Real-Time Soft Shadows Using Temporal Coherence

Daniel Scherzer*, Michael Schwärzler**, Oliver Mattausch*, and Michael Wimmer*

*Vienna University of Technology and **VRVis Research Company

Abstract. A vast amount of soft shadow map algorithms have been presented in recent years. Most use a single sample hard shadow map together with some clever filtering technique to calculate perceptually or even physically plausible soft shadows.
On the other hand there is the class of much slower algorithms that calculate physically correct soft shadows by taking and combining many samples of the light.
In this paper we present a new soft shadow method that combines the benefits of these approaches. It samples the light source over multiple frames instead of a single frame, creating only a single shadow map each frame. Where temporal coherence is low we use spatial filtering to estimate additional samples to create correct and very fast soft shadows.

1 Introduction

Shadows are widely acknowledged to be one of the global lighting effects with the most impact on scene perception. They are perceived as a natural part of a scene and give important cues about the spatial relationship of objects. In reality most light sources are area light sources and these create soft shadows. We are not used to hard shadows and perceive them as distinct objects. For the realistic shadowing of a scene, soft shadows are therefore considered a must. Soft shadows consist of an umbra region where the light source is totally invisible and a penumbra region where only part of the light source is visible.

Typical soft shadowing methods for real-time applications approximate an area light by a point light located at its center and use heuristics to estimate penumbrae, which leads to soft shadows that are not physically correct [1, 2]. This is because the area visibility that is the result of an area light interacting with a scene is replaced by a simpler from point visibility.

As the human eye is not very sensitive to the correctness of soft shadows, the results can be acceptable from a perceptual point of view or can even be physically plausible. Additionally most inherent shadow map artifacts like aliasing are often hidden through the low frequency soft shadows. Nevertheless some perceptually concerning artifacts remain: Overlapping occluders can lead to unnatural looking shadow edges, or large penumbrae can cause single sample soft shadow approaches to either break down or become very slow (see Figure 1, 5).

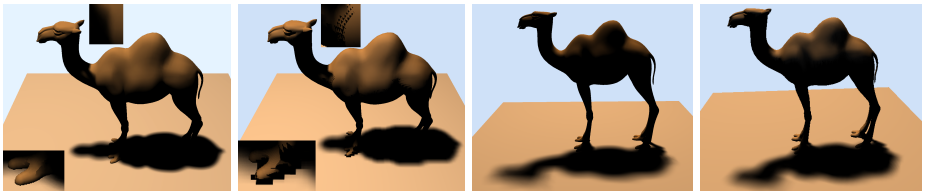

Fig. 1. *From left to right*: our Method (634FPS), bitmask soft shadows with a 8x8 search area (156 FPS), our Method with a bigger penumbra (630FPS) and bitmask soft shadows with the same penumbra and a 12x12 search area (60FPS). Even very good single sample soft shadow methods show some artifacts, like biasing problems and contact shadow undersampling that can be avoided by using multiple samples.

Accurate methods use light source sampling: The idea is to calculate soft shadows by sampling the area of the light source. Hard shadow calculations are performed for every sample and the results are combined [3]. The primary problem of these methods is that the number of samples to produce smooth penumbrae is huge. N samples produce $N-1$ attenuation levels. High-quality soft shadows need 256 or more to create the 256 attenuation levels available with an $8Bit$ color channel. If we have to render a shadow map for each sample and store all the shadow maps for the final combination pass, real-time frame rates become unlikely, even for simple scenes. But under the assumption of a light source with uniform color and dense enough sampling of the light source, the result of these approaches are correct soft shadows.

Our approach can be described as a combination of light source area sampling over time and single sample filtering:

- The area sampling is done one sample per frame by creating a shadow map from a randomly selected position on the area light. For each screen pixel the hard shadow results obtained from this shadow map are combined with the results from previous frames (accumulated in a screen space buffer called the *shadow buffer*) to calculate the soft shadow for each pixel.
- When a pixel becomes newly visible and therefore no previous information is available in the shadow buffer, we use a fast single sample approach (PCSS with a fixed 4x4 kernel) to generate an initial soft shadow estimation for this pixel.
- To avoid discontinuities between sampled and estimated soft shadows, all the estimated pixels are augmented by using a depth-aware spatial filter to take their neighborhood in the shadow buffer into account.

