DOI: 10.1111/cgf.12506

COMPUTER GRAPHICS *forum*
Volume 00 (2014), number 0 pp. 1–11

Filtering Multi-Layer Shadow Maps for Accurate Soft Shadows

K. Selgrad[1], C. Dachsbacher[2], Q. Meyer[3] and M. Stamminger[1]

[1]Computer Graphics Group, University of Erlangen-Nuremberg, Erlangen, Germany
kai.selgrad@cs.fau.de, marc.stamminger@fau.de
[2]Karlsruhe Institute of Technology, Karlsruhe, Germany
dachsbacher@kit.edu
[3]Siemens AG, Corporate Technology, Munich, Germany
quirin.meyer@siemens.com

Abstract

In this paper, we introduce a novel technique for pre-filtering multi-layer shadow maps. The occluders in the scene are stored as variable-length lists of fragments for each texel. We show how this representation can be filtered by progressively merging these lists. In contrast to previous pre-filtering techniques, our method better captures the distribution of depth values, resulting in a much higher shadow quality for overlapping occluders and occluders with different depths. The pre-filtered maps are generated and evaluated directly on the GPU, and provide efficient queries for shadow tests with arbitrary filter sizes. Accurate soft shadows are rendered in real-time even for complex scenes and difficult setups. Our results demonstrate that our pre-filtered maps are general and particularly scalable.

Keywords: soft shadow, shadow mapping, filtering, rendering

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism Color, shading, shadowing, and texture

1. Introduction

The realism of interactively rendered scenes has seen great improvements over the last years and decades. Nevertheless, even problems at its core, such as shadow computation, remain challenging. In interactive and real-time applications, shadow mapping [Wil78] is the dominant approach, and numerous extensions thereof have been presented, in particular for soft shadows and, more recently, shadow map filtering, e.g. [DL06, AMB*07, SFY13].

Filtering shadow maps is challenging, because not the shadow map depth values are to be filtered, but rather the binary result of the depth comparisons. Since a fragment's depth value for the shadow test is not known in advance, it has been proposed to store information about the distribution of depth values instead [DL06], [AMB*07]. This distribution information can then be filtered and used to approximate filtered depth comparisons. These approaches can be extended to render soft shadows by adapting the filter size to an estimated occluder distance [ADM*08, SFY13]; however, they often suffer from light bleeding.

In this paper, we present an alternative, more direct pre-filtering technique based on multi-layer shadow maps [XTP07], which offers several benefits for rendering soft shadows with varying penumbra sizes, as shown in Figure 1. Our shadow map filtering technique can, naturally, be used for shadow anti-aliasing by including the current pixel's footprint size into the filter size calculation. However, in this paper we focus on using filtering for rendering soft shadows.

In our shadow map, we store a variable-length list of fragments with depth and opacity for every texel. The key observation is that we can filter (and thus MIP-map) these shadow maps by merging fragment lists of neighbouring texels. During the merge, fragments in an ϵ-depth interval are combined to larger fragments, where partial coverage within this range translates to reduced opacity. Our approach enables efficient soft shadowing and handles opaque and semi-transparent occluders in a unified manner (Figure 1, left). By maintaining multiple layers in the shadow map, we can handle difficult shadow configurations (see Figure 1) that cannot be captured by methods using a single layer only. We demonstrate that our approach is significantly more flexible and accurate than previous work and enables trade-offs between quality and performance.

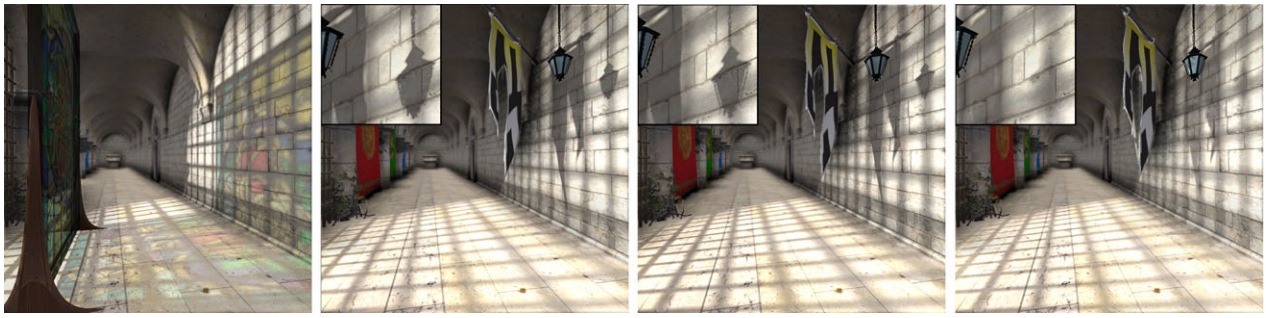

Figure 1: *Our novel filtering technique for shadow maps enables soft shadow rendering in real-time and handles complex scenes efficiently and accurately. It unifies the computation of soft shadows from opaque and semi-transparent surfaces (left) and also handles complicated geometric configurations, e.g. distant and close occluders (centre left, 117 fps). Our method yields renderings close to the reference solution (centre right, Optix, 2.1 s). Previous methods for soft shadow computation or filtering often fail in such cases. Note how, for example exponential soft shadow mapping (right, 169 fps) does not capture the sharp shadows of the lantern.*

In summary, our contributions are:

- A novel approach to shadow map filtering.
- A robust, real-time, soft shadow mapping technique.
- Unified handling of opaque and semi-transparent occluders in soft shadow computation.

2. Previous Work

The importance of the visual cue of shadows in computer graphics has led to an enormous body of research. In interactive applications, shadow volumes [Cro77] and even more shadow mapping [Wil78] are the dominant approaches, while ray-traced shadows, previously only used in offline rendering, have started to become an alternative option. Two recent books [ESAW11, WP12] provide a comprehensive summary of this topic. Consequently, our presentation of prior work focuses on approaches closely related to our paper, in particular filtering methods for shadow map lookups and multi-layer shadow maps.

Shadow map filtering. Shadow maps cannot be filtered in the same way as colour textures, as the binary result of the depth comparison, rather than the depth values, must be filtered (percentage closer filtering [RSC87]). Since the depth of a fragment for the comparison is not known in advance, simple pre-filtering and MIP-mapping as with colour textures is not possible. Instead, the filtering must happen at the time of the shadow lookup, which is very costly for large filter sizes. Fernando [Fer05] approximates soft shadows by determining an average occluder depth from blocking texels and adapting the filter size (Percentage Closer Soft Shadows). The search for blockers is the most taxing part of this algorithm.

