

Real-Time Soft Shadows with Adaptive Light Source Sampling

Michael Schwärzler*

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Vienna / Austria

Abstract

Simulating physically accurate soft shadows in Computer Graphics applications is usually done by taking multiple samples from all over the area light source and accumulating them. Due to the unpredictability of the size of the penumbra regions, the sampling density has to be quite high in order to guarantee smooth shadow transitions in all cases, making the process computationally extremely expensive and therefore hardly usable in anything else than offline rendering applications.

Thus, we suggest a new approach, in which we select the sampling points adaptively and avoid the generation and evaluation of shadow maps which do not contribute to an increased image quality. The main idea is to reproject the shadow maps of two neighboring sampling points into camera space, investigate how much they differ by exploiting the functionality of hardware occlusion queries, and recursively add more sampling points only if necessary.

While sampling the light source with a fixed number of samples leads to either over- (guarantees constant quality) or undersampling (guarantees constant frame rates) in most cases, our method is capable of selecting only the samples which really contribute to an improved shadow quality. This saves rendering time, but still generates shadows of comparable quality and accuracy. Even though additional calculation time is needed for the comparison step, interactive frame rates are possible in most cases, since the calculation of many unnecessary samples is saved.

Keywords: Soft Shadows, Sampling, Shadow Mapping, Real-Time

1 Introduction

Using shadows in applications and games greatly improves the quality of the generated images and has a big impact on the scene perception: Apart from giving visual cues on the relationship between the scene object, shadows are perceived to be a natural part of the scene. Missing shadows can give a rendered scene an unnatural, artificial look.

While algorithms for the creation of hard shadow are already widely used in today's games and applications, the



Figure 1: A comparison between a regular sampling approach (left, 81 shadow maps, 12 frames per second) and our algorithm (right, 14 shadow maps, 40 frames per second) using an area light source. Images created in the DirectX 10 framework using Texture Arrays.

fast and correct calculation of soft shadows is a complex task and still an area of active research. Soft shadows are, in opposite to hard shadows, not cast by theoretical point lights without extents, but by area light sources. They do therefore consist of umbra (areas where the light source is completely blocked) and penumbra (areas where the light source is partly visible) regions, and despite the increased computational costs, using them is worth the effort: Nearly every shadow in reality has soft boundaries, so using soft shadows in rendering applications significantly increases the realism of the generated images (see Figure 2). Moreover, aliasing artifacts at the boundaries can be hidden by them as well due to their low frequency.

Nearly all current soft shadow approaches are either based on the shadow mapping (see Section 3.1) or the shadow volumes algorithm [5] – the two most widely used hard shadow generation methods – and have been extended to work together with area light sources. There exist both *physically accurate*, and so-called *fake* or *single-sample* soft shadow approaches (see Section 2). Fake approaches are usually faster, but inaccurate, since the area-from-point visibility is simplified to a point-to-point visibility there. Physically accurate algorithms usually suffer from speed problems, and real-time or even interactive frame rates are hard to achieve.

We suggest a new approach, which is based on the idea of sampling the light source several times in order to obtain physically correct shadows in Section 3: We optimize the number of sampling points needed for satisfying results by starting with only very few sampling points and adaptively adding more and more of them, depending on

*michael@schwaerzler.com

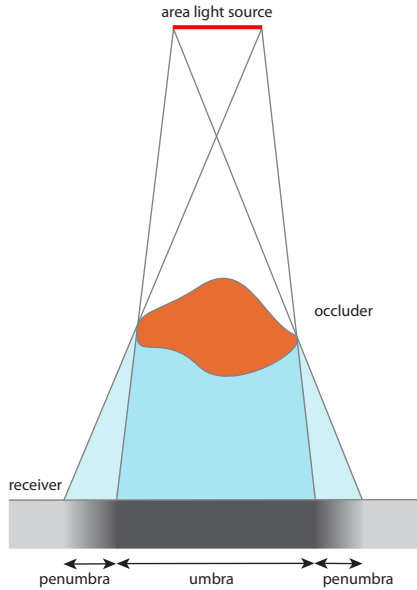


Figure 2: An area light source leads to a soft shadow, which consists of umbra and penumbra regions.

whether the sampling density is already high enough or not. This decision is made by projecting the shadow maps of two neighboring sample points to the camera’s view point, and by comparing there how much they differ. Only if the percentage of differing shadow map texels is higher than a predefined value ϵ , a new sampling point is created in-between the compared ones.

