aligned} p(\Delta y) &= \frac{\frac{1}{4}\Delta x^2}{\frac{1}{4}\Delta x^2 + (\Delta y + \frac{1}{2}\Delta x)^2} \\ &= \frac{\frac{1}{4}\Delta x^2}{\frac{1}{2}\Delta x^2 + \Delta x\Delta y + \Delta y^2} \end{aligned} \qquad (4)$$

Therefore for a given Δx, $p(\Delta y)$ falls off like $O(\frac{1}{\Delta y^2})$.

The correct visibility function, however, equals zero for all $\Delta y > 0$ since everything further from the light than object B is fully occluded by it. This is exactly the discrepancy that causes light bleeding: the $O(\frac{1}{\Delta y^2})$ "tail" of Chebyshev's Inequality.

This is not a problem with the inequality itself. We simply have not kept enough information about the depth distribution to disambiguate more complex situations.

Figure 3 shows an example of light bleeding in practice. In this example, object A is the tree, object B is the bench and object C is the ground. Note that the ratio of Δx (tree to bench) to Δy (bench to ground) is high, leading to significant artifacts.

To eliminate light bleeding, we need to store more information. As discussed in Section 1, using more moments is undesirable as higher-order moments are numerically unstable. Instead, we note that warping the depth distribution in different ways before computing the moments can provide additional information about the distribution. This information can be used to reduce or eliminate light bleeding.

3.2 Depth Warping

Recall that we are trying to approximate $P(x \geq t)$. This probability does not depend on the actual depth values stored in the VSM, but only on their ordering. Chebyshev's Inequality, however, *does* depend on the actual depth values, but is guaranteed to provide an upper bound regardless. Thus by using different monotonic warps of the depth values we can obtain different upper bounds for $P(x \geq t)$, some tighter than others.

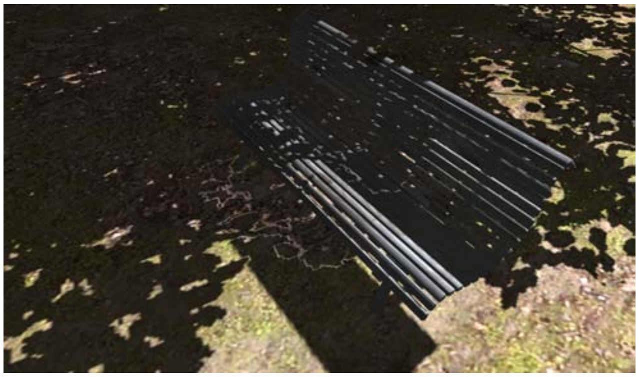


Figure 3: Light bleeding artifacts: the penumbra of the tree shadow is improperly bleeding through the bench.

Consider for example the following warp φ_c, again using the example from Figure 2 (recall that c is the depth of object C):

$$\varphi_c(t) = \begin{cases} 0 & \text{if } t < \text{c} \\ 1 & \text{if } t \geq \text{c} \end{cases} \qquad (5)$$

Proceeding as before:

$$\begin{aligned} M_1 &= \frac{\varphi_c(a) + \varphi_c(b)}{2} = 0 \\ M_2 &= \frac{\varphi_c(a)^2 + \varphi_c(b)^2}{2} = 0 \\ \mu &= 0 \\ \sigma^2 &= 0 \end{aligned}$$

Now when we evaluate $p(t)$ for objects B and C:

$$\begin{aligned} p(\varphi_c(b)) &= p(0) = 1 \\ p(\varphi_c(c)) &= p(1) = \frac{\sigma^2}{\sigma^2 + (1-\mu)^2} = 0 \end{aligned}$$

This time the visibility for object C is correct. Although the value for object B is no longer correct (it should be $\frac{1}{2}$), it still provides an upper bound, albeit a trivial one. The function $\varphi_c(t)$ is an example of a "perfect" warp for evaluating the shadowing of an object at depth c. Such a function warps Δx to 0 so that all light bleeding is eliminated as per (4).

In the next section, we describe how to choose warps that result in an approximation that is at least good as standard VSMs, but can also be scaled up to reduce or eliminate light bleeding.

3.3 Layers

While any monotonic warping function is a potential candidate, roughly linear ones are desirable, so filtering works as expected. Assuming that the original depth metric has a good distribution of precision along the depth range, linear warps will maintain that distribution. There are also several other advantages to linear warps that we discuss later in this section.