Soler and Sillion [SS98] solve the filtering problem by generating blocker images and filtering them. The approach makes very restricting assumptions mainly because it relies on decomposing the scene into blocking layers. Eisemann and Décoret [ED08] lift this restriction and improve blending between layers.

Variance Shadow Maps [DL06] store the mean of depth values and their mean squared. These values can be pre-filtered, resulting in rough information about the depth distribution within larger shadow map regions. Filtered shadow map lookups are possible by assuming a normal distribution of the depth values and applying Chebyshev's inequality. The normal distribution assumption is, in general, not valid for larger filter regions; thus, the method is mainly applied for shadow anti-aliasing and not for soft shadows.

Convolution shadow maps (CSMs) [AMB*07] approximate visibility functions, which replace the standard shadow test with a weighted summation of basis terms. This allows filtering and blurring a shadow map. Exponential shadow maps (ESMs) [AMS*08] use exponential basis functions to achieve better results more compactly. Filter sizes can also be varied depending on occluder depth to render plausible soft shadows [ADM*08, SFY13].

In general, the above pre-filtering techniques become problematic if shadows from multiple occluders overlap, because the resulting depth value distribution is difficult to capture. Figure 1 shows typical problems due to complex geometric configurations. Because our method explicitly stores multiple layers, we can represent and handle such occluders more robustly and generate more accurate soft shadows.

Multi-layer shadow maps. A problem with many soft shadow methods is that shadow maps only store the front-most depth layer, and thus, relevant information for soft shadows becomes lost. Wyman and Hansen [WH03] solve this issue by generating only an *outer* penumbra, resulting in overestimated and sometimes implausible shadows. By storing multiple depth layers, high-quality soft shadows can be achieved. Agrawala et al. [ARHM00] cast rays through such a representation to generate shadows, however, for offline rendering. Layered variance shadow maps [LM08] reduce the variance in depth distributions by storing multiple layers.

Chan and Durand [CD03] generate soft shadows by adding semi-transparent 'smoothies' (often called shadow fins) to occluders' silhouettes. However, this method can only generate outer penumbra and thus overestimates the umbra.

Shadow maps with multiple layers can also be used to render shadows of transparent occluders. Multi-layer transparent shadow maps [XTP07] store a list of fragment depth and transparency values and directly enable computing the total occlusion at a given

depth. While these lists have been created in software, per-pixel linked lists [YHGT10] can be used on modern graphics hardware and have been used for the stochastic rendering of transparent shadows [ME11].

Deep shadow maps. Deep Shadow Maps [LV00], and their modern variants [KN01, YK08, SVLL10], are used to render shadows by out-scattering and absorbing media such as smoke. Instead of storing individual fragments, they sample and encode the monotonous transmittance function along a ray, which can also be computed from highly-detailed geometry such as hair, fur or foliage. While transmittance functions can be filtered, their typical encoding is not practical for computing shadows from surfaces. Variants of deep shadow maps are *Opacity Shadow Maps* [KN01] and *Deep Opacity Maps* [YK08], which store attenuation using layers of transparent shadow maps with uniformly distributed depth values.

Shadow map backprojection. Another prominent class of soft shadow approaches is backprojection [GBP06], where occluding shadow map texels are found and projected back to the rectangular light source to approximate their induced occlusion. As these operations become costly for large penumbrae, multi-resolution approaches are required to achieve real-time frame rates. We refer to the aforementioned books which cover the many variants and techniques in great detail. Shadow map backprojection is not directly applicable to multi-layer shadow maps, because no overlap test of the backprojected texels is performed. Yet, further layers contain important information for soft shadows. Schwarz and Stamminger [SS07] propose an efficient overlap test based on bitmasks sampling the light source, whereas Bavoil et al. [BCS08] implicitly compute overlaps by sampling multiple layers in parallel.

3. Observations and Motivation

The key component to soft shadow mapping approaches is efficiently determining occluders for a shading point. These occluders reside within the *sample frustum*: the pyramid with the area light source as its base and the surface sample to be shaded at the tip. The individual approaches then differ in how they compute the actual shadowing, which can be done using reprojection [GBP06] (which requires tracking the actual coverage of the area light), or combining the occlusion using multiplication or addition [WP12] (which can under- or overestimate shadowing). Our method is based on the following observation: Consider the simple configuration of an area light source, a receiver and an occluder casting a soft shadow (Figure 2, left). Depending on their relative size and distance, we can easily estimate the size of the umbra and penumbra (the umbra can also vanish completely). The shadow on the receiver is a filtered version of the occluder, whereas the penumbra size is equivalent to the filter size. If we now replace the occluder with a semitransparent version using smoothly increasing transparency at the boundaries, as shown in Figure 2 (right), this filtered occluder generates the same shadow if lit by a point light. Soler and Sillion [SS98] followed the same observation; however, they filter only images, while we filter the actual geometry instead.

In Figure 2, we can also see that the filter size is not constant, but rather depends on the relative position of the occluder between the point to shade and the light source. We thus need a way to generate and query filtered versions of the geometry at arbitrary filter sizes.

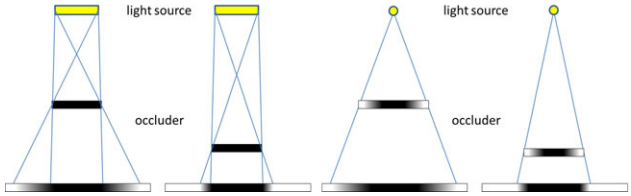

Figure 2: *Umbra and penumbra of an area light source (left). The same shadow can be obtained using an unsharp occluder and a point light source (right).*

To facilitate all of this, our method uses a multi-layer shadow map which stores the scene as a collection of lists with opaque and semi-transparent fragments. First, we show that it is possible to pre-filter such maps and even generate a MIP-map representation with varying filter size. Next, we show how the resulting data structure is used to efficiently generate soft shadows.

4. Filtering Multi-Layer Shadow Maps

Our method relies on a pre-filtered representation of the scene geometry that is obtained by filtering our base-level multi-layer shadow map. In the following, we show how the base level is obtained and describe the filtering process.

4.1. Multi-layer shadow map generation

The base level of our extended shadow map is essentially a multi-layer shadow map [XTP07]. To obtain it, we rasterize the scene from the view of the light source and store the depth and opacity of all fragments. Note that for closed geometry, storing back-facing fragments suffices. The per-texel fragments can either be stored in an array or a linked list (see Section 8) and are then sorted by distance to the light.