After creating n shadow maps by applying our adaptive sampling strategy, we show how these can be used to render physically accurate soft shadows at interactive or even real-time frame rates. We employ both *Deferred Rendering* as well as *Texture arrays* for our tests and comparisons in our rendering framework (see Section 3.4).

2 Related Work

A vast amount of real-time soft shadow algorithms have been published during the last few years; it is therefore out of the scope of this paper to explain all of them. A not completely up-to-date, but still valuable survey of soft shadowing approaches was published by Hasenfratz et. al in 2003 [13]. We will focus on the most relevant and recent publications which are related to our work.

Since the calculation of physically correct soft shadows is often considered to be too costly, most real-time soft shadow approaches estimate the visibility by calculating it from only one point of the area light source, and simulate the penumbra using approximative heuristics. Fernando [9] suggests using a technique called *Percentage Closer Soft Shadows (PCSS)*, where the *Percentage Closer Filtering (PCF)* [15] method is exploited and combined with a blocker search: PCF softens hard shadow boundaries by not only comparing the current depth to a single value in

the shadow map, but to do so with the neighboring pixels in the shadow map as well. The percentage of successful shadow tests specifies the shadow intensity. It helps to reduce aliasing artifacts at the softened shadow boundaries, but the penumbra is far from being accurate, as it always has the same size. PCSS therefore uses an additional blocker search in the shadow map, so the filter kernel can be adjusted according to the relation between light, blocker and receiver. Newer approaches like Variance Shadow Maps [2] or Convolution Shadow maps [7] propose ways to pre-filter the shadow map, so the vast amount of shadow map lookups needed for PCF can be avoided. It is possible to achieve real-time frame rates with these kinds of filtering algorithms, and even though the created shadows are not physically accurate, they produce perceptually convincing results, especially since the human visual perception is not very sensitive to the geometrical correctness of soft shadows.

Several papers (e.g. Guennebaud et. al [11][12], Atty et. al [4], Aszdi & Szirmay-Kalos [3], Schwarz & Stamminger [17]) have recently been released, which all suggest to use a technique called *backprojection*. It was originally presented by Drettakis [8] in order to generate soft shadows in offline-rendering. The idea is to use a single shadow map not only for depth comparison, but to employ it as a discretized representation of the scene. In order to calculate the visibility factor v for a screen-space pixel p , the shadow map texels are backprojected from p onto the light source, where the amount of occlusion is estimated. These approaches produce physically convincing results in many cases, but are prone to artifacts (e.g. in cases when occluders overlap, when the light source is too close, or when the penumbra is extremely large) and require a costly blocker search in the shadow map.

The probably most intuitive approach to generate soft shadows is to generate hard shadows from several sampling points on the area light source and accumulate this information. If the sampling density is high enough, this idea would lead to the exact, physically correct soft shadow solution. The main problem of this approach is speed: It can be necessary to compute more than 256 hard shadows in order to obtain a smooth penumbra region with 256 possible attenuation levels in an 8Bit color channel, which leads to a long computation time and makes real-time application a hard challenge. Therefore, Heckbert & Herf [14] suggest using only a few regularly distributed samples for the calculation. For each shadow receiver, a so-called *attenuation map* is computed by summing up the individual shadows, which is then used to modify the illumination of the object. So, for n sampling points and m receivers, $m \times n$ shadow maps are required. An improvement of this idea has been suggested by Agrawala et. al [1]: Instead of calculating and using an attenuation map for each receiver, a *single layered attenuation map* for the whole scene is created, which allows interactive frame rates on modern graphics hardware. In the method proposed by St-Amour et. al [19], the visibility information of many

shadow maps is combined into a precomputed compressed 3d visibility structure, and use this structure for rendering. Sintorn et. al [18] use the CUDA capabilities on modern NVidia graphics hardware to compute and evaluate the sampled shadow maps.