Additionally, since we do not know the points and filter regions at which we will need to evaluate the visibility function when rendering the shadow map, we need to be able to reconstruct the function over the entire depth range with good accuracy. To this end, we split the scene into multiple depth *layers* (in light space) and define a warping function for each layer that we use to approximate the visibility for fragments that fall into the layer's depth range.

Thus for each layer L_i covering an interval $[p_i, q_i]$ we define the following warp:

$$\varphi_i(t) = \begin{cases} 0 & \text{if } t \leq p_i \\ \frac{t-p_i}{q_i-p_i} & \text{if } p_i < t < q_i \\ 1 & \text{if } q_i \leq t \end{cases} \tag{6}$$

It is also necessary to find a consistent minimum variance for each layer. Because we are using a linear warping, these values can be computed from the global $\sigma^2_{\min}$ as follows:

$$(\sigma^2_{\min})_i = \frac{\sigma^2_{\min}}{(q_i - p_i)^2} \tag{7}$$

This new warp has light bleeding reduction properties similar to the $\varphi_c(t)$ function defined in the previous section. Specifically, if all occluders have depths at most p_i and the receiver has depth greater than p_i, the visibility function will be computed exactly. Additionally, occlusion within a single layer is resolved in the same manner as with variance shadow mapping.

For a geometric interpretation of what is happening, consider the silhouette cast by some set of occluders onto some receiver. Imagine flattening this silhouette onto a plane sitting above the receiver, turning the occluders into a simple occlusion "cutout". The shadow cast on the receiver will be identical, assuming a point source, but will be easier to compute since we have transformed the problem into one of a single occluder and receiver, which VSMs handle perfectly.

When shading a surface, we could theoretically sample all of the layers, compute Chebyshev's Inequality and take the minimum value over all layers. However with our layered warping functions we only need to sample a single layer: the layer L_j that contains the receiver surface that we are shading (at depth t). It can be shown that for all layers closer to the light ($i < j$), the visibility function will be at least as high (lit) since as we move closer to the light we can only remove occluders. Conversely for all layers further from the light ($i > j$), the warp $\varphi_i(t) = 0$ and so the probability $p(\varphi_i(t)) = 1$ since $\mu \geq 0$. Therefore, for a given depth we only need to sample a single layer.

Another advantage of the layers approach is that each of the layers requires less numeric precision, since they each cover a smaller depth range. This allows LVSMs to run on a wider range of graphics hardware since high precision texture filtering is not required.

In summary, instead of building the entire visibility function using only two moments, we use a piecewise function that is exact at the interval boundaries, with each interval being reconstructed independently. When only one layer is used, the technique reduces to standard VSMs.

3.4 Layer Overlaps

Our analysis thus far has focused on the simple case shown in Figure 2. With a few additional details, however, the arguments generalize to different occluder and receiver distributions.

One problem that we must handle is layer splits that cut through receiver distributions. If this occurs in shadowed regions the minimum variance clamp can cause a small, lit gradient while approaching the layer edge. Conversely, lit regions can have small shadow gradients due to bilinear filtering. Both of these artifacts can be seen in Figure 4.

Fortunately these problems can be avoided by allowing a small overlap between adjacent layers. For N layers with split locations $s_i, i = 0..N$ we widen the layer intervals by $\pm\delta$:

$$\begin{aligned} L_1 &= [s_0, s_1 + \delta] \\ L_2 &= [s_1 - \delta, s_2 + \delta] \\ L_3 &= [s_2 - \delta, s_3 + \delta] \\ &\cdots \\ L_N &= [s_{N-1} - \delta, s_N] \end{aligned}$$

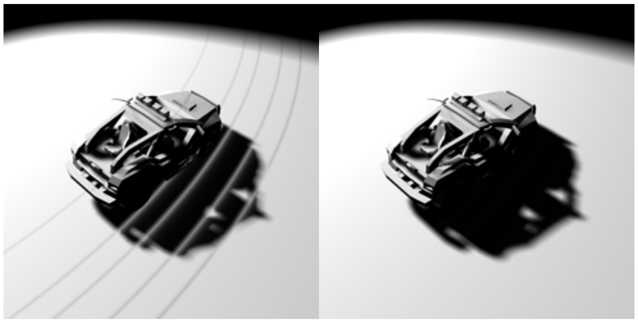


Figure 4: Artifacts visible at layer edges (left). These can be completely eliminated by employing small overlaps between layers (right). Note that the filter sizes have been exaggerated to demonstrate the artifact more clearly.

The only remaining question is how to choose the split locations s_i, which is the topic of the next section.

4 Placing Layers

The strategy used to partition the depth range depends on application goals. If the goal is to achieve the best numeric precision over the entire depth range, a simple uniform split scheme is ideal. If the goal is to eliminate light bleeding on a specific receiver surface, one or more split points can be placed near the surface manually.

