Adaptive Depth Bias for Soft Shadows

Alexander Ehm
Munich University of Applied Sciences
Lothstrasse 64
80335 Munich, Germany
ehm@hm.edu

Alexander Ederer
Munich University of Applied Sciences
Lothstrasse 64
80335 Munich, Germany
aederer@hm.edu

Andreas Klein
MBDA Deutschland GmbH
Hagenauer Forst 27
86529 Schrobenhausen, Germany
andreas-herbert.klein@mbda-systems.de

Alfred Nischwitz
Munich University of Applied Sciences
Lothstrasse 64
80335 Munich, Germany
nischwitz@cs.hm.edu

Abstract

With shadow mapping the need of a suitable biasing technique due to shadow aliasing is indisputable. Dou et al. [Dou14] introduced a biasing technique that computes the optimal bias adaptively for each fragment. In this paper, we propose enhancements for this algorithm. First, we extend the algorithm for soft shadows, such as percentage closer filtering (PCF) and percentage closer soft shadows (PCSS). Second, we minimize the projective aliasing by introducing a scale factor depending on the ratio between surface and light direction. We show that our enhancements increase the shadow quality and introduce only a small overhead.

Keywords

shadow mapping, bias, adaptive bias, projective aliasing, automatic bias adjustment

1 INTRODUCTION

The most common ways to achieve interactive shadows is shadow mapping. One of the major drawbacks of shadow mapping is surface acne, which is erroneous self-shadowing. This is commonly addressed by a depth bias. There are different biasing techniques and research has been (and is still) done in this area. Most of these biasing methods suffer from two problems: first, they require hand tweaking of different parameters for different scenes, for them to work well, and second, they do not use the minimal bias and therefore cause shadow detachment.

In [Dou14], Dou et al. proposed an approach, which adaptively computes the optimal bias for each fragment. But in the original paper, the adaptive bias is only used with hard shadow mapping, and to be of common interest today, the technique should be suitable for use with soft shadowing algorithms.

For that reason we introduce an enhancement to extend the "Adaptive Depth Bias for Shadow Maps" to the soft shadowing techniques PCF and PCSS. We start with an explanation of the first approach of the extension, which turned out to be slow, and then introduce the optimized extension, which only introduces reasonable overhead. Furthermore we found that the adaptive bias still suffers heavily from projective aliasing, and therefore introduce an enhancement of the original algorithm to be more robust against this kind of aliasing. We show the detailed modifications of the algorithm for all enhancements. The main contributions of this paper are:

- Extending the adaptive bias to the soft shadowing techniques PCF and PCSS;
- Enhancing the original algorithm to be more robust against projective aliasing.

2 STATE OF THE ART

A complete overview of biasing algorithms is out of scope for this paper. In [Eis11a] a nice assembly of shadowing and biasing algorithms can be found. Although biasing methods use different approaches to obtain the bias, they all have in common that they apply an offset, the bias, to the sampled depth values to remove false self-shadowing.

The OpenGL function glPolygonOffset, is (more general speaking) the combination of a constant bias and a slope-scaled bias. Constant bias means that the same offset is used for every fragment. A slope-scaled bias, in contrast, scales the bias up the higher the slope of the surface is compared to the shadow map plane.

With PCF filtering and large filter kernels, the assumption of a single depth value for comparison across the whole filter kernel does not suffice anymore. Isidoro [Isi06a] presents a method to use an individual depth value for each sample, by using a bias based on the depth changes in the receiver plane.

Dual depth layer bias as originally proposed in [Woo94a], uses the depth difference between the closest and the second closest surface as bias. To remove the surface-acne which was left at silhouette lines, this method was improved by [Wei03a] by the addition of a constant bound to this bias. This method introduces significant extra costs, since it needs an extra rendering pass.

3 ADAPTIVE BIAS AS BASIS

We will now shortly go through the Adaptive Bias introduced by Dou et al. [Dou14], which forms the basis to this paper. The "Adaptive Bias for Shadow Mapping" is yet another biasing technique, and functions as a drop-in replacement for other biasing techniques.

The basic idea behind the adaptive bias is to calculate the optimal bias for each fragment, since the minimal amount of bias needed to remove false shadows differs for each fragment. To achieve this, firstly the potential occluder that may cause false shadow, is computed. Afterwards, a small adaptive epsilon is added to shift the current, already biased, fragment just a little closer to the light source, just above its potential occluder. For an overview of the whole algorithm see [Dou14].

3.1 Optimal Depth Bias

Shadow mapping suffers from aliasing artifacts, such as false shadows, due to the discretization of a scene into a 2d texture of depth values. If the current fragment F_c lies in the region of this shadow map texel but not exactly where the depth value was sampled, its depth value may be bigger than the depth of fragment F_o stored in the shadow map, which will cause erroneous shadow on this fragment. F_o is the potential occluder of the current texel F_c. See Figure 1 for a basic illustration.

The optimal bias is the depth difference between the current fragment F_c, and the fragment sampled for the shadow map F_o, since this bias is the minimal bias to move the fragment F_c to its occluder F_o. In order to obtain the optimal bias, the potential occluder F_o is located under the assumption that the underlying geometry is a planar surface. This can be done by computing the intersection of the light ray $\overrightarrow{R}$, which is the ray traced from the light source through the center of the corresponding shadow map texel, and the plane P, which is the tangent plane defined by the current fragment F_c and its normal N.