The idea of using multiple sampling points on an area can also be used with modified versions of the shadow volumes algorithm which was introduced by Crow in [5]: Forest et. al [10] calculate the depth complexity of the scene with a method based on silhouette extraction and light source sampling.

Our algorithm is based on the approaches which use multiple shadow maps per light, but we propose a clever adaptive sampling strategy in order to reduce both the number of shadow maps needed to obtain high quality soft shadows and the rendering time per frame.

3 The Algorithm

In this Section, we present our new sampling strategy, which is based on adaptive refinement of the sampling positions and employs the shadow mapping algorithm for the evaluation of the shadow information. Furthermore, possible ways to render the large amounts of generated shadow maps are discussed.

3.1 The Shadow Mapping Algorithm

Shadow mapping is an image based algorithm first introduced by Williams in 1978 [20]. It is widely used in games and application to compute hard shadows, but we exploit it for the calculation of soft shadows in Section 3.2. Its basic idea is to view the scene from the position of the light source in a first pass, and store the depth values of the fragments in a texture (called the *shadow map*). The shadow map therefore contains the distances to all sampled surface points which are illuminated by the light source. Depending on the type of used light source, a perspective (for point lights) or an orthographic (for directional lights) projection has to be used. The shadow map has to be updated whenever an object in the scene or the light source moves.

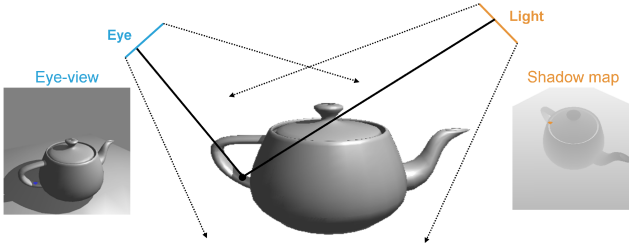


Figure 3: The shadow mapping algorithm: The depth values as seen from the light source are stored in a shadow map, and are then used in a second pass to generate shadows on the objects [16].

In the second pass, the scene is rendered from the camera's point of view. Every fragment is transformed into light space, where its distance to the light source is compared to the corresponding value in the shadow map. If the distance to the current fragment is larger than the shadow map value, it lies in shadow; otherwise it has to be illuminated by this light source. Figure 3 illustrates the basics of the algorithm.

The shadow mapping algorithm can be implemented in a fully hardware accelerated way on modern GPUs and is therefore a comparatively fast and often used method to simulate hard shadows in real-time applications and games. Furthermore, it can handle any geometry due to the fact that it is an image based approach, its speed is independent of the scene and object complexity, and self-shadowing is handled automatically. Due to the sampling of the scene during the creation of the shadow maps, aliasing and undersampling artifacts are likely to occur (see Scherzer's diploma thesis [16] for details and solutions).

3.2 Estimating Soft Shadows with Light Source Sampling

An area light source can be approximated by n different light source samples. We use the shadow mapping algorithm described in Section 3.1 to gather shadow information at each of these sampling points, as it is fast, feasible for rasterization hardware and independent of the scene complexity. A shadow map allows us to evaluate for every screen space fragment if it is illuminated by its associated point light.

$$\tau_i(x, y) = \begin{cases} 0 & \text{lit from point light } i \\ 1 & \text{in shadow of point light } i \end{cases} \quad (1)$$

$\tau_i(x, y)$ is the result of the hard shadow test for the i th shadow map for the screen space fragment at position (x, y) . Under the assumption that the point sampling on the area light source is dense enough (i.e. n is high enough), the soft shadowing result ψ (i.e., the fractional light source area occluded from the fragment) can be estimated by the proportion $\hat{\psi}_n$ of shadowed samples

$$\hat{\psi}_n(x, y) = \frac{1}{n} \sum_{i=1}^n \tau_i(x, y). \quad (2)$$

In order to create smooth penumbras in all cases, the number of sampling points n has to be quite high. This has a negative impact on both the calculation time as well as the memory consumption, making a real-time application hard to achieve. Thus, we propose the use of a new adaptive sampling strategy in Section 3.3.