This information can naturally be used directly to render hard shadows from semi-transparent and opaque surfaces. To this end, a fragment list would be traversed up to the query depth position and the found opacities accumulated. In this case, all fragments after the first opaque one along a ray can be omitted. However, in contrast to hard shadows, these fragments are required to compute soft shadows and are therefore kept by our method.

In order to prevent the lists from becoming excessively long and to keep the sorting work low, we optionally join fragments of similar depth values and replace them with one fragment of accumulated opacity (see below).

4.2. Downsampling and filtering

After generating the base level, every texel p in the shadow map stores a list of fragments F_p with depth and opacity: $F_p = \{(d_{p,0}, \alpha_{p,0}), ..., (d_{p,n-1}, \alpha_{p,n-1})\}$. If coloured transparent shadows

```
while fragments left
  d_avg = 0, α_i = 0
  do
    i = list_with_closest_fragment();
    (d, α) = F_i.pop();
    d_avg += d
    α_i = accumulate(α_i,α) // see Sect. 4.2
  while fragments within ε-interval
  d_avg = d_avg / fragments_merged
  α = (α_0+α_1+α_2+α_3) / 4
  output(d_avg, α)
```

Figure 3: *Pseudo-code for merging* 2×2 *fragment lists.*

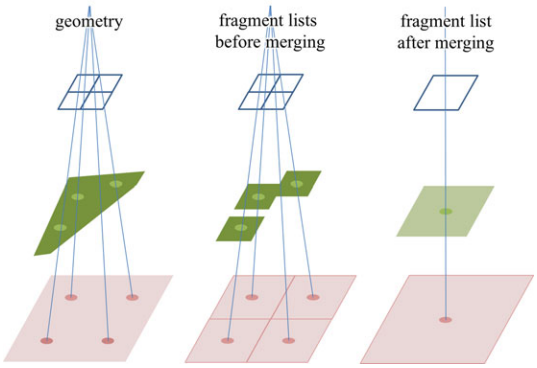

Figure 4: *Merging fragment lists. The three green fragments are merged into a larger one with reduced opacity, the four red ones into a single opaque one. Note that the red and green fragments are not considered close enough to be merged.*

are desired, α must be an RGB-triplet; for soft shadows, a single channel is sufficient.

MIP-mapping. The key to our method is filtering lists of texels, which can be achieved in a manner similar to MIP-mapping, where we average 2×2 texel lists.

To merge four fragment lists $F_1 \dots F_4$ we use a procedure similar to merge sort's combination step, i.e. we assume that fragments in F_i are pre-sorted for depth. Pseudo-code is given in Figure 3, and Figure 4 illustrates the process for a simple example. We generate the merged list by first taking the respective closest fragment out of $F_1 \dots F_4$ ($(d, \alpha) = F_i$ .pop), and then gathering all fragments within the ϵ-interval $[d; d + \epsilon]$. The depths of the fragments are simply averaged. α-Values of fragments from the same list F_i are accumulated into separate α_i using a general function `accumulate` (see below). In the end, the α_i of the different lists are simply averaged, since they do not overlap in image space.

When constructing the MIP-map pyramid, we relax the threshold ϵ for every subsequent level (see Section 6). This results in fragments being merged that were considered too far apart at previous levels of higher resolution.

Figure 5: *Different levels of our shadow map, projected back into 3D space, and the shadow generated by these (including bilinear interpolation). Note the increasing penumbra size and the transparent fragments at the silhouette.*

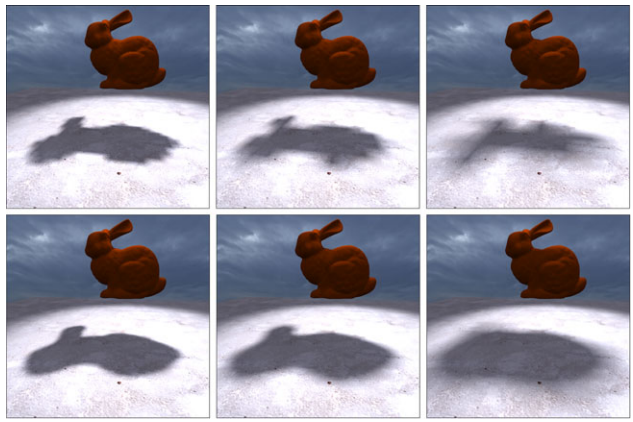

Figure 6: *Pre-filtered shadow maps. Levels four to six are shown, corresponding to filter sizes of* 16×16 *up to* 64×64*. Top row: MIP-maps; bottom row: Y-maps.*

Filtered shadow lookup. An illustration of the different levels stored in a MIP-mapped shadow map is shown in Figure 5. The filter sizes are associated with MIP-map levels and increase by a factor of 2 with every level. We can now use each of these new levels to generate filtered shadows with an increasing, albeit constant per level, penumbra size.

In standard shadow mapping, the depth of a shading point is tested against the depth stored in the respective texel of the shadow map. With our multi-layer shadow map, this binary test is replaced with a traversal of the texel's fragment list, thereby accumulating the opacities of the fragments between the light and the shading point, which naturally handles semi-transparent occluders.

This shadow map lookup should, of course, be interpolated to obtain smooth results. In practice, this means we traverse the four lists closest to the shading point and bilinearly interpolate the result. The top row of Figure 6 shows the results using different filter levels. We can even approximate continuously varying filter sizes by interpolating between MIP-map levels.