For arbitrary scenes, a more automated approach is desirable. We describe one such algorithm in this section. Ideally we want to place layers between the second and third objects in a light bleeding situation. Generally we do not know where light bleeding is going to occur, so in practice we want to find depth discontinuities that *may* be susceptible to this artifact and preemptively avoid it.

Note that even if we cannot place a layer "perfectly" between the second and third objects, if we place a layer between the first and second we will still *reduce* any light bleeding. Placing such a layer clamps the magnitude of Δx, and thus reduces the relative intensity of the light bleeding (see (4)).

We will now present an iterative algorithm based on Lloyd relaxation [5] that attempts to place the split points as intelligently as possible without any specific knowledge of the scene.

4.1 Lloyd Relaxation Algorithm

Light bleeding situations happen at depth discontinuities in the shadow map, that is, at shadow edges. Our approach treats the problem of placing layer splits as a 1-dimensional weighted K-means clustering problem, where the "means" are the second occluder depths and the weights are chosen based on the likelihood of light bleeding. By applying Lloyd's relaxation algorithm using these definitions, we place K layer boundaries at reasonably optimal peaks in the second occluder depth histogram.

To find potentially problematic discontinuities, we consider small neighbourhood filter regions in the unwarped variance shadow map. For each of these regions, we estimate the second occluder depth and compute suitable weights using the first two moments of the depth distribution.

A useful weight is the variance (σ^2) over some neighbourhood in the unwarped VSM, optionally clamping very low or high variances. This has the benefit of more strongly weighting discontinuities that could produce severe light bleeding due to high variance. While we have found this weight to work quite well in practice, application-specific knowledge could potentially produce even better choices.

To recover the second occluder depth, consider the case of only two discrete depths a and b in a given filter with coverages α and $1-\alpha$ respectively:

$$\begin{aligned}
\mu &= \alpha a + (1-\alpha)b \\
\sigma^2 &= M_2 - M_1^2 \\
&= \alpha a^2 + (1-\alpha)b^2 - (\alpha a + (1-\alpha)b)^2 \\
&= \alpha(1-\alpha)(a^2 - 2ab + b^2) \\
&= \alpha(1-\alpha)(a-b)^2
\end{aligned}$$

When the filter is centered over the depth discontinuity ($\alpha = \frac{1}{2}$), σ^2 is maximized. In this situation:

$$\begin{aligned}
\mu + \sigma &= \frac{a+b}{2} + \sqrt{\frac{(a-b)^2}{4}} \\
&= \frac{a+b}{2} + \frac{|a-b|}{2} \\
&= \max\{a, b\}
\end{aligned}$$

Thus when $\alpha = \frac{1}{2}$ we can recover the second occluder depth simply by computing $\mu + \sigma$. For other values of α we will not get an exact answer, but $\alpha = \frac{1}{2}$ is the most important case, since it will be weighted the most highly by the variance weight.

The real benefit of this approach is that it will still produce something reasonable even with large filters and complicated distributions that may not be bimodal. The same cannot be said for assuming a bimodal distribution and attempting to solve for a or b directly.

With these "means" and weights, we perform an iterative Lloyd relaxation entirely on the graphics hardware (implementation details are discussed in the next section). The result of the algorithm is to move the layer split points toward the peaks in the weighted "second occluder" histogram that we have computed.

5 Implementation

We have implemented layered variance shadow maps in Direct3D 10 [2]. Although the algorithm is fully compatible with previous APIs and graphics hardware, several features of Direct3D 10 are beneficial for LVSMs. Specifically, we can make use of texture arrays and have access to the pre-resolved multi-sample anti-aliasing (MSAA) samples.

Generating the layered variance shadow map is straightforward. Geometry is rendered from the light's perspective as usual, and some depth metric is computed for each fragment. Our implementation uses the normalized Euclidean distance to the light point or plane. For each layer, the depth is warped using (6), and the resulting warped depth and warped depth squared are output to a layer render target. Since we only need two components per layer, we pack two layers into each four-component data texture.

We currently use multiple render targets (MRTs) to output all of the layer data in one pass. Alternatively all of the layer data could be computed in a post-process from a single unwarped depth texture (using a fast z-only pass). Using the latter approach, the additional cost of layered variance shadow maps over standard or variance shadow maps is independent of the scene complexity, since all of the warping operations are performed entirely in image space. Moreover the latter approach does not require MRTs, which may not be supported efficiently on some platforms.