3.3 Adaptive Refinement of the Sampling Density

Generating soft shadows with multiple shadow maps per light is computationally expensive due to the high sam-

pling density which is required to render visually appealing penumbra regions. If the density is too low, band artifacts are likely to appear, and the human visual system does not perceive a soft shadow anymore, but several hard shadows.

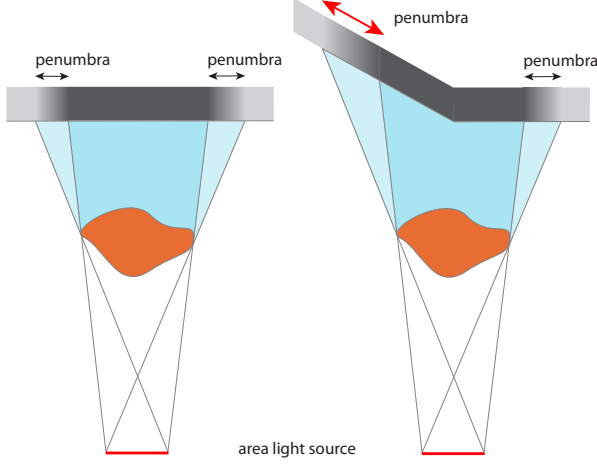


Figure 4: Already a slight change in the receiver geometry can lead to a significant increase of the penumbra size.

The larger a penumbra is, the more samples are necessary to create a smooth transitions between the individual hard shadows. The minimum required sampling density is not easy to predict, though: It depends on the relation between light, blocker and occluder. As can be seen in an example in Figure 4, a slight rotation of the receiver geometry leads to a drastic increase of the penumbra size, making more samples a necessity. Due to the perspective projection, the camera's point of view plays an important role here as well, as it determines the size of the penumbra in screen space: if the camera is very close to the shadow, the penumbra region can be as large as the whole frame buffer. So, in regular multi-sample approaches, the sampling rate has to be quite high in order to be sufficient in such worst cases.

To omit the oversampling caused by the high sampling density required for the worst case, we suggest not to use a fixed sampling rate, but to select the sampling points adaptively. Despite the additional time needed for evaluating the need for another sampling point, the overall rendering performance can be significantly increased, since the generation and evaluation of all the unnecessary shadow maps can be saved.

3.3.1 Generating Shadow Maps

The first step in our algorithm is to create the initial shadow maps at the borders of the area light source. These are the only shadow maps which are always generated; all the others are only computed if it is necessary (see Section 3.3.3). To simplify the explanation, we only use a 1-dimensional light source here (i.e. all sample points lie

on a straight line), so the sampling points are placed on both ends. For a subdivision strategy for two-dimensional light sources, see Section 3.3.5. The shadow maps are generated from these sampling points using standard uniform shadow mapping with a perspective projection.

3.3.2 Reprojection

After the creation of the initial sampling points, the shadow maps have to be projected into the same space in order to be comparable. It is important that the refinement is dependent on the observer position and view: For example, it makes no sense to refine a soft shadow which is far away and hardly visible. For shadows very close to the camera, on the other hand, it is important to have more samples in order to obtain a smooth penumbra transition. The generated shadow maps are therefore projected into camera space, where a comparison makes such a view-dependent refinement possible.

The reprojection step is done similar to the second step in the regular shadow mapping algorithm, but instead of using the stored values from the shadow map for a depth comparison, they are directly used for comparisons as described in Section 3.3.3.

3.3.3 Comparison

The comparison of two neighboring shadow maps in camera space is done in a pixel shader: The reprojected depth values of the two shadow maps are compared and considered equal if they fulfill the following condition:

$$|depth_{left}(x,y) - depth_{right}(x,y)| < \gamma, \quad (3)$$

where γ is the maximum allowed depth difference, and (x,y) are the fragment coordinates in camera space. If a fragment's shadow map values are considered equal, the pixel shader depth output is 0, otherwise it is 1. By exploiting the functionality of an *occlusion query*, which usually returns the number of pixels that pass z-testing, we can now identify the number of pixels which have been classified to be different (see Figure 5).