The main contribution of this paper is the application of temporal coherence (through the use of the shadow buffer) to the soft shadowing problem together with spatial filtering (penumbra estimation and pixel neighborhood) in a soft shadow mapping algorithm that is even faster than a very fast variant of PCSS (see Section 4 for details), but produces *accurate real-time* soft shadows. Ad-

ditionally we extend temporal reprojection to handle moving objects such as dynamic shadow casters and receivers.

2 Previous Work

Soft shadows are a very active research topic, therefore we can only give a brief overview of the most relevant publications for our work. A still valuable survey of a number of different soft shadow methods is due to Hasenfratz et al. [1]. There are two major paradigms: methods that use shadow volumes (object-based algorithms) and methods that use shadow maps (image-based algorithms). We concentrate entirely on algorithms that employ shadow mapping due to their higher performance in real-time applications.

Filtering: Since physically based soft shadow mapping requires many light source samples, it was previously considered too costly for real-time rendering. Therefore a number of algorithms were proposed that offer cheaper approximations. Most popular among those is Fernando et al.'s [4] Percentage Closer Soft Shadows (PCSS), which estimates the soft shadow from a single sample and employs a blocker search to estimate the penumbra width and then uses PCF-filtering accordingly to soften the shadows. Very fast speeds can be achieved by using small fixed sized kernels and only adapting the sample spacing to the penumbra estimation. Unfortunately this introduces artifacts for large penumbrae (see Figure 5). We will use this approach for our initial guess for freshly disoccluded fragments due to its speed. Annen et al. use a weighted summation of basis terms [5]. Using a variation of convolution shadow maps and the average blocker search from Fernando et al. [4], Annen et al. [6] extract area light sources from environment maps.

Back projection: A whole class of methods use back projection to get a more physically based estimation of the soft shadow [7–9, 2]. These methods treat the shadow map as a discrete approximation of the blocker geometry. By back projecting shadow map samples onto the light source, an accurate calculation of the percentage of light source visibility can be done. Convincing results are produced, but a region (related to the size of the penumbra) of the shadow map has to be sampled for each fragment, which becomes costly for large penumbrae.

Sampling: Maybe the most straightforward approach to computing soft shadows is by sampling an area or volume light source. Such methods are mostly targeted at off-line or interactive rendering. Heckbert and Herf [3] propose to sample the light source at random positions, render the scene and accumulate the results. Our approach moves this method into the real-time domain by exploiting temporal coherence. Agrawala et al. [10] create a layered attenuation map out of the shadow maps, which allows interactive frame rates. St-Amour et al. [11] combine the visibility information of many shadow maps into a compressed 3d visibility structure which is precomputed, and use this structure for rendering.

Temporal reprojection: Finally, temporal reprojection was used by Scherzer et al. [12] to improve the quality of hard shadow mapping and by Velázquez-

Armendáriz [13] and Nehab et al. [14] in a more general way to accelerate real-time shading. Our algorithm takes up the idea of temporal reprojection and uses it to compute shadows from area light sources.

3 The Algorithm

If we want to find the contribution of an area light source to a specific fragment, we have to calculate the fraction of the area of the light source that is visible from the fragment. For our purposes we will use the reverse value, which we call the occlusion percentage or soft shadow result $\psi(x, y)$ for a fragment at screen space position (x, y). $\psi(x, y)$ is 0 for a fragment that is illuminated by the whole area of the light source and 1 for a fragment that is not illuminated by the light source at all. Due to the fact that most calculations we perform are done per fragment and for the sake of notational simplicity, we will only use the (x, y) notation when introducing a function and will afterward omit it.