Accumulation. We accumulate opacities at various stages of our algorithm. When merging fragments from different lists, opacity addition is the best choice, as we know that fragments do not overlap. When merging fragments within a list and for the accumulation of

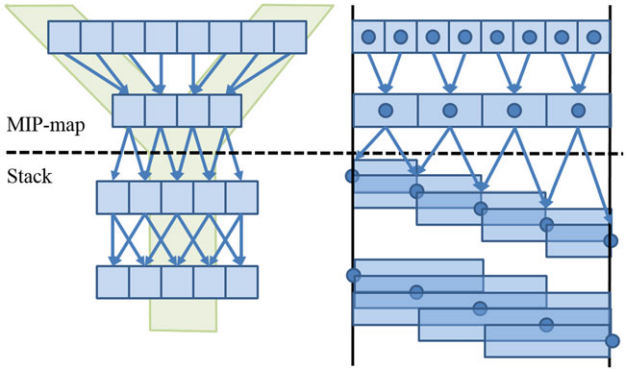

Figure 7: *Generation of a Y-map: First, 2×2 neighbouring texels are merged in each step (MIP-map). At a given resolution (here at 4 texels), we switch to a stack. From then on, pairs with increasing distance are merged. Due to boundary handling, the first stack image becomes one texel wider. The right image shows the resulting filter sizes.*

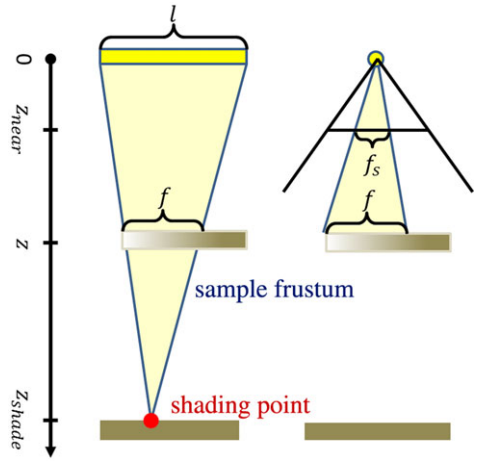

Figure 8: *Left: Computing the filter size required for an occluder at depth z. Right: Projecting this filter size into shadow map space.*

transparencies during the shadow lookup, three different modes can be found in previous work: addition (assuming non-overlapping occluders), multiplication (for transparent occluders) or maximation (for overlapping occluders) [WP12]. We discuss these choices in Section 6 where we analyse our soft shadow method.

Generating Y-maps. We note that the decrease in resolution of the MIP-map levels impairs the visual quality. For large filter sizes, the resolution of the MIP-map levels becomes lower, leading to artefacts that accentuate the underlying texel structure, see Figure 6 (top row).

The reason for these artefacts is that, in contrast to colour texture MIP-mapping, our MIP-map is significantly magnified for large filter sizes. Note that we do not use MIP-maps to avoid aliasing from minification, but instead to efficiently compute and store versions of the filtered scene.

To avoid these artefacts, we need to reduce magnification for the coarse levels. To this end, we build a hierarchy corresponding to a filter stack instead of a pyramid starting from some level, e.g. from 128×128. This means that consecutive images are still filtered with progressively larger filters, however, without further halving their resolution. Similar data structures have been proposed previously under the name Y-map [SS08]. Figure 6 compares the results with standard MIP-mapping.

This computation of such a Y-map is depicted in Figure 7. In the first levels, 2×2 texels are merged, resulting in a halved resolution at each step. At some given resolution, we switch to a filter stack. As shown in Figure 7, the filter kernels start overlapping, in contrast to the MIP-map part. Nevertheless, each texel can be computed by merging exactly 2×2 texels from the previous level. Note that the first stack-filtered level is one texel wider and offset by half a texel (see Figure 7, right). For all following levels, the size then remains constant, as does the half-texel offset.

5. Filtered Multi-Layer Soft Shadows

With the different levels of our filtered shadow map, we can generate softened shadows with different penumbra sizes. However, for real soft shadows, the penumbra size varies according to the position of the occluder. A particularly difficult constellation is when penumbrae of different sizes overlap, as shown in Figure 1, a case that is not handled properly by many real-time approaches.

In this section, we describe how we can combine layers with varying filter sizes to achieve accurate, real soft shadows with varying penumbra size, even for the difficult overlapping case.

5.1. Soft shadow lookup

Consider the geometric configuration shown in Figure 8. An area light of size l at depth $z = 0$ illuminates a receiver at depth z_{shade}. If we consider an occluder at depth z, then we can see how large the region is where surfaces can potentially occlude the area light source. The size of this region defines the required filter size f (in world space) for computing soft shadows using pre-filtered geometry:

$$f(z) = l\left(1 - \frac{z}{z_{\text{shade}}}\right).$$

Projecting this filter size f into shadow map space (Figure 8, right) results in

$$f_s(z) = f(z)\frac{z_{\text{near}}}{z} = l z_{\text{near}}\left(\frac{1}{z} - \frac{1}{z_{\text{shade}}}\right).$$

Using this, when looking at a shadow map fragment at depth z, we can determine which filter size and thus which resolution level of the pre-filtered map should be applied.

```
float shadow(vec2 uv, float z)
  float opacity = 0
  foreach resolution level i
    float (z_s, z_e) = depth_range(i, z)
    foreach fragment frag
         in list from z_s to z_e at uv
      opacity = accumulate(opacity,
                           frag.opacity)
  return opacity
```

Figure 9: *Pseudo-code for soft shadow lookup without interpolation.*

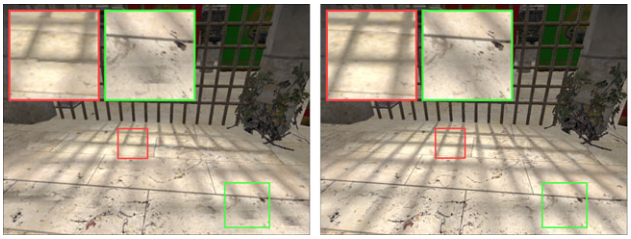

Figure 10: *With bilinear interpolation only, the levels of our hierarchy become visible. Trilinear interpolation of fragment lists ensures smooth level transitions at negligible cost.*

Straightforward manipulation of $f_s(z)$ yields

$$z(f_s) = \frac{l z_s z_n}{f_s z_s + l z_n}$$

for which the filter size is the input. With this we can determine at which depth values a given filter size is appropriate for each shading point. This leads to an efficient traversal method where, for each shading point, we iterate over all resolution levels of our hierarchy and determine the depth range $[z_s, z_e]$, e.g. $[z(w^i), z(w^{i+1})]$, for which each level fits the filter size w^i. Within this range, we accumulate the opacity of the stored fragments. Figure 9 details this step as pseudo-code. As mentioned above, we can add bilinear filtering by interpolating the result from the shadow test with the four closest texel lists.

5.2. Trilinear interpolation

In the pseudo-code of Figure 9, for each resolution i, we only use fragments from a depth interval $[z_s, z_e]$. For different resolution levels, depth intervals do not overlap, i.e. for an occluder at depth z, we use exactly one shadow map level, resulting in discontinuous switches of penumbra sizes (Figure 10, left). To get a smooth interpolation of penumbrae, we use the following approach: We extend the depth intervals such that fragments are gathered from the *two* best-fitting resolution levels. The gathered fragments are weighted depending on their z-value by a hat-function over the depth interval, resulting in smooth blending between the resolutions (Figure 10, right). Note that, in practice, we cannot guarantee that the blending weights add to one, since the depth values may vary slightly between different levels. However, we have not noticed visible artefacts from this; in fact, the gain in visual quality is considerable and the scheme's impact on performance is negligible.