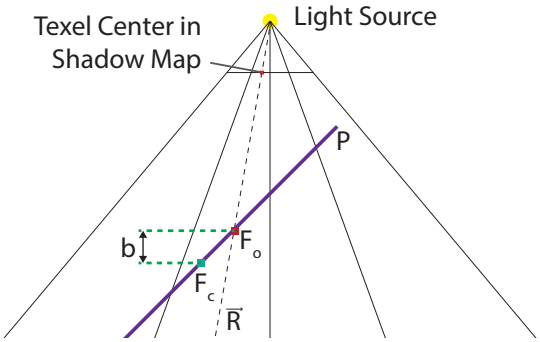


Figure 1: Illustration of the adaptive bias computation. F_c represents the current fragment, F_o is the potential occluder where this shadow map texel has been sampled. b is the optimal bias for this case.

3.2 Adaptive Epsilon

The optimal bias shifts the current fragment to its potential occluder. In order to shift it just above its potential occluder, which is desired to have a robust shadow test, a small epsilon value is needed. A constant epsilon value is not a good choice, since depth values are stored non-linearly in perspective projection.

Therefore, Dou et al. propose to use a constant epsilon but transform it based on the standard OpenGL depth compression function for perspective projections. The standard OpenGL depth compression function is the function that maps the depth values in between the near and the far clipping plane non-linear into the interval $[0,1]$.
For the constant component they recommend a value computed from *sceneScale*, which is defined as the length of the scene's bounding box diagonal and an empiric constant K. Thus the formula proposed by Dou et al. to obtain the adaptive epsilon is:

$$\varepsilon = \frac{(lf - depth \times (lf - ln))^2}{lf \times ln \times (lf - ln)} \times sceneScale \times K \quad (1)$$

With ln being the distance to the light's near plane, lf the distance to the light's far plane and depth the normalized depth value of the current fragment.

Dou et al. state, that they used $K = 0.0001$ for all their experiments. We could not prove that $K = 0.0001$ would be a good choice for different scenes. As a matter of fact, we had to modify the value of K for each scene, in order to get good results. This problem can be seen in Figure 11.

3.3 Performance

Dou et al. explain that their method is not very much slower, to be precise, around 20 %, than a constant bias. It is therefore much faster than the dual depth layer method, which they used as reference method for quality comparison, but can achieve results which are

of equal quality as those from dual depth layers. See [Dou14] for seeing the original quality comparisons. In our experiments we could prove that the adaptive bias performs well, it was 12 % slower than biasing with the glPolygonOffset function.

4 ADAPTIVE BIAS FOR SOFT SHADOWS

To be of use for today, a biasing technique needs to work with soft shadows. We will now firstly go through the combination with PCF [Ree87a] and then go through the combination of the adaptive bias with PCSS [Fer05a].

4.1 PCF

To obtain good results, it is not sufficient for PCF shadowing to use the adaptive bias of the current fragment for all shadow map texels in the filter kernel. This would lead to erroneous self-shadowing for large filter kernels. It would be best if we could compute the optimal bias for every texel in the PFC filter kernel. This is possible with some modifications to the original algorithm.

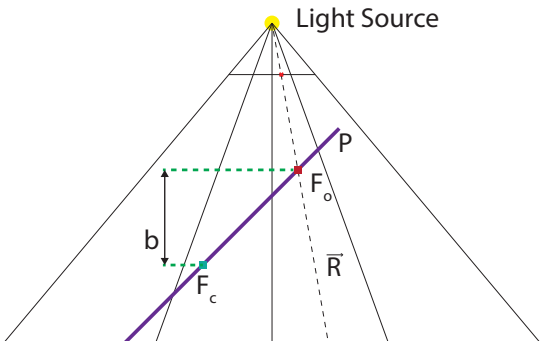


Figure 2: Illustration of the adaptive bias for PCF computation. F_c represents the current fragment, F_o is the potential occluder for the shadow map / filter kernel texel offset by 1 texel "to the right". b is then the optimal bias for this texel in the filter kernel.

Still making the simplification of taking the tangent plane defined by the current fragment and its normal as the underling geometry, we can compute the optimal bias for any texel in the PCF filter kernel. We therefore define the light ray $\overrightarrow{R}$ (which is intersected with the tangent plane P), by the light source and the texel center of the corresponding shadow map texel in the filter kernel instead of the texel center of the shadow map texel corresponding to the current fragment. See Figure 2 to see a basic illustration.

But for the PCF filtering this means, that the adaptive bias computation needs to be done for any texel in the filter kernel. This leads to a significant performance impact (For an 11x11 filter kernel, rendering time was about 5x as high as without adaptive bias. Since the computation is done for any texel in the filter kernel, it gets worse as the filter kernels get bigger.). So some optimization was needed, to be able to use the adaptive bias with PCF.