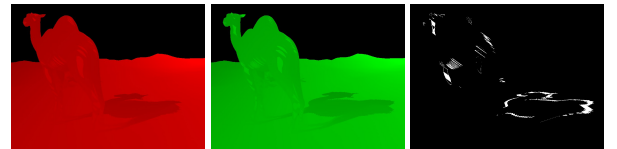


Figure 5: The comparison of two neighboring shadow maps: *Left*: The depth values of the first shadow map. *Middle*: The depth values of the second shadow map. *Right*: The pixels on which the depth values are larger than γ (See Equation 3).

3.3.4 Generating additional Sampling Points

Based on the number of differing pixels p , we evaluate a simple error metric

$$\frac{p}{SM_x \times SM_y} < \varepsilon, \quad (4)$$

where SM_x and SM_y are the shadow map dimensions, and ε is a predefined value which is used to control the sampling granularity. If the above inequation is true, the sampling density in the area of the two shadow maps is high enough. Otherwise a new sampling point is created in-between them. A higher ε therefore leads to fewer created shadow maps.

For the new sampling point, the whole procedure is repeated again: A shadow map is generated from its position and compared to its two neighbors. This refinement process is repeated until either the sampling density is high enough in all areas to fulfill the condition in inequation 4, or the maximum number of shadow maps has been created.

3.3.5 Subdivision Strategy for 2D light sources

In case a two-dimensional rectangular area light source is used instead of the linear one, we suggest using a quadtree-like structure: Starting with 4 sampling points on each corner, the shadow maps which lie diagonally opposed are compared (i.e. the top left corner shadow map is compared to the bottom right one, and the top right shadow map is compared to the lower left one) and evaluated using the Equations 3 and 4.

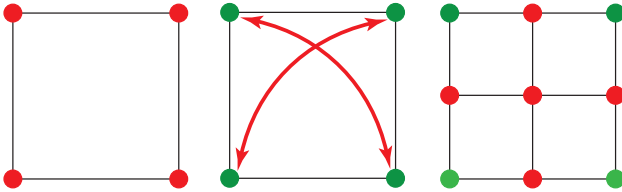


Figure 6: Subdivision strategy for area light sources: *Left*: Generate initial maps. *Middle*: Compare opposing maps. *Right*: Subdivide the quad in 4 sub-quads, repeat steps 1-3 for each sub-quad.

If at least one of the comparisons makes a subdivision necessary, the rectangle is split into 4 sub-quads, and new shadow maps are generated on all the new corners (See Figure 6). This refinement is repeated until either the sampling density is high enough, or the maximum refinement depth is reached.

3.3.6 Assigning Shadow Map Contribution Weights

If all shadow maps generated with our refinement strategy contribute to the final soft shadow solution with equal weight, the darkness of the penumbra can sometimes vary

slightly from the exact solution, if the distribution of the adaptively selected sampling points is very nonuniform. We therefore apply weights on the sampling points: In areas of many subdivision, the individual samples are assigned a smaller weight, and will not contribute as much to the darkness of the penumbra as the ones with a large weight.

In case of a linear light source, the weight ω_i assigned to the i^{th} shadow map is calculated with

$$\omega_i = \frac{1}{2^d + 1}, \quad (5)$$

and for area light sources with

$$\omega_i = \frac{1}{(2^d + 1)^2}, \quad (6)$$

where d is the subdivision depth. The sum of all weights is 1 if all samples reached the same subdivision depth d . Otherwise, the weights have to be normalized.

3.4 Evaluating the shadow map information

After having all the necessary shadow maps obtained, they have to be evaluated in an illumination render pass, which is basically similar to the second render pass in the standard shadow mapping algorithm. Still, difficulties can arise due to the large amount of depth textures which have to be sampled. We show two ways how this problem can be solved in Sections 3.4.1 and 3.4.2.

3.4.1 Soft Shadow estimation using an Accumulation Buffer and n render passes

In the standard shadow mapping algorithm, the shadow map is evaluated in a second pass rendered from the camera's point of view, and the pixels in screen space are then illuminated according to their occlusion. For the calculation of soft shadows, the information from all generated shadow maps has to be checked for each screen space pixel. The hard shadow test values (0 or 1) from all shadow maps i are multiplied with their weight ω_i and summed up, resulting in an estimate for the percentage of occlusion.