Note that this interpolation scheme furthermore facilitates shadow anti-aliasing by choosing the starting level and weight accordingly.

Figure 11: *A fence with fine detail from alpha maps casting shadows (rendered using our method) with a close and distant light source.*

5.3. Alpha mapping

Our method adapts nicely to transparent occluders, as illustrated in Figure 1, which shows a transparent occluder casting coloured shadows. This also encompasses transparency from alpha maps and especially from MIP-mapped alpha textures, as exemplified in Figure 11.

6. Parameters

Our soft shadow algorithm has a set of intuitive parameters which control the trade-off between visual quality and performance. In this section, we focus on these parameters' impact on visual quality, while performance will be considered in Section 8.

Depth layers. We can prune the fragment lists by using only the first n depth layers of the base shadow map (filtered maps can have more layers). With $n = 1$, our method produces soft shadows comparable to exponential soft shadow mapping (ESSM) [SFY13] (see Figure 12, left). More complex constellations as shown in Figures 1, 12 and 13 require a larger n. Note that already with $n = 2$, the shadow in Figures 12 and 13 looks plausible.

Fragment lists. Recall that we combine fragment lists during filtering, which then become longer. Yet, we also reduce the number of fragments by merging those whose depth values are closer than a threshold ϵ. In our examples, we used $\epsilon(w) = 2w$, where w is the size of a fragment at the corresponding depth. This implies an increase in the threshold by a factor of 2 at each level. Note that as long as additive accumulation is used (discussion in the following section), only very large values for ϵ result in artefacts (see Figure 13, centre right, with $\epsilon = 130w$). For additive accumulation, this parameter introduces artefacts where distant geometry

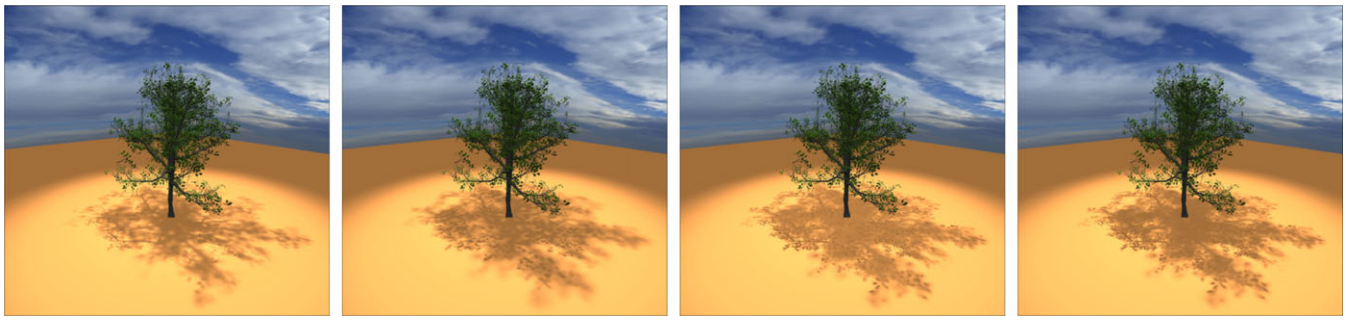

Figure 12: *Shadows of a high-polygon tree. Exponential soft shadow mapping (ESSM) [SFY13] (left) compared to our method with only 1 (centre left), 2 (centre right) and 16 fragments (right) stored in the base layer of the multi-layer shadow map.*

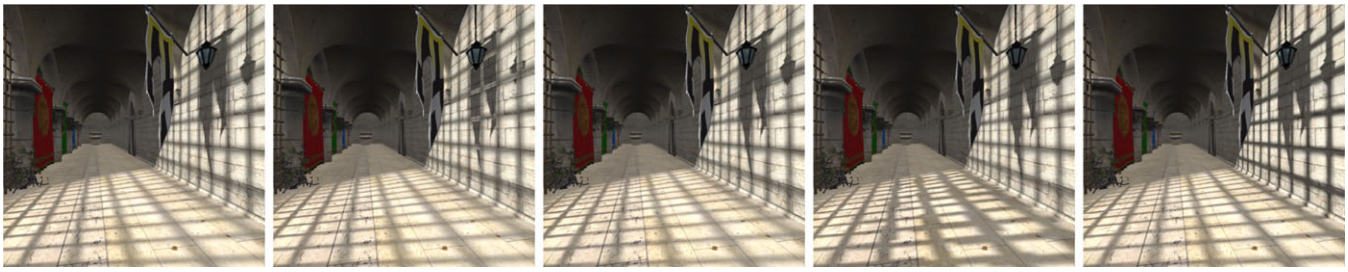

Figure 13: *Different parameter settings with impact on the image quality. From left to right: reference using conservative settings; only a single layer on the base level of our shadow map; two layers on the base level; large value for ϵ (here $130w$); maximum filter level too restricted.*

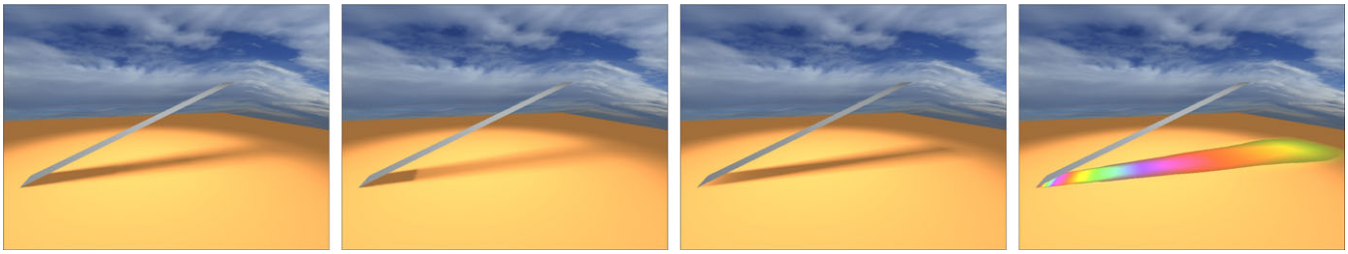

Figure 14: *An occluder spanning multiple levels of our hierarchy. Additive shadow accumulation (left) yields smooth results, whereas careless use of ϵ with multiplicative accumulation (centre left) may break occluders apart. Note that ESSM [SFY13] (centre right) is missing contact shadows which our approach captures nicely. The (colour coded) levels and the interpolation weights employed during shadow computation with our method are shown in the rightmost panel.*

is merged (starting at $\epsilon = 70w$ in Figure 13). However, for multiplicative accumulation, occluders can break apart (see Figure 14).