Suppose that the light direction is the same for any texel in the filter kernel. This is exactly true for an orthogonally rendered shadow map, when a directional light is used, but also holds nearly unchanged for shadow maps rendered using a perspective projection, since filter kernels do not get too large compared to the whole scene. Then the depth difference between the potential occluder for any texel in the filter kernel and the potential occluder for one texel further in x- or y-direction is the same as for any other texel in the filter kernel. And therefore, the depth difference for a texel which is offset n texels in x- or y-direction, it is the same corresponding value times n. So the bias for a texel in the filter kernel can be obtained as in equation 2.

$$bias = bias(F_c) + n \times \Delta bias_X + m \times \Delta bias_Y \quad (2)$$

If we know the depth differences, we will not have to compute the potential occluder again for each texel in the filter kernel. We could simply compute the potential occluder for the original texel, and than add the according value for the x- and y-direction multiplied by the offset - measured in texels - in x- and y-direction. See Figure 3 for clarification.

These values, from now on called $\Delta bias_X$ and $\Delta bias_Y$, can be obtained by computing the potential occluder for the original texel, and for the texels offset in x-direction and y-direction by one shadow map texel. With these three potential occluders, we can obtain the $\Delta bias$ for x- and y-direction. When PCF filtering, the potential occluder for each texel in the filter kernel is the already computed potential occluder for the original texel plus $\Delta bias_X$ times the offset in x-direction plus $\Delta bias_Y$ times the offset in y-direction. With this approach, the potential occluder needs to be calculated three times for a filter kernel, regardless of the size of the filter kernel, instead of for each texel in the kernel. This leads to the complete algorithm looking like Algorithm 1, and reduces the performance loss to reasonable 27 %.

4.2 PCSS

The whole PCSS algorithm has many similarities with PCF, since it is an extension to the PCF algorithm. On the one hand the final filtering is a PCF filtering, so the whole adaptive bias enhancement for PCF can be reused on this stage, and on the other hand the initial blocker search step does not differ from PCF in any step that is of importance for the adaptive bias enhancement, so this is no problem as well.

The blocker search step is identical to a PCF filtering, since, in a given filter kernel, the depth of the fragment

 ISBN 978-80-86943-65-7

Algorithm 1 optimized PCF with Adaptive Bias

$SM \leftarrow generateShadowMap(LightPosition)$
for *each fragment F with normal N* **do**
  $isLit \leftarrow 0$, $n_{shadowTests} \leftarrow 0$
  $F_{o_{original}} \leftarrow calculatePotentialOccluder(F)$
  $F_{o_{X+1}} \leftarrow calculatePotentialOccluder(F + (1\ texel\ in\ SM\ x-direction))$
  $F_{o_{Y+1}} \leftarrow calculatePotentialOccluder(F + (1\ texel\ in\ SM\ y-direction))$
  $\Delta bias_X \leftarrow z(F_{o_{X+1}}) - z(F_{o_{original}})$
  $\Delta bias_Y \leftarrow z(F_{o_{Y+1}}) - z(F_{o_{original}})$
  for *each texel T in filterkernel* **do**
    $F_o \leftarrow F_{o_{original}} + \Delta bias_X \times xOffset + \Delta bias_Y \times yOffset$
    $\varepsilon \leftarrow calculateAdaptiveEpsilon(F_o)$
    $isLit \leftarrow isLit + shadowTest(SM, F_o, \varepsilon)$
    $n_{shadowTests} \leftarrow n_{shadowTests} + 1$
  end for
  $isLit \leftarrow isLit / n_{shadowTests}$
  $fragColor \leftarrow isLit \times shadeFrag(F)$
end for

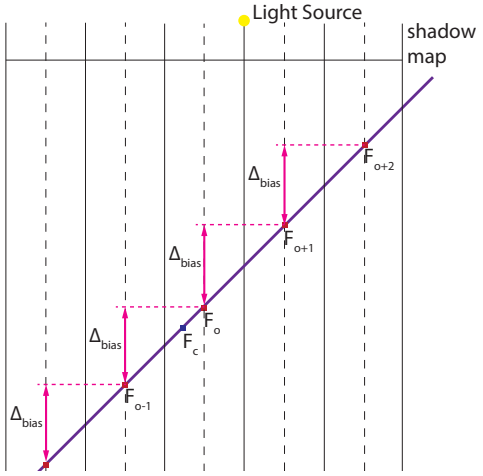


Figure 3: Illustration of the delta bias. Assuming the light direction is the same for any texel in the filter kernel, the depth difference between the potential occluders of neighboring texels is the same for any texel. (But different in x- and y-direction.)

is compared to the depth saved in the corresponding shadow map texel. There are two differences: firstly, the filter kernel for the blocker search differs from the PCF filter kernel. And secondly, not the results of the shadow tests are saved and then averaged as with PCF, but for all texels which are shaded according to the shadow test, the depth values of the blockers are collected and averaged. Since this difference has nothing to do with the biasing, the same modifications as for PCF with adaptive bias are suitable here.

Of course, the initial idea for a combination without the optimization already made with PCF, brings an even bigger performance penalty than with PCF shadows, since two PCF filters are applied for any texel in the final image. The actual PCF filtering for creating a soft shadow and the blocker search, which - in terms of PCF - depending on the light's position and size and the objects, often uses very large filter kernels.

But with the same $\Delta bias$ optimization as with PCF (and Poisson Disc sampling in the blocker search), the performance impact can be put into reasonable bounds. The $\Delta bias$ only needs to be computed once in the beginning, and can then be used for both the blocker search, and the final filtering. This does not introduce any performance loss in the blocker search, since, for standard PCSS, a receiver plane depth bias [Isi06a] should already be used in this stage to obtain clean results. And taken the $\Delta bias$ as given here, the computational effort to obtain the bias is very similar to the one that obtains the receiver plane depth bias. The final PCF filtering performs as the optimized enhancement to PCF , which it actually is, and is therefore of adequate speed. The overall performance impact is, since there is none in the blocker search, as with PCF 27%.