We generally use 16-bit per component fixed-point (normalized) textures to store the layer data as they provide sufficient precision when used with even a small number of layers. However, a different texture format may be desirable depending on the filtering capabilities of the target hardware.

After rendering the LVSM, we optionally blur all of the layers, and generate mipmaps. We store the layers in a texture array. However, standard texture atlasing techniques can be used if texture arrays are unavailable.

While rendering the scene, we determine which layer the current fragment falls into and sample the associated shadow map using hardware anisotropic filtering. The relevant layer index can be computed directly for the uniform split scheme, or it can be determined in the general case by searching the monotonic split point sequence. After retrieving the two warped moments, we warp our fragment depth using the same layer warping function. We then evaluate Chebyshev's Inequality (3), optionally clamping the variance to the minimum value for this warp (7). The resulting visibility value is used to attenuate the light reaching the surface.

To avoid any discontinuities at layer boundaries in light bleeding regions, we also use a small transition region, in which the visibility functions of two adjacent layers are blended (recall that we already needed a small overlap between our layers). This step can, however, be skipped with minimal quality loss to avoid the additional sampling and computation.

The Lloyd relaxation algorithm is implemented by rendering a low precision (16-bit fixed point) unwarped VSM in addition to the warped layers. Subsequently we choose two split points to update and generate the K-means clustering data by taking bilinearly filtered samples from the unwarped VSM, positioned in between the texel locations, resulting in a simple 2x2 filter region. We compute our K-means value ($\mu + \sigma$) and determine whether it falls into either (or both) of the split clusters that we are updating (i.e. the given split point is the nearest split point to the sample). If so, we assign a sample weight based on the variance and write out the weighted sample values and the weights for both splits to a four component render target.

At this point we need to compute the weighted mean of all of the data in the new render target. We do this by generating a mipmap pyramid for the texture and reading back the coarsest mipmap level. Dividing the resulting average weighted values by the average weight produces the desired result. In our implementation we defer reading back the results to the CPU (and updating the splits) for a few frames to avoid stalling the graphics pipeline. Alternatively if the split point data is not required on the CPU it can simply be left on the GPU.

If convergence latency is an issue, more splits can be updated per frame (even all of them). In our experience, however, convergence is generally very fast even for dynamic scenes even when only updating two split points per frame.

To avoid the algorithm getting stuck in local optima, we also use a simple split "teleportation" heuristic: if the average weight in a cluster is zero (i.e. the layer has no objects in it), we merge the two adjacent layers and split whatever layer has the highest weight. This simple heuristic greatly improves the quality of the final solutions.

6 Results

All of the pictures and performance numbers in this section were generated on an Intel Core 2 Duo machine with a single NVIDIA GeForce 8800 GTX using Direct3D 10. Unless otherwise stated, all of the layer positions were determined automatically using our modified Lloyd relaxation algorithm.

Figure 5 shows the same scene as in Figure 3, except now rendered with a 4-layer LVSM. Note that the light bleeding has been completely eliminated by a single well-positioned layer boundary. Figures 1 and 6 show examples from a large scene with many overlapping shadows (Figure 7). Note that this scene was designed as a worst-case scenario for VSM-based tecniques due to the extreme

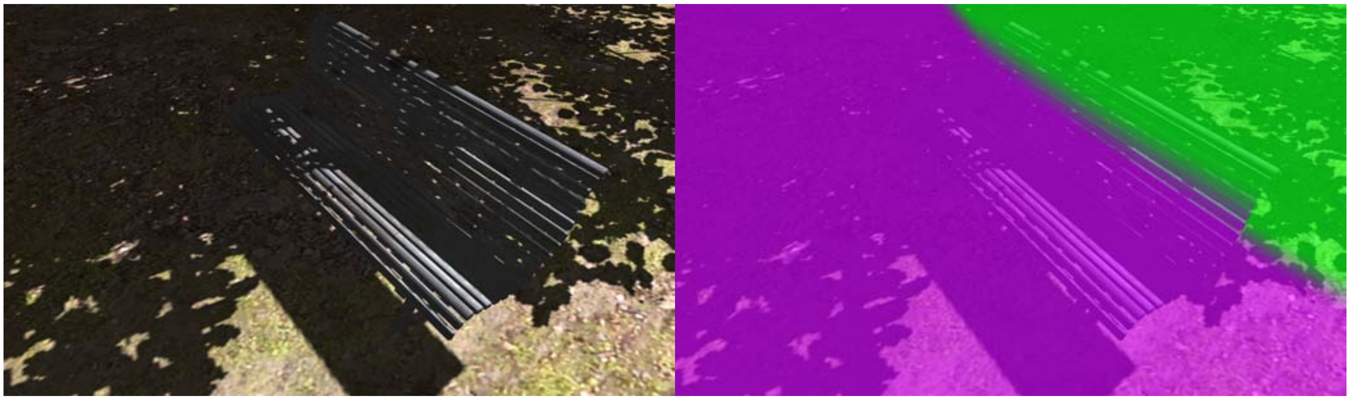


Figure 5: Light bleeding artifacts are not present in an image rendered with LVSMs (left). The layers are visualized using different colors in the right image.