Accumulation. As the above example shows, an important point to consider is the accumulation of opacities. Modes found in previous work are addition (assuming non-overlapping occluders), multiplication (for transparent occluders) or maximation (for overlapping occluders) [WP12].

While multiplicative accumulation may result in light leaks (see Figure 14, centre left), this was usually not visible in our test scenes (see Figures 12 and 17 in Section 8). It also naturally prevents shadow overestimation (see Figure 15, centre) which may result from additive accumulation (Figure 15, left). Note that the presented

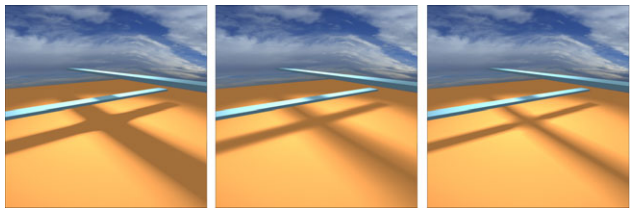

Figure 15: *Additive accumulation (left) is prone to overestimating shadows; multiplicative accumulation (centre) yields correct results. Note that this is a simple yet challenging configuration which also many state-of-the-art algorithms, e.g. ESSM (right), do not handle well.*

Table 1: *Timings (in ms) for the Sponza scene shown in Figure 1 (centre left), rendered at* 1280×720 *on a GeForce Titan (summary in fps). The table shows the effect of the different parameters discussed in Section 6. Values influenced by a parameter change are marked in boldface.*

	Parameter settings				
	default	$\epsilon = 10w$	5 levels	sub	all
Scene rendering	1.46	1.46	1.46	1.46	1.46
Frag counting	0.83	0.83	0.83	0.83	0.83
Scan	0.27	0.27	0.27	0.27	0.27
Frag collecting	1.29	1.29	1.29	1.29	1.29
Frag sorting	0.63	0.63	0.63	0.63	0.63
Filtering	3.35	**2.10**	**2.31**	3.35	**1.81**
Soft shadows	7.93	**6.24**	**5.59**	**3.04**	**2.23**
Frames s^{-1}	63.45	78.00	80.77	91.99	117.37

Table 2: *Timings (in ms) for the 'Streets of Asia' scene as shown in Figure 16, rendered with our soft shadow method (MIP-mapped) at 720p on a GeForce Titan. The different components of our algorithm are shown along with the effect of the parameters presented in Section 6. As before, values influenced by a parameter change are marked in boldface.*

	Parameter settings				
	Default	$\epsilon = 10w$	Five levels	Sub	All
Scene rendering	1.78	1.78	1.78	1.78	1.78
Frag counting	0.98	0.98	0.98	0.98	0.98
Scan	0.27	0.27	0.27	0.27	0.27
Frag collecting	1.85	1.85	1.85	1.85	1.85
Frag sorting	0.60	0.60	0.60	0.60	0.60
Filtering	5.40	**1.87**	**3.78**	5.40	**1.66**
Soft shadows	7.71	**5.17**	**5.81**	**2.54**	**2.16**
Frames s^{-1}	53.7	79.7	68.8	73.2	104.9

configurations are not managed well by state-of-the-art soft shadow mapping approaches, see Figures 14 (centre right) and 15 (right).

Maximum MIP-map level. A very intuitive parameter is to impose a maximum MIP-map level. This directly affects the maximum filter size and thus the maximum penumbra size, which translates to filtering and traversing fewer levels, leading to better performance. Figure 13 (right) shows an example using this parameter. The fact that the penumbra size is too small is only clear when compared to the reference (left).

7. Implementation

We implemented our method using OpenGL 4.3 with compute shaders as we make use of scattered writes and atomic counters to build the multi-layer shadow map.

We experimented with per-pixel linked lists and an array-based implementation to gather fragments. In the latter case, the lists are built by first counting the number of list elements, followed by a scanning pass [SHZO07], and finally collecting and tightly packing in a second render pass. This overhead easily amortizes as the filtering and shadow computation greatly benefit from the simpler data structure; all our timings are reported for this implementation choice. Furthermore, arrays provide the additional benefit of being able to use binary search (e.g. when searching the start of an interval for a given depth value), which leads to significant performance improvements.

8. Results

We evaluated our method with various test scenes on current generation GPUs. If not stated otherwise, we use a base layer of 1024×1024 for our shadow map. Table 1 lists timings measured for the Sponza scene as shown in Figure 1 (centre left), rendered at 720p using our MIP-mapped multi-layer shadow maps. The first column is measured with very conservative parameter settings (listed as 'default'): $\epsilon = 2w$ (in normalized depth), using all 11 levels of our hierarchy. The subsequent columns show parameter changes which did not result in reduced visual quality to illustrate their impact on performance. The first column changes the ϵ threshold to $10w$, resulting in much fewer individual fragments, especially at levels higher-up. This affects filtering and shadowing times as both stages are dependent on the lists' length. In the second column, only the five finest layers of our hierarchy are generated and used for shadow computation. This, too, affects filtering and shadowing performance, because the outer loop over the levels can be terminated early in both cases. The column 'sub' denotes an image space sub-sampling of the soft shadow at every other pixel and subsequent interpolation. By this, we can trade a minor decrease in image quality for a significant reduction in rendering time. Note that the application of more elaborate sub-sampling schemes, e.g. [GBP07], might reduce rendering time further. However, as the assumption of low-frequency penumbrae is not generally applicable to our layered approach, we refrained from more aggressive sub-sampling. The last column shows the performance attained for all three options combined. Table 1 also lists the timings for the individual steps of our method: standard rendering of the scene, fragment counting and scanning provide indexing for tightly packed fragment arrays, which are then filled in with the fragment collection pass. Fragments are then sorted before they can be filtered. Table 2 shows the same evaluation for the more complex 'Streets of Asia' scene consisting of 400 000 triangles with much higher depth complexity. Table 3 compares the rendering performance at different resolutions and also compares using Y-maps to using MIP-maps. On the GeForce Titan, the filtering time of a 1024×1024 shadow map is increased by 22% and shadow lookup by 9% when switching to a stack starting from 128×128 in the Sponza scene.