5 MAKING THE ADAPTIVE BIAS MORE ROBUST AGAINST PROJECTIVE ALIASING

As the sampling density of the shadow map is not adapted, the adaptive bias algorithm still suffers from projective aliasing. By increasing the value for K, the aliasing is reduced, however light leaking occurs (Figure 11a-d).

Projective aliasing appears in areas where the surface becomes parallel to the light direction. These areas can be detected with a scalar product between the light direction and the fragment normal vector. We use these scalar product to scale the adaptive bias in order to reduce the projective aliasing. This leads to the following enhanced formula for the adaptive epsilon:

$$\varepsilon = \frac{(lf-depth \times (lf-ln))^2}{lf \times ln \times (lf-ln)} \times sceneScale \times K \times scaleFactor \quad (3)$$

$$scaleFactor = min\left(\frac{1}{(lightDirection \cdot normal)^2}, 100\right) \quad (4)$$

So by using this scale factor, the epsilon gets scaled up as the surface and the light direction become more parallel. The square of the scalar product simply comes from the fact, that if we only use the scalar product, the epsilon does not scale fast enough. And introducing another constant that may not be suitable for different scenes is not a good idea. The value of 100 that limits the scaling is thus not a constant in the sense that it

needs to be hand tweaked for different scenes, it simply causes the adaptive epsilon to not scale further when the angle between light direction and surface normal is over 84.26° (this value is computed from the threshold of 100, it is not a meaningful value itself). Figure 9 shows what happens with projective aliasing artifacts for different thresholds. For too small threshold values, artifacts due to projective aliasing remain present, while overlarge thresholds produce light leaking problems. Also the range of threshold values producing good results for this scene is very large, it may be significantly smaller for other scenes. We have chosen 100 as threshold, since it firstly is, compared to the whole range of threshold values producing good results, a rather small value, so no unnecessary large scaling factors and therefore unnecessary large biases are used. Secondly 100 has proven itself as a good value for all scenes we used for testing (any image in this paper is rendered with 100 as threshold), also for the scenes for which the range of thresholds giving good results is much smaller than in the above example. Even though the value of 100 worked for all our scenes, there is no guarantee it would work for any scene, so it still needs to be hand tweaked. If we did not limit the scaling factor (or if a overlarge threshold is used), it would get very large, leading to visible strips of light in these areas, since we would use a very large bias due to the large factor, which would additionally stack with the light leaking problem which already comes with the original bias. Note that this does not make the epsilon completely adaptive. The value of K has still to be adapted for each scene. However, the range of scenes where a given value of K performs well is extended compared to the original definition in [Dou14].

6 RESULTS AND DISCUSSION

We will now show results and performance of our methods. The implementation was done with OpenGL and GLSL shaders. The scenes were rendered on an Intel Core i7 with 4 GHz and a NVIDIA GTX770 graphics card. All images where rendered with a resolution of 1024x768. We will follow the same order as before, and go firstly through the results of the combination of the adaptive bias and PCF, then the combination with PCSS and finally show the results from the enhancement against projective aliasing.

6.1 PCF

Since Dou et al. explicate in [Dou14] how the adaptive bias preserves more shadow detail than other biasing methods and therefore increases the quality of the shadows, we solely focus on the quality of the PCF shadow, and therefore use a suitable scene.

A naive PCF implementation is used, meaning any texel in the filter kernel is sampled. Figure 4 shows the comparison between PCF with receiver plane depth bias, the adaptive bias and our optimized adaptive bias for PCF. The difference image in Figure 4 shows that there are small differences in the self shadow, however no differences in the shadows cast on the plane. In Figure 5 a comparison between a PCF shadow - emerging from being on the backside (from the light's point of view) of an object - with receiver plane depth bias and the optimized adaptive bias can be seen. In this case the optimized adaptive bias gives a significantly better result. Furthermore Figure 8c demonstrates that also with a nonplanar shadow receiver, our combination of PCF and the adaptive bias produces good results. In addition in figure 10 you can see a comparison of PCF with receiver plane depth bias and PCF with our adaptive bias on the complex sponza scene. Due to the left artifacts and the less preserved shadow detail, PCF with our adaptive bias clearly outperforms PCF with receiver plane depth bias.

Table 1 shows the corresponding performance for the scene in the above mentioned Figure 4. As displayed, the adaptive bias is extremely slow compared to the receiver plane depth bias. But the optimized adaptive bias brings the performance back into reasonable boundaries, since the optimized adaptive bias does not cost more than 27% more overall rendering time compared to the receiver plane depth bias, depending on the filter kernel size.

For the performance comparison with different shadow map resolutions, see Figure 7b. As you can see the rendering time of both, the adaptive bias, and of course the more interesting optimized one, comparatively do not increase faster, instead, the rendering times of the different biasing methods converge for higher shadow map resolutions.

6.2 PCSS

The PCSS implementation without adaptive bias uses a receiver plane depth bias, for both the blocker search and the final PCF filtering. The PCSS with adaptive bias uses the optimized adaptive bias on both stages. The not optimized adaptive bias is excluded in this stage, since the PCF section proves that the optimized one produces equally good results, and PCSS with the unoptimized version is extremely slow. Both implementations use Poisson Disc sampling with 25 samples in the blocker search, and a naive PCF implementation for the final filtering.