If the number of shadow maps is too high, this can lead to problems due to technical limitations: For example, in some older rendering APIs, it is not possible to sample more than 16 textures in a single rendering pass. This limits the theoretical number of shadow maps to 16; in most games and applications this number will be even lower, since most objects do usually have diffuse, normal, or other textures as well.

A way to solve this is to make use of a *Deferred Rendering System* [6] as well as a so-called *Accumulation Buffer*, which is a screen-space buffer with a single data channel. For each shadow map, we render the scene in a separate render pass, but instead of using the obtained hard shadow value of a screen space fragment $f(x, y)$ for illumination,

we multiply it with its weight and add it to the accumulation buffer at the position $f_{AccumulationBuffer}(x,y)$.

After n render passes, all shadow maps have been evaluated, and the accumulation buffer is filled. Now, in a final rendering pass, the scene is illuminated: For each screen space fragment $f(x,y)$, the corresponding Accumulation Buffer value $f_{AccumulationBuffer}(x,y)$ is sampled and used as the occlusion percentage.

Care has to be taken whenever several shadow receivers lie behind each other from the camera's point of view: their individual shadow values get summed up if no additional depth comparison is done, leading to wrong results. It is therefore necessary to render an additional depth pass from the camera's point of view into a texture with the same size as the framebuffer. During the accumulation passes, the depth value stored there is sampled for each fragment and compared to its depth, so that only fragments which are visible for the observer can have an influence on the data in the accumulation buffer.

Note: Since current graphics hardware does not support read and write operations on render targets at the same time, two instances of the accumulation buffer have to be created and swapped each rendered frame, resulting in an additional need for memory on the GPU.

3.4.2 Soft Shadow estimation using Texture arrays

In newer graphics APIs, the limitation to only 16 textures in a shader was dropped: the introduction of so-called *Texture Arrays* makes it possible to send up to 512 textures with the same size and format to the shader, where they can be sampled arbitrarily.

Using this new technique, it is not necessary to evaluate the shadow maps in multiple passes anymore: Since the shadow maps are all of the same size and format, the texture array functionality is perfectly suited for our purposes, making all maps accessible in a single shader. An array of corresponding matrices needed for the shadow map lookups is made available by using a *Constant Buffer*.

Furthermore, there is no need to use an accumulation buffer anymore: By simply summing up the hard shadows values multiplied with their corresponding weights, we obtain the fragment's occlusion value. This value can be directly used in the same pass to modify the illumination. So, even though it is still necessary to sample all shadow maps, n read/write operations as well as the memory previously consumed by the accumulation buffer and the depth buffer can be saved.

3.4.3 Filtering

In order to improve the smoothness of the transitions between the individual shadow maps, we suggest sampling them using a small PCF kernel. PCF filtering softens the shadow boundaries, and a special version with a 2×2 kernel can be used on modern graphics hardware without performance hit.

By increasing the PCF filter kernel, it is moreover possible to increase the ϵ value as well while still obtaining smooth penumbras: In such a case, fewer shadow maps are generated and used, and the time needed for rendering is reduced. This comes of course at the cost of physical accuracy, as PCF filtering blurs and widens the penumbras artificially.

4 Implementation

We implemented the method described in Section 3.4.1 in a DirectX 9 rendering framework application using a one-dimensional light source, and the method described in Section 3.4.2 in a DirectX 10 rendering framework application using a two-dimensional light source. For the shadow maps, we used 32Bit floating point textures with a size of 512^2 .

4.1 Deferred Rendering (DirectX 9)

In the DirectX 9 framework, we applied 32Bit floating point textures with the same dimensions as the frame buffer for both the accumulation buffer as well as for the needed depth buffer.

The use of the Deferred Rendering system and the accumulation buffer allow a shadow map to be evaluated as soon as it is created: Right after its generation and reprojection, the shadow information is stored in the accumulation buffer, and the texture can be re-used again in the next subdivision step. This saves a lot of memory, since not all shadow maps have to be available at the same time, making an execution on older hardware with limited resources feasible. The drawback of this approach is that it is not possible to foresee the contribution weight of the current shadow map as described in Section 3.3.6 at the time the shadow map is evaluated, making it necessary to weight all shadow maps equally. This can of course lead to physically incorrect soft shadow results.