We compare our results to a ray tracing reference using Optix [PBD*10] where we stochastically sample 128 shadow rays for each image pixel. As Figures 1 and 17 show, our results are very close to this reference. Because we interpolate between two levels of the hierarchy to approximate the desired filter size shadows can become slightly softer than the reference. Furthermore, we compare our method to a recent soft shadow mapping approach which, while being faster to compute, does not capture complicated scene configurations as well as our approach (see Figures 1 and 12). Table 4 lists

Table 3: *This tables details how MIP-maps and Y-maps compare in terms of rendering performance for the Sponza and Streets of Asia scenes shown in Figures 1 and 16, respectively. The timings include the generation of our extended shadow map every frame; the stack part of the Y-map starts at* 128×128 *resolution.*

		Frames s^{-1}	
Resolution	Filtering	Sponza	Streets of Asia
1280×720	MIP-map	117.4	104.9
	Y-map (128)	101.2	86.4
1920×1080	MIP-map	90.2	84.2
	Y-map (128)	80.8	77.9

Table 4: *Performance evaluation of our method as compared to ESSM and our ray tracing reference for the scenes shown in Figures 1, 12 and 16. All measurements were conducted on a Geforce Titan at 720p.*

	Sponza	Tree	Strees of Asia
ESSM	5.2 ms	5.5 ms	6.7 ms
Our	8.5 ms	6.1 ms	9.5 ms
Our (2k)	45 ms	28 ms	38 ms
Our (512)	5.7 ms	4.4 ms	8.5 ms
Ray tracing	2.1 s	1.8 s	1.6 s

Figure 16: *Streets of Asia: A more game-like scene. Detailed timings for the different steps of our algorithm are shown in Table 2, timings for Y-Maps versus MIP-maps are presented in Table 3.*

the corresponding timings, including our method when run with a higher or lower resolution base level shadow map. High-resolution base maps result in a significant increase in computation time due to expensive filtering of the base level. The lower resolution base maps provide further performance enhancement, yet often lead to noticeable artefacts, especially for the highly detailed tree scene shown in Figure 17. As can be seen in Figure 18, the visual difference at coarse levels of our hierarchy is minor. A simple solution is thus to replace the first filter step(s) with simple sub-sampling instead of merging, i.e. to simply reference base level lists at the next coarser level.

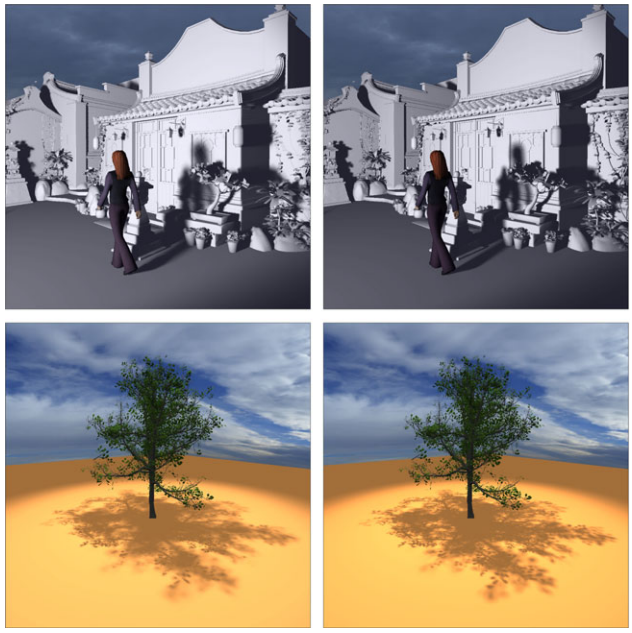

Figure 17: *Our results (left) compare nicely to ray-traced shadows (right).*

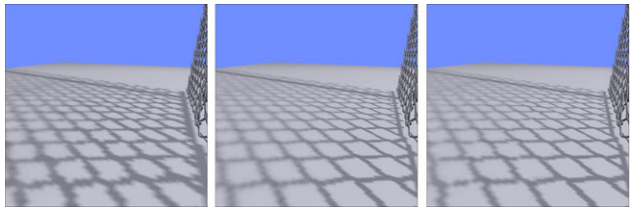

Figure 18: *For very thin geometry, the difference between 1k (left) and 2k shadow maps (centre) is visible for contact shadows. Using sub-sampling during filtering (right) computation time is reduced while maintaining the higher resolution in important regions. The fence shown is high-poly and not alpha-mapped.*

Discussion of parameters. Lowering the maximum filter size requires fewer resolution levels and thus affects the performance of pre-filtering and shadow lookups. This is applicable if an upper limit on the penumbra size is acceptable. In our test scenes (Figures 1 and 16), the filtering performance increases by $1.44\times$ and shadow lookup performance by $1.3\times$ to $1.4\times$ without any effect on visual quality.

The most forgiving parameter, according to our experiments, is the ϵ-threshold. It directly influences the length of the (merged) fragment lists and greatly affects filtering and shadow lookup performance. Tables 1 and 2 report filtering speedups of $1.6\times$ to $2.8\times$, and shadow lookup speed increases of $1.27\times$ to $1.49\times$, without noticeable impact on visual quality.

Failure cases. Figure 13 shows examples of failures due to aggressive parameter settings. It compares our results using conservative parameters sets (left) to using just one layer (centre left) and two

layers (centre) in our structure's base level. As can be expected, using just a single layer does not suffice to capture the scene configuration. With just two layers (even when the objects on the wall are aligned to overlap), the scene still looks plausible (see Figure 12 for a similar case). Figure 13 (centre right) shows an inappropriate merging of occluders with a very large ϵ. Limiting the maximum level used by our algorithm, i.e. the maximum filter size, is shown in Figure 13 (right). Compared to the reference, the shadow is clearly too hard, however, when viewed in isolation, it still looks plausible, making this parameter a very handy tool for optimization.

9. Conclusion and Future Work

In this paper, we presented an efficient real-time approach to soft shadow computation based on a novel filtering technique for multi-layer shadow maps, which is more accurate than previous approaches. It handles opaque and semi-transparent surfaces in a unified manner. Our soft shadow computation yields results close to ground truth, even in difficult geometric configurations where previous methods fail. Additionally, our method is scalable and a set of intuitive parameters can be used to trade shadow quality for performance, as illustrated in Figure 13.