Figure 6 shows a comparison of PCSS with and without the adaptive bias. And again, you can clearly see, that all results are equally good, which proves that not only the PCF enhancement, but also the enhancement of the adaptive bias for PCSS, works. The difference image shows some minor differences at the shadow boundaries, resulting from the blocker search, and which are not even visible with the naked eye. Table 2 shows the

corresponding performance measures. The optimized adaptive bias is of reasonable performance, as it is again about 27% slower than the receiver plane depth bias version.

Figure 7a shows the performance under different shadow map resolutions. As you can see, the performance of PCSS with the optimized adaptive bias does not lag a lot behind the performance of PCSS with receiver plane depth bias and scales equally well. Actually, for higher shadow map resolutions, the computational extra cost reduces, as for a shadow map resolution of 8192^2 the optimized adaptive bias is only about 12% more expensive.

6.3 Making the adaptive bias more robust against projective aliasing

Figure 8 shows a simple scene, that demonstrates where projective aliasing causes problems with the original [Dou14] algorithm, and that the enhancement against projective aliasing is able to remove these artifacts. In Figure 11 the complex island scene is pictured with and without the enhancement against projective aliasing and rendered by a ray tracer as reference, and it is clearly visible that it looks a lot better with the enhancement. There are falsely lit points, but as you can see in the picture with the original adaptive epsilon, they mostly come from the light leaking problem that already comes with the adaptive bias algorithm. Some very few additional light points are introduced by the enhanced adaptive epsilon. These are so scarce, that, assuming that the light leaking problem that comes with the original algorithm is not a big issue as stated in [Dou14], we claim that this is still no problem. Additionally Figure 10 shows the sponza scene, as another complex scene. As in Figure 11, the result of rendering with the enhancement against projective aliasing looks much cleaner.

In the comparison of the different views from the island scene in Figure 11, you can see that now, while the "backside" of the scene where projective aliasing still was a huge problem looks a lot better, we have no problem with shadow detachment on the "frontside". This was not possible without the enhanced adaptive epsilon, since, with the original adaptive epsilon, a constant value of K that was large enough to remove the projective aliasing already caused shadow detachment in other parts of the scene, and the other way around, a value that did not cause any shadow detachment left projective aliasing.

7 CONCLUSION

In this paper, we made the adaptive bias algorithm of Dou et al. more robust against projective aliasing and presented a strategy for incorporating it into soft shadow algorithms such as PCF and PCSS. Our idea is to calculate the potential occluder only once for each texel and interpolate it for the kernel offsets of a PCF filter. This results only in a small performance penalty compared to the receiver plane depth bias.

Furthermore, we extended the adaptive bias algorithm with a light dependent factor in order to make it more robust against projective aliasing. However, the sampling density of the shadow map is not increase and therefore, projective aliasing is still present. In order to reduce the projective aliasing further, adaptive partitioning approaches, such as queried virtual shadow maps [Gie07a], are required.

In future work, we wish to replace scene dependent parameters, such as the scene scale, with scene independent parameters in order to avoid parameter tweaking for multiple scenes. Furthermore, we wish to increase the performance of the technique when used in soft shadow algorithms.

8 REFERENCES

[Dou14] Dou, H., Yan, Y., Kerzner, E., Dai, Z., and Wyman, C. Adaptive depth bias for shadow maps. In Conf.proc ACM Symposium on Interactive 3D Graphics and Games. 2014

[Eis11a] Eisemann E., Schwarz M., Assarsson U., Wimmer M. Real-Time Shadows, Taylor & Francis, 2011.

[Fer05a] Fernando R. Percentage-Closer Soft Shadows. ACM SIGGRAPH 2005 Sketches, 2005.

[Gie07a] Giegl, M., and Wimmer, M Queried virtual shadow maps. In Conf.proc ACM Symposium on Interactive 3D Graphics and Games, 2007.

[Isi06a] Isidoro J. R. Shadow Mapping GPU-based Tips and Techniques. GDC 2006, 2006.

[Ree87a] Reeves W. T., Salesin D. H., Cook R. L. Rendering antialiased shadows with depth maps. In Conf.proc SIGGRAPH '87, ACM, 283-291, 1987.

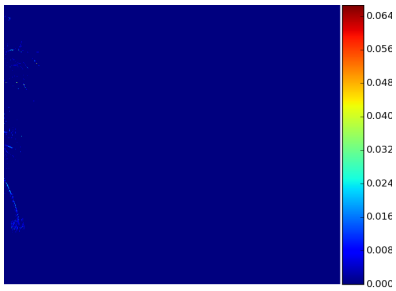
[Wei03a] Weiskopf, D., and Ertl, T. Shadow mapping based on dual depth layers. In Conf.Proc. Eurographics, ACM, 173-180, 2003.

[Woo94a] Woo, A. The shadow depth map revisited. In Graphics Gems III, Morgan Kaufmann, 338–342, 1994.