The reprojection step was implemented in two ways for testing purposes: Apart from the projecting the shadow maps to the camera's point of view, we also tried to project them to the center of light source. With this method, the subdivision of the light source is not view-dependent anymore, but for applications with a foreseeable minimum viewing distance (e.g. a strategy game using a top-down-view), the generated penumbras are smooth enough, and a constant frame rate can be achieved more easily.

4.2 Texture Arrays (DirectX 10)

We implemented the texture array rendering method as described in Section 3.4.2 using a single render pass for the evaluation of the shadow maps in our DirectX 10 rendering framework. In the shadow maps, we store the depth linearly and use the following condition instead of the one

given in equation 3 to check if two fragments are equal:

$$\left| 1 - \frac{\text{depth}_{\text{left}}(x,y)}{\text{depth}_{\text{right}}(x,y)} \right| < \hat{\gamma}, \quad (7)$$

resulting in an error threshold that is relative to the distance.

We furthermore implemented a subdivision strategy for two-dimensional light sources using a quad tree as described in Section 3.3.5. An example image can be seen in Figure 1.

5 Results

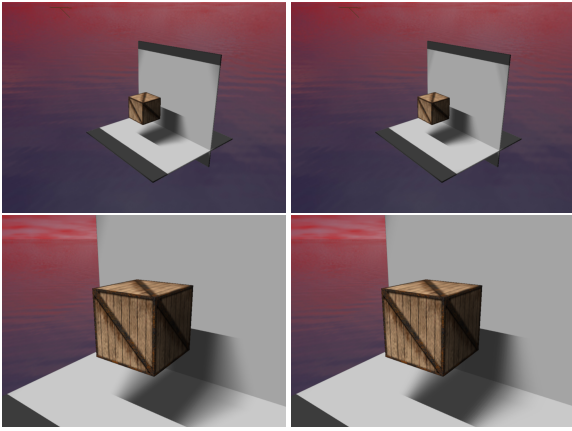


Figure 7: *Left side:* Sampling with 128 Shadow maps. *Right Side:* Our Method. For the scene in the upper row, our approach generates only 7 shadow maps and renders at more than 60 fps, while the 128 shadow maps can only be rendered at 11 fps. In the lower row, 44 shadow maps are created in our approach, but the rendering speed is still better (10 fps vs. 5 fps). Images created in the DirectX 9 framework using a linear light source and Deferred Shadowing.

All tests and images in this paper were calculated with a frame buffer size of 1024×768 , a shadow map size of 512^2 and PCF filtering turned on. The system on which we were testing our approach consisted of an AMD Athlon 64 X2 Dual Core Processor 4200 with 2GB RAM, and a NVidia Geforce 8800GT with 512 MB Memory.

The goal of our algorithm was to improve the generation of physically correct soft shadows by adaptively selecting only the light source samples which do really contribute to the visual quality of the penumbrae. In nearly all our tests, our approach performed better than a version with 128 fixed equally distributed samples (Using 128 samples results in reasonable accurate soft shadows and is still feasible in terms of performance.). In scene configurations where the penumbra regions are small, even real-time performance could be achieved; but also in cases when up to 50 shadow maps were necessary, our method was still

approximately twice as fast. Only when a penumbra is extremely large and fills a wide area of the frame buffer, the performance of our algorithm was worse, since it has to evaluate the same number of shadow maps *plus* the reprojection and comparison steps. In such a worst case, our algorithm performs at approximately 66% of the speed of the sampling approach with 128 fixed equally distributed samples.

The DirectX 10 implementation using texture arrays performed slightly better than the Deferred Rendering solution, since the shadow map evaluation can be done in a single pass, and no additional read/write operations on the accumulation buffer are needed.

As you can see in Figure 7, the visual quality of the generated shadows is very similar to the exact solution. By varying the parameters ϵ and γ as well as by using a larger PCF filter kernel, the soft shadow quality and accuracy can moreover easily be traded for better performance.

5.1 Limitations

A big drawback of our approach is the global error metric as described in Equation 4: It fails in cases when hardly any shadows are visible on the screen, as the number of differing pixels on the screen is very low in such a case. The algorithm will stop the subdivision, no matter how coarse the shadow boundaries are. Furthermore, very small light leaks could be missed due to the global error metric as well.