We believe that our method can be further improved, e.g. by better optimizing the number of fragments stored per texel, improving screen space interpolation, or optimizing the filter sizes and resolution levels. Lastly, rendering anti-aliased shadows by generating semi-transparent fragments at silhouette edges also reflects potential for future work.

Acknowledgements

The authors would like to thank Marko Dabrovic and Frank Meinl for the Sponza model, Guillermo Leal Laguno for the lamp model, Flickr user John Dill for the stained glass window (http://www.flickr.com/photos/gobikey/4568069317/) and the BlendSwap users Sqorck, Zuendholz, TiZeta and ScoutingNinja for their respecive models (http://www.blendswap.com/blends/view/{1323, 16544,66413,53828}). We also gratefully acknowledge the generous funding from the German Research Foundation (GRK 1773).

References

[ADM*08] Annen, T., Dong, Z., Mertens, T., Bekaert, P., Seidel, H.-P., Kautz, J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 27*, 3 (2008), 34:1–34:8.

[AMB*07] Annen, T., Mertens, T., Bekaert, P., Seidel, H.-P., Kautz, J.: Convolution shadow maps. In *Proceedings of EG Symposium on Rendering* (2007), pp. 51–60.

[AMS*08] Annen, T., Mertens, T., Seidel, H.-P., Flerackers, E., Kautz, J.: Exponential shadow maps. In *Proceedings of Graphics Interface* (2008), pp. 155–161.

[ARHM00] Agrawala, M., Ramamoorthi, R., Heirich, A., Moll, L.: Efficient image-based methods for rendering soft shadows. *Computer Graphics (Proceedings of SIGGRAPH)* (2000), 375–384.

[BCS08] Bavoil, L., Callahan, S. P., Silva, C. T.: Robust soft shadow mapping with backprojection and depth peeling. *Journal of Graphics, GPU, and Game Tools 13*, 1 (2008), 19–30.

[CD03] Chan, E., Durand, F.: Rendering fake soft shadows with smoothies. In *Proceedings of EG Symposium on Rendering* (2003), pp. 208–218.

[Cro77] Crow, F. C.: Shadow algorithms for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH) 11*, 2 (1977), 242–248.

[DL06] Donnelly, W., Lauritzen, A.: Variance shadow maps. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2006), pp. 161–165.

[ED08] Eisemann, E., Décoret, X.: Occlusion textures for plausible soft shadows. *Computer Graphics Forum 27*, 1 (July 2008), 13–23.

[ESAW11] Eisemann, E., Schwarz, M., Assarsson, U., Wimmer, M.: *Real-Time Shadows*. A.K. Peters, Boca Raton, FL, USA, 2011.

[Fer05] Fernando, R.: Percentage-closer soft shadows. In *Proceedings of the ACM SIGGRAPH Sketches* (Los Angeles, CA, USA, 2005).

[GBP06] Guennebaud, G., Barthe, L., Paulin, M.: Real-time soft shadow mapping by backprojection. In *Proceedings of EG Symposium on Rendering* (2006), pp. 227–234.

[GBP07] Guennebaud, G., Barthe, L., Paulin, M.: High-quality adaptive soft shadow mapping. *Computer Graphics Forum (Proceedings of Eurographics) 26*, 3 (2007), 525–534.

[KN01] Kim, T.-Y., Neumann, U.: Opacity shadow maps. In *Proceedings of EG Workshop on Rendering* (2001), pp. 177–182.

[LM08] Lauritzen, A., McCool, M.: Layered variance shadow maps. In *Proceedings of Graphics Interface* (2008), pp. 139–146.

[LV00] Lokovic, T., Veach, E.: Deep shadow maps. *Computer Graphics (Proceedings of SIGGRAPH)* (2000), 385–392.

[ME11] McGuire, M., Enderton, E.: Colored stochastic shadow maps. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2011), pp. 89–96.

[PBD*10] Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of of SIGGRAPH)* (2010), vol. 29, pp. 66:1–66:13.

[RSC87] Reeves, W. T., Salesin, D. H., Cook, R. L.: Rendering antialiased shadows with depth maps. *Computer Graphics (Proceedings of SIGGRAPH) 21* (1987), 283–291.

[SFY13] Shen, L., Feng, J., Yang, B.: Exponential soft shadow mapping. *Computer Graphics Forum (Proceedings of EG Symposium on Rendering) 32*, 4 (2013), 107–116.

[SHZO07] Sengupta, S., Harris, M., Zhang, Y., Owens, J. D.: Scan primitives for GPU computing. In *Proceedings of Symposium on Graphics Hardware* (2007), pp. 97–106.

[SS98] Soler, C., Sillion, F. X.: Fast calculation of soft shadow textures using convolution. *Computer Graphics (Proceedings of SIGGRAPH)* (1998), 321–332.

[SS07] Schwarz, M., Stamminger, M.: Bitmask soft shadows. *Computer Graphics Forum (Proceedings of Eurographics) 26*, 3 (2007), 515–524.

[SS08] Schwarz, M., Stamminger, M.: Quality scalability of soft shadow mapping. In *Proceedings of Graphics Interface* (2008), pp. 147–154.

[SVLL10] Salvi, M., Vidimče, K., Lauritzen, A., Lefohn, A.: Adaptive volumetric shadow maps. *Computer Graphics Forum (Proceedings of EG Symposium on Rendering) 29*, 4 (2010), 1289–1296.

[WH03] Wyman, C., Hansen, C.: Penumbra maps: Approximate soft shadows in real-time. In *Proceedings of EG Symposium on Rendering* (2003), pp. 202–207.

[Wil78] Williams, L.: Casting curved shadows on curved surfaces. *Computer Graphics (Proceedings of of SIGGRAPH) 12*, 3 (1978), 270–274.

[WP12] Woo, A., Poulin, P.: *Shadow Algorithms Data Miner*. A K Peter/CRC Press, Boca Raton, FL, USA, 2012.

[XTP07] Xie, F., Tabellion, E., Pearce, A.: Soft shadows by ray tracing multilayer transparent shadow maps. In *Proceedings of EG Symposium on Rendering* (2007), pp. 265–276.

[YHGT10] Yang, J. C., Hensley, J., Grün, H., Thibieroz, N.: Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum (Proceedings of EG Symposium on Rendering) 29*, 4 (2010), 1297–1304.

[YK08] Yuksel, C., Keyser, J.: Deep opacity maps. *Computer Graphics Forum (Proceedings of Eurographics) 27*, 2 (2008), 675–680.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's web site.