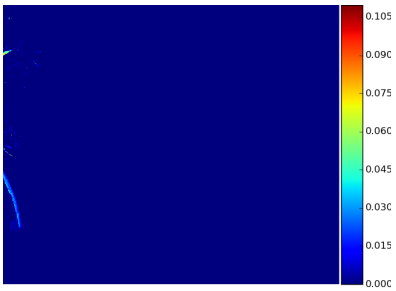
(a) receiver plane depth bias

(b) adaptive bias

(c) optimized adaptive bias

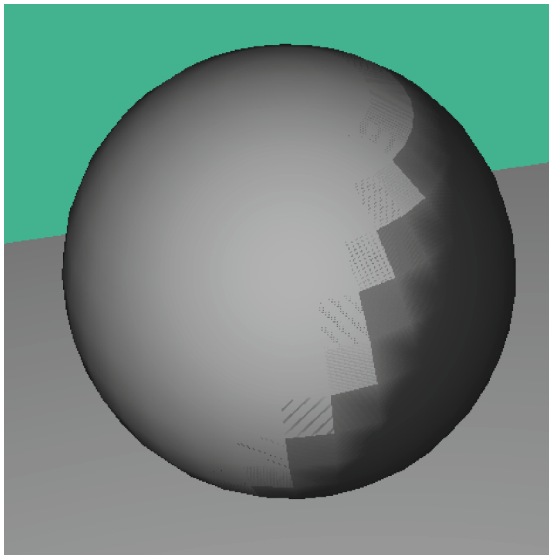


(d) difference image between (a) and (b)

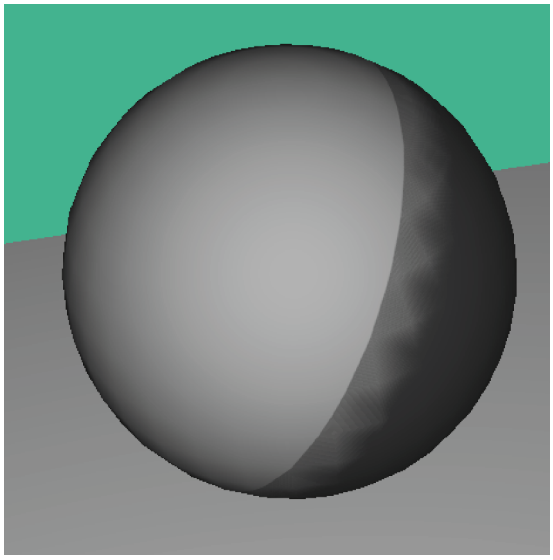


(e) difference image between (a) and (c)

Figure 4: Comparison between a PCF shadow (which is actually cast) with an 11x11 filter kernel and with (a) receiver plane depth bias, (b) adaptive bias and (c) the optimized adaptive bias. The difference image between the receiver plane depth bias and the adaptive bias is shown in (d) or the optimized adaptive bias in (e). There is no difference, which means the quality of our approach is at least as good as these approaches.



(a) receiver plane depth bias



(b) optimized adaptive bias

Figure 5: Comparison between a PCF shadow (emerging from being on the side turned away from the light) and with (a) receiver plane depth bias and (b) the optimized adaptive bias. The optimized adaptive bias gives a significantly better result.

(a) receiver plane depth bias (b) optimized adaptive bias (c) optimized adaptive bias

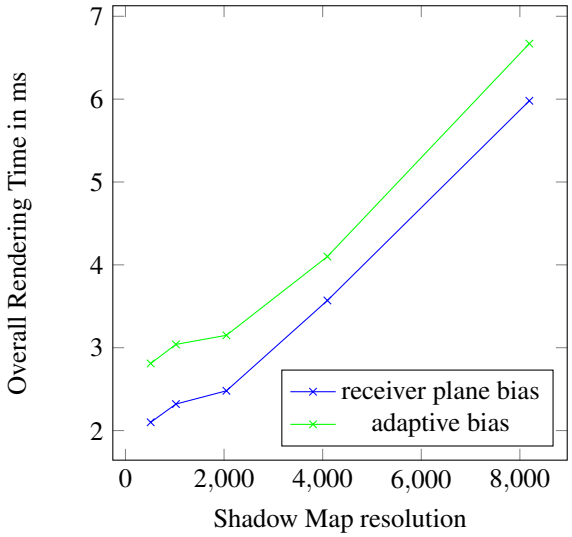
Figure 6: Comparison between PCSS with (a) receiver plane depth bias and (b) optimized adaptive bias. In (c) the difference image can be seen. There are very few differences at the shadow outlines, still resulting in a shadow of equal quality.

biasing method	kernel	Shadow Map	Final Shading	Overall
receiver plane depth bias	5x5	0.44ms	0.74ms	1.18ms
adaptive bias		0.44ms	1.38ms	1.82ms
optimized adaptive bias		0.44ms	0.86ms	1.30ms
receiver plane depth bias	7x7	0.44ms	1.05ms	1.49ms
adaptive bias		0.44ms	2.49ms	2.93ms
optimized adaptive bias		0.44ms	1.38ms	1.82ms
receiver plane depth bias	11x11	0.44ms	2.27ms	2.71ms
adaptive bias		0.44ms	5.75ms	6.19ms
optimized adaptive bias		0.44ms	3.01ms	3.45ms