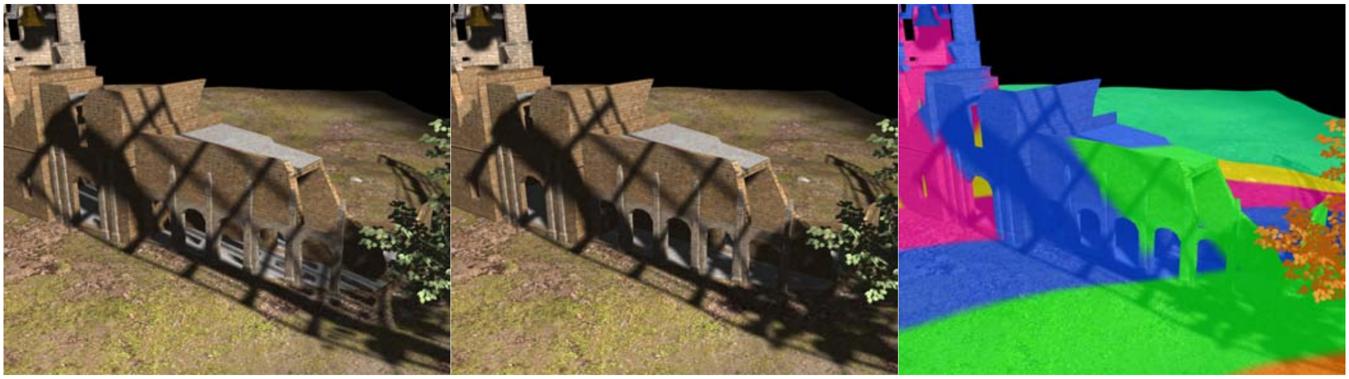


Figure 6: Significant light bleeding (left) is reduced or eliminated using LVSMs (center) with several layers, visualized on the right.