Even though the rendering speed can be significantly increased in most cases, our approach is very not suitable for applications where a guaranteed constant frame rate is necessary, as the number of needed samples can vary heavily.

6 Conclusions and Future Work

We presented an algorithm which is able to render physically accurate soft shadows in most cases faster than the regular light sampling method with a fixed sampling rate, since only the samples which contribute to the visual quality are computed and evaluated. The decision whether another sampling point is needed in-between two neighboring ones is being reached by reprojecting the corresponding shadow maps to the camera's point of view and comparing them there using an occlusion query. The time needed for these checks is compensated by the reduced number of shadow maps which have to be calculated.

In our test applications, we were able to render soft shadows of a quality similar to the ones generated with 128 or 256 samples, but at interactive or even real-time frame rates.

The method could be further improved by finding a better error metric for the decision whether more sampling points are needed or not; this could be for example done by using a local error metric and not a global one. We

also plan to employ more advanced filtering strategies than PCF for smooth penumbra transitions.

References

- [1] Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laurent Moll. Efficient image-based methods for rendering soft shadows. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 375–384. ACM Press/Addison-Wesley Publishing Co., 2000.
- [2] Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Convolution shadow maps. In Jan Kautz and Sumanta 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.
- [3] B. Aszdi and L. Szirmay-Kalos. Real-time soft shadows with shadow accumulation. In *Eurographics 2006 Short Presentations*, pages 53–56, 2006.
- [4] Lionel Atty, Nicolas Holzschuch, Marc Lapierre, Jean-Marc Hasenfratz, Chuck Hansen, and François Sillion. Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum*, 25(4), dec 2006.
- [5] Franklin C. Crow. Shadow algorithms for computer graphics. In James George, editor, *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, volume 11, pages 242–248. ACM Press, July 1977.
- [6] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *SIGGRAPH Comput. Graph.*, 22(4):21–30, 1988.
- [7] William Donnelly and Andrew Lauritzen. Variance shadow maps. In *In SI3D 06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, pages 161–165. Press, 2006.
- [8] George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In *SIGGRAPH '94 Proc.*, pages 223–230, 1994. <http://safran.imag.fr/Membres/George.Drettakis/pub.html>.
- [9] Randima Fernando. Percentage-closer soft shadows. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 35, New York, NY, USA, 2005. ACM.
- [10] Vincent Forest, Loc Barthe, and Mathias Paulin. Accurate Shadows by Depth Complexity Sampling. *Computer Graphics Forum, Eurographics 2008 Proceedings*, 27(2):663–674, 2008.
- [11] Gael Guennebaud, Loc Barthe, and Mathias Paulin. Real-time soft shadow mapping by backprojection. In *Eurographics Symposium on Rendering (EGSR), Nicosia, Cyprus, 26/06/2006-28/06/2006*, pages 227–234, <http://www.eg.org/>, 2006. Eurographics.
- [12] Gael Guennebaud, Loc Barthe, and Mathias Paulin. High-Quality Adaptive Soft Shadow Mapping. *Computer Graphics Forum, Eurographics 2007 proceedings*, 26(3):525–534, septembre 2007.
- [13] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. In *Eurographics*. Eurographics, Eurographics, 2003. State-of-the-Art Report.
- [14] Paul S. Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, CS Dept., Carnegie Mellon U., Jan. 1997. CMU-CS-97-104, <http://www.cs.cmu.edu/ph>.
- [15] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291. ACM Press, 1987.
- [16] Daniel Scherzer. Shadow mapping of large environments. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 8 2005.
- [17] Michael Schwarz and Marc Stamminger. Bitmask soft shadows. *Comput. Graph. Forum*, 26(3):515–524, 2007.
- [18] Erik Sintorn, Elmar Eisemann, and Ulf Assarsson. Sample-based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008)*, 27(4):1285–1292, June 2008.
- [19] Jean-François St-Amour, Eric Paquette, and Pierre Poulin. Soft shadows from extended light sources with penumbra deep shadow maps. In *Graphics Interface 2005*, pages 105–112, May 2005.
- [20] Lance Williams. Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):270–274, Aug. 1978.