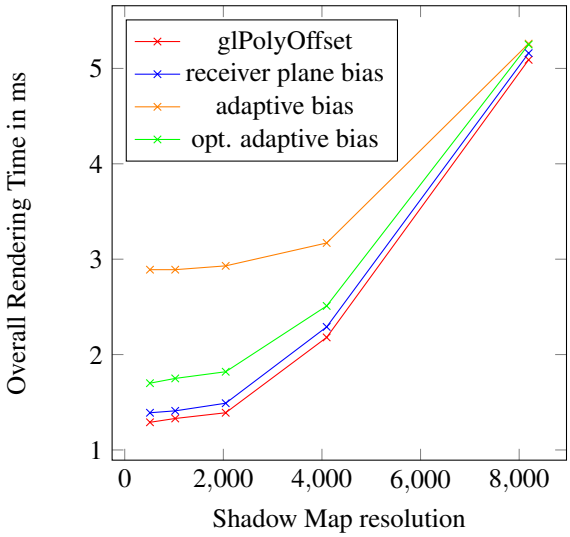
Table 1: Performance measurement of PCF with different filter kernel sizes. The shadow map resolution was constantly 2048^2.

biasing method	light size	Shadow Map	Final Shading	Overall
receiver plane depth bias	small (0.05)	0.44ms	3.61ms	4.05ms
adaptive bias		0.44ms	5.11ms	5.55ms
plane bias & poisson		0.44ms	1.92ms	2.36ms
adaptive bias & poisson		0.44ms	2.57ms	3.01ms
receiver plane depth bias	medium (0.10)	0.44ms	13.65	14.09ms
adaptive bias		0.44ms	19.10ms	19.54ms
plane bias & poisson		0.44ms	2.04ms	2.48ms
adaptive bias & poisson		0.44ms	2.71ms	3.15ms
receiver plane depth bias	large (0.16)	0.44ms	34.74ms	35.18ms
adaptive bias		0.44ms	48.51	48.95ms
plane bias & poisson		0.44ms	2.17ms	2.61ms
adaptive bias & poisson		0.44ms	2.86ms	3.30ms

Table 2: Performance measurement of PCSS with different light sizes in the tree scene. The shadow map resolution was 2048^2 for all measurements. Receiver plane depth bias according to [Isi06a]. If not specifically mentioned naive sampling is used, meaning that any texel in the filter kernel was sampled, poisson means Poisson disk sampling was used.

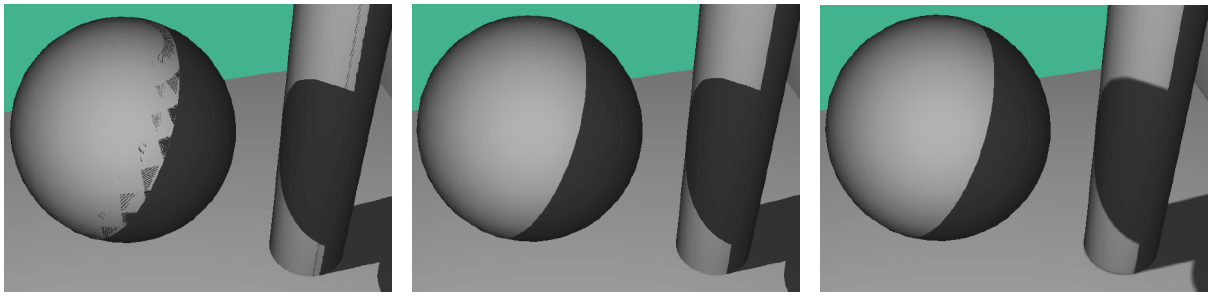


(a) Performance of PCSS with optimized adaptive bias and receiver plane depth bias with Poisson Disc sampling under different shadow map resolutions.



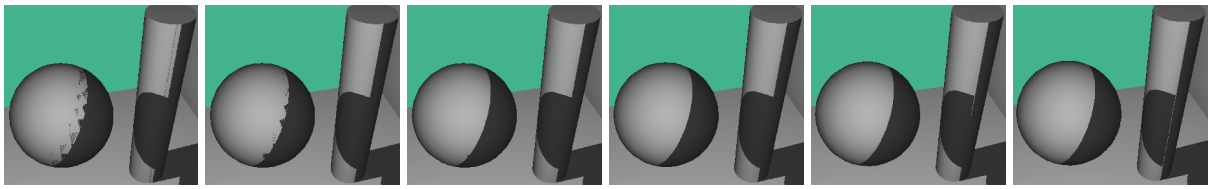
(b) PCF Performance with different biasing and varying shadow map resolutions.

 ISBN 978-80-86943-65-7



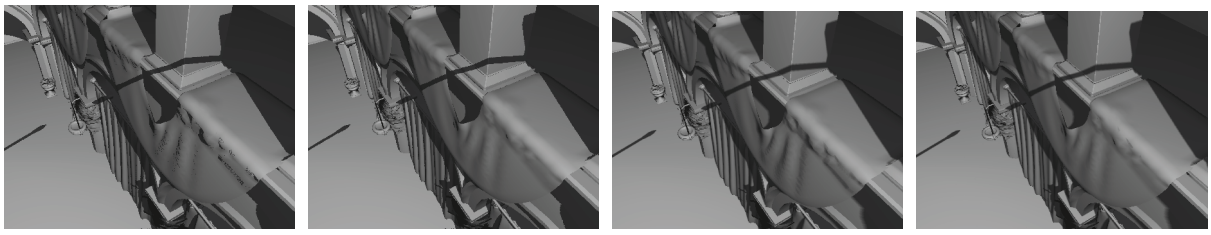
(a) original [Dou14] algorithm (b) enhanced adaptive epsilon (c) enhanced adaptive epsilon with PCF (19x19 filter kernel)

Figure 8: Casted shadow on a nonplanar surface. While the original [Dou14] algorithm (a) still has artifacts, the enhanced adaptive epsilon (b) has no visible arifacts, and generates good results on the nonplanar shadow-receiver, even with PCF filtering with large filter kernels (c).