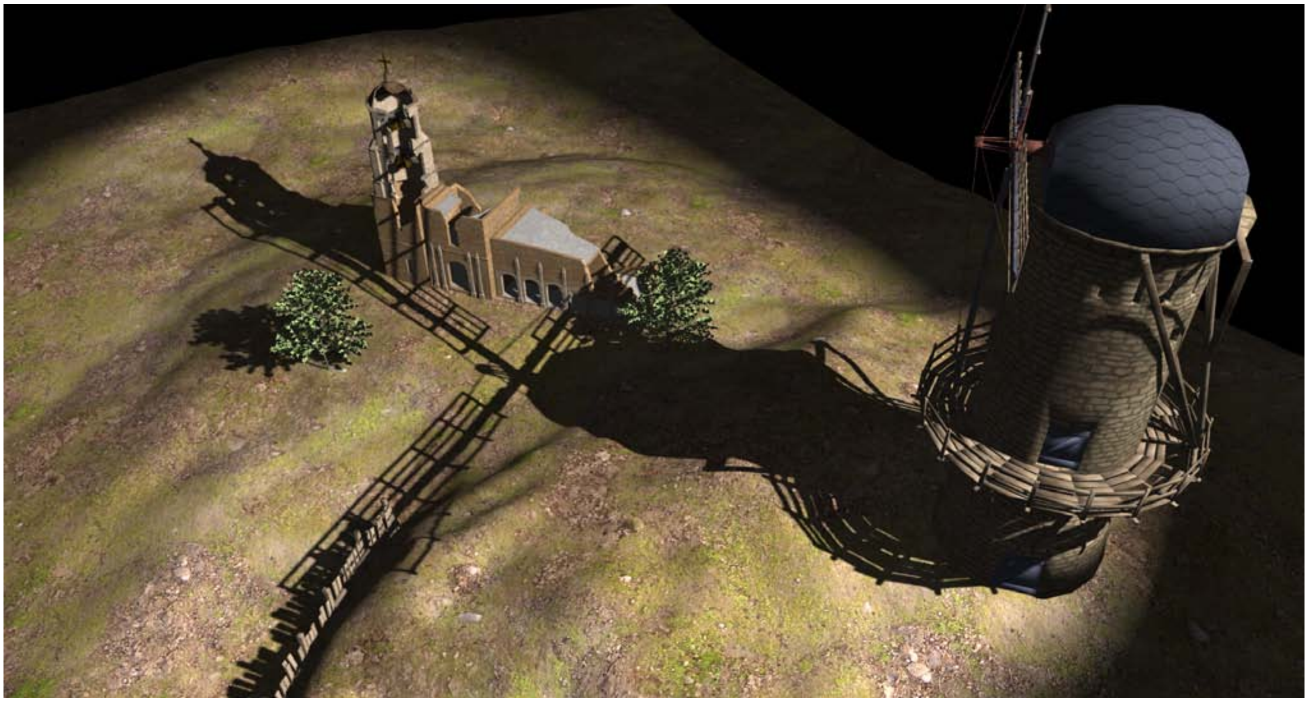


Figure 7: High quality shadow filtering using our algorithm.

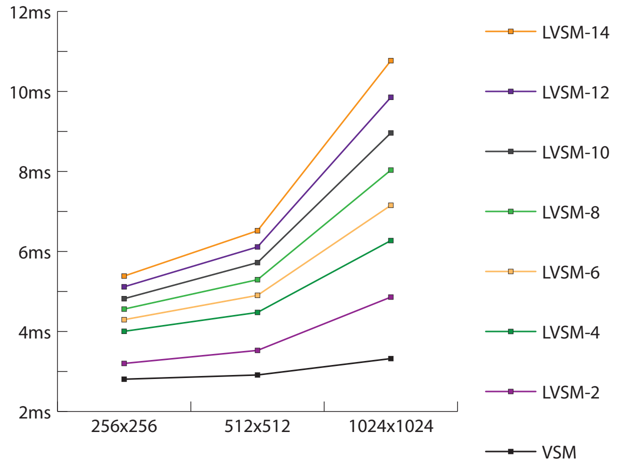


Figure 9: Frame rendering time comparison of VSM and LVSM for various shadow map sizes. LVSM-N indicates LVSM with N layers. All tests were performed with with a framebuffer resolution of 1920x1200 with 4x MSAA.

depth ratios of the overlapping shadows. Artifacts are less pronounced in more typical scenes.

Figure 8 compares convolution shadow maps [1] to layered variance shadow maps, with both uniform and Lloyd relaxation-driven layer positions.

Both CSM and LVSM are scalable and can increase the shadow quality at the cost of additional storage. Each column in the table repesents an equal amount of storage; note that CSM can use twice as many coefficients (M) as LVSM can use layers because CSM can store the coefficients in 8-bit textures while LVSM requires 16-bit textures when few layers are used. It should also be noted that both techniques can fully eliminate the artifacts present in this examble by using even more storage.

As the figure demonstrates:

- LVSMs are usable with less storage than CSMs particularly when Lloyd relaxation is used. The quality of both techniques scales similarly with increased storage.
- LVSMs and CSMs both have light bleeding in similar areas, but LVSMs only have a problem with overlapping occluders while CSMs have a problem with all occluders.
- As the amount of storage increases, LVSMs outperform CSMs due to requiring only a single texture sample per shaded fragment. CSMs must sample all of the coefficients for every fragment.

Figure 9 shows a performance comparison of our algorithm using different numbers of layers with standard variance shadow maps. All of the 1920x1200, 4x MSAA framebuffer is covered and shaded, and every LVSM layer is visible onscreen. Since both algorithms operate in image space, we used a simple scene to eliminate equalizing bottlenecks (Figure 4 with textures). Complex scenes impose an equal performance cost on all image-space shadow map algorithms, including layered variance shadow maps.

The scene is fully dynamic and all of the shadow data is recomputed every frame. Our LVSM implementation is full-featured: it employs real-time Lloyd relaxation, layer transition regions, trilinear and 16x anisotropic filtering, and operates on 16-bit per component fixed-point textures.

Performance data clearly demonstrate that LVSMs are well-suited for use in real-time applications. Even with a 14-layer 1024x1024 LVSM performance remains high, dipping to just under 100 frames per second at 1920x1200 with 4x MSAA. This is largely due to our layer parameterization, which allows us to sample a single layer per shaded fragment. For more complex scenes the difference in performance between VSMs and LVSMs is even less pronounced.

For instance, a 2-layer, 16-bit per component, uniform LVSM uses the same amount of memory as a standard VSM and provides similar performance. Indeed using a 2-layer LVSM can be conceptualized as a simple way of distributing data precision into multiple components, with the additional benefits of reduced light bleeding and improved texture filtering performance.

7 Other Warps

While layered variance shadow maps have several good scaling characteristics, they require a non-trivial amount of storage to fully eliminate light bleeding. Thus it is interesting to consider some other ways to warp the depth function.

As discussed in Section 3, light bleeding artifacts are the worst when the ratio of Δx to Δy is large. LVSMs address this by attempting to clamp Δx to zero or a very small number. There are, however, other monotonic warps that decrease this ratio.

For instance, it is interesting to consider the exponential warping function e^{cx} (where c is a constant) as suggested by [11], but still using a second moment and Chebyshev's Inequality. This warping has the effect of relatively moving object B toward object A which reduces the above ratio. The resulting *exponential variance shadow map* (EVSM) has greatly reduced VSM-like light bleeding while still avoiding any bleeding near occluders.

As c is increased, light bleeding is more aggressively eliminated. The author of [11] notes, however, that using a very high c value produces artifacts on non-planar or multiple receivers. With EVSMs, however, this can be avoided by using the positive e^{cx} warping in conjunction with its negative counterpart: $-e^{-cx}$. This negative exponential warps object B toward object C, making the shadows on B smooth (as they are on C) and thus avoiding the non-planarity problem with exponential shadow maps (ESMs) [11]. These two warps can be used together: since they both provide upper bounds on the visibility function (via Chebyshev's Inequality), taking the minimum of the two bounds gives a good approximation to the visibility function in most cases. Artifacts will only occur in places where both VSMs and ESMs have artifacts, and increasing c will reduce those instances.

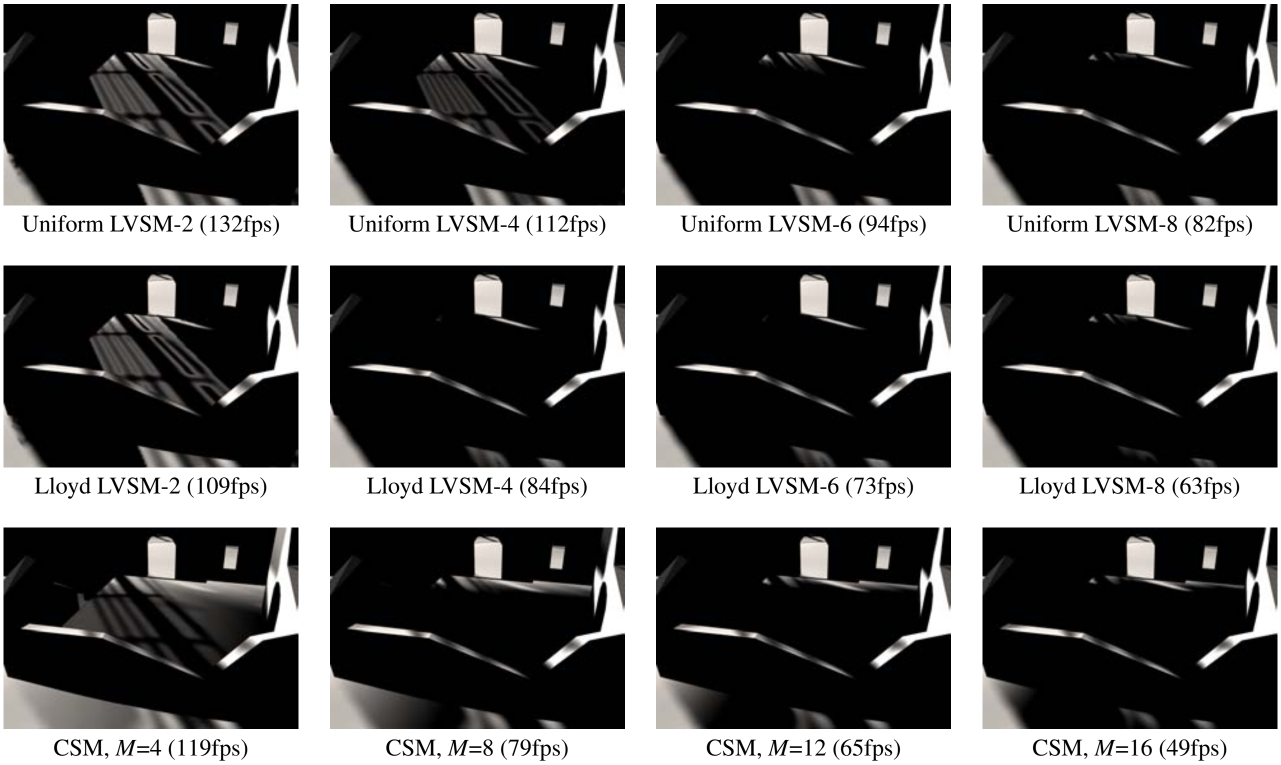
Figure 10 shows the result of this exponential warping compared to the algorithms in Figure 8. EVSMs not only produce better-looking shadows than both CSMs and LVSMs, but they are faster and require only a single, four-component 32-bit float texture (two moments each for the positive and negative warps).