(a) threshold of 1 (b) threshold of 10 (c) threshold of 50 (d) threshold of 100 (e) threshold of 1700 (f) threshold of 2000

Figure 9: Different threshold values for the scale factor, from very low values to very high values. It is clearly visible, that a too small threshold results in remaining artifacts, while overlarge values results in lightleaking. These values cover a very large range in which good results are produced for this scene, but this range might be significantly smaller for other scenes.

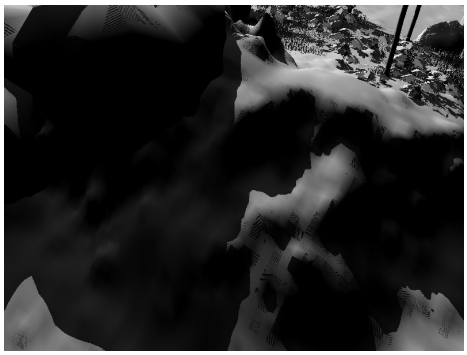


(a) original [Dou14] algorithm (b) enhanced adpative epsilon (c) PCF with receiver plane depth bias (d) PCF with our adaptive bias

Figure 10: Complex sponza scene for comparison. Comparison of the original [Dou14] algorithm (a) and the enhanced adaptive epsilon (b). The original algorithm still suffers from projective aliasing, while the enhanced adaptive epsilon creates a satisfying result. Comparison of PCF with receiver plane depth bias (c) and PCF with our adaptive bias (d). With the receiver plane depth bias there are still artifacts left, while less shadow detail is preserved.

(a) original adaptive epsilon, $K = 0.0001$

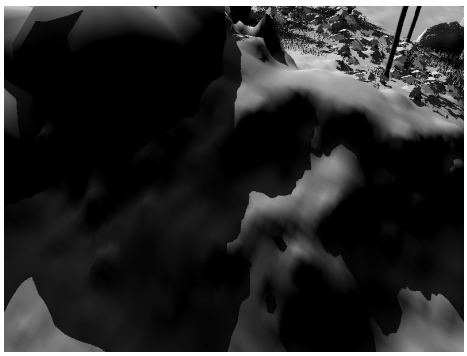
(b) tree trunk



(c) original adaptive epsilon, $K = 0.0001$

(d) original adaptive epsilon, $K = 0.01$

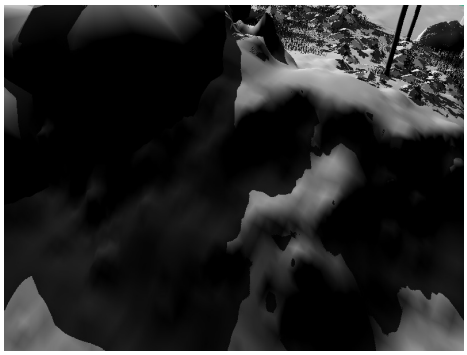
(e) tree trunk



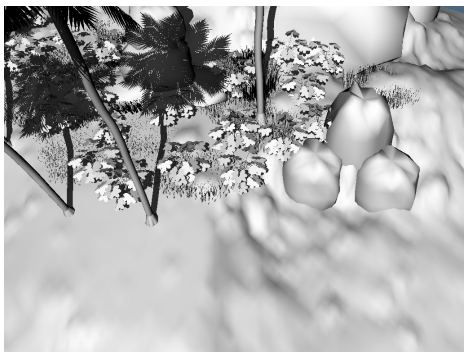
(f) original adaptive epsilon, $K = 0.01$

(g) enhanced adaptive epsilon, $K = 0.0001$

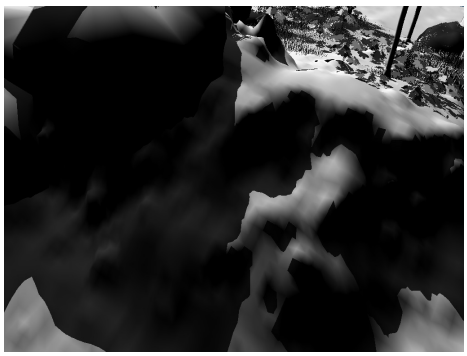
(h) tree trunk



(i) enhanced adaptive epsilon, $K = 0.0001$



(j) ray tracing result



(k) ray tracing result

Figure 11: The island scene, with in (a) - (c) the original epsilon producing good results in (a) but with projective aliasing on (c) due to the grazing angle of the light. In (d) - (f) the original formula is still used, but with $K = 0.01$ which reduces the projective aliasing in (f), but causes shadow detachment (tree trunk shadows) in (d). In (g) - (i) the enhanced adaptive epsilon is used, causing good results for the same value of $K = 0.0001$. In (j) and (k) are rendered with a ray tracer as a reference implementation.

 ISBN 978-80-86943-65-7