While these results are preliminary, they suggest that there are many interesting warps that can be used in conjunction with variance shadow maps. Exponential warps in particular appear to be a promising direction for future research.

8 Conclusion

We have described a novel generalization of variance shadow maps that provides a scalable way to reduce or eliminate the light bleeding artifacts associated with VSMs. Another advantage is that layered VSMs are applicable to a wider range of graphics hardware than standard VSMs. The additional storage and performance cost of the algorithm is at worst linear in the number of layers.

We have also presented a useful algorithm for automatically placing layers. Further improving layer placement is a good direction for future research. One approach would be to perform the Lloyd relaxation in view space, so that the layer positions are decided based on the size of potential light bleeding regions onscreen, rather than regions that may be completely hidden.



Uniform LVSM-2 (132fps) Uniform LVSM-4 (112fps) Uniform LVSM-6 (94fps) Uniform LVSM-8 (82fps)

Lloyd LVSM-2 (109fps) Lloyd LVSM-4 (84fps) Lloyd LVSM-6 (73fps) Lloyd LVSM-8 (63fps)

CSM, M=4 (119fps) CSM, M=8 (79fps) CSM, M=12 (65fps) CSM, M=16 (49fps)

Figure 8: Comparison between convolution shadow maps and layered variance shadow maps. LVSM-N indicates LVSM with N layers. Each column of the grid uses an equal amount of storage. Performance numbers were taken at 1920x1200, 4x MSAA with a 1024x1024 shadow map.

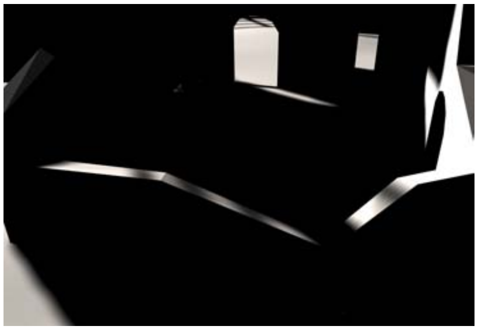


Figure 10: Exponentially-warped variance shadow maps show promising results with good performance (140fps).

In conclusion, we have shown that LVSMs produce good quality shadows while maintaining high performance. They are quite suitable for use in games and other real-time graphics applications that require high quality shadow filtering.

Acknowledgements

We would like to thank William Donnelly and Marco Salvi for their many helpful ideas and suggestions, and Chris Iacobucci for the 3D models and textures used in the scenes. Additionally, we would like to thank our reviewers for their useful feedback.

References

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

[2] D. Blythe. The direct3d 10 system. *ACM Trans. Graph.*, 25(3):724–734, 2006.

[3] F. Crow. Shadow algorithms for computer graphics. In *Computer Graphics (Proc. SIGGRAPH)*, volume 11, pages 242–248, July 1977.

[4] W. Donnelly and A. Lauritzen. Variance shadow maps. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165, New York, NY, USA, 2006. ACM Press.

[5] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, 1999.

[6] T.-Y. Kim and U. Neumann. Opacity shadow maps. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 177–182, London, UK, 2001. Springer-Verlag.

[7] A. Lauritzen. Summed-area variance shadow maps. *GPU Gems 3*, pages 157–182, 2007.

[8] B. Lloyd, D. Tuft, S. Yoon, and D. Manocha. Warping and partitioning for low error shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2006*, pages 215–226. Eurographics Association, 2006.

[9] T. Lokovic and E. Veach. Deep shadow maps. In *Computer Graphics (Proc. SIGGRAPH)*, pages 385–392, 2000.

[10] W. Reeves, D. Salesin, and R. Cook. Rendering antialiased shadows with depth maps. In *Proc. SIGGRAPH*, volume 21, pages 283–291, July 1987.

[11] M. Salvi. Rendering filtered shadows with exponential shadow maps. *ShaderX6*, pages 257–274, 2008.

[12] L. Williams. Casting curved shadows on curved surfaces. In *Proc. SIGGRAPH*, volume 12, pages 270–274, Aug. 1978.