3.1 Estimating Soft Shadows from n Samples

To make the calculation of the soft shadow value feasible for rasterization hardware, we use sampling and shadow maps. We approximate an area light source by n different point light sources and compute a shadow map for each of them. A shadow map allows us to evaluate for every screen space fragment if it is illuminated by its associated point light.

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

$\tau_i(x, y)$ is the result of the hard shadow test for the ith shadow map for the screen space fragment at position (x, y). Under the assumption that our point light placement on the area light source is sufficiently random, 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

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

The number of shadowed samples $n\hat{\psi}_n$ has a Binomial distribution with variance $n\psi(1 - \psi)$. We can thus give an unbiased estimator for the variance of the proportion $\hat{\psi}_n$ as

$$\hat{var}(\hat{\psi}_n(x, y)) = \frac{\hat{\psi}_n(x, y)(1 - \hat{\psi}_n(x, y))}{n - 1} \tag{3}$$

The importance of Equation 3 is that it allows us to estimate the quality of our soft shadow solution after taking n samples. We will later use this estimate (the standard error derived from this estimate) to judge if sampling alone will give

sufficient quality. Table 1 shows these formulas applied to some real-world τ_i and increasing sample sizes. Please note that although the standard error decreases when the sample size is increased, the estimator for the standard error is not guaranteed to do so.

Table 1. Evaluation of the presented formulas for one fragment. Increasing the sample size generally reduces variance and standard error, $\hat{s} = \sqrt{v\hat{a}r}$

n	1	2	3	4	5	6	7
τ_n	1	0	1	1	1	0	1
$\hat{\psi}_n$	1.00	0.50	0.67	0.75	0.80	0.67	0.71
$v\hat{a}r(\hat{\psi}_n)$	0	0.25	0.11	0.06	0.04	0.04	0.03
$\hat{s}$	0	0.50	0.33	0.25	0.20	0.21	0.18

3.2 Temporal Coherence

We want to be able to solve Equation 2 iteratively, so that we only need to create one shadow map each frame. We will then use the temporal coherence of the current frame with previous frames to increase convergence.

The temporal coherence method introduced by Scherzer et al. [12] improves the resolution of standard hard shadow maps. It is based on the assumption that most screen space fragments stay the same from frame to frame. It can be determined which fragments of the new frame have also been present in the last frame by reprojecting (to account for camera movement) fragments from the new frame into the old and comparing their respective depths. If the depth difference is smaller than a predefined ϵ, the two fragments are considered equal and therefore fragment data from the previous frame is reused. If $d_k(x, y)$ is the depth of the fragment at position (x, y) of the current frame and $d_{k-1}(x, y)$ is the same for the previous frame, then the test for fragment equality is given by

$$|d_k(x, y) - d_{k-1}(x, y)| < \epsilon \tag{4}$$

If on the other hand the difference is greater, the fragment was not present in the last frame and is therefore new (a.k.a. disoccluded) and no previous data for this fragment is available.

For our approach we want to be able to compute $\hat{\psi}_n$ iteratively for each frame and fragment from the information saved from previous frames together with the information gathered for the current frame. We do this by keeping $\rho_n(x, y) := \sum_{i=1}^{n} \tau_i(x, y)$ from the previous frame. ρ_n stores all the shadow map tests already performed for the n previously calculated shadow maps. We also need the sample size n, which is equal to the number of shadow map tests we have already performed for this fragment. If the fragment was occluded in the last frame, we get $n = 0$ and $\rho_n(x, y) = 0$ because no previous information is

available. Therefore n can be different for each screen space fragment. We can now calculate $\hat{\psi}_{cur}(x, y)$ for the current frame as

$$\hat{\psi}_{cur}(x, y) = \frac{\tau_{cur}(x, y) + \rho_n(x, y)}{n(x, y) + 1} \tag{5}$$

This formula only needs access to n (the count of samples available for this fragment stored in the previous frame), ρ_n (i.e. sum of the shadow map tests up to the previous frame) and the current shadow map. We provide access to these values by storing in each frame the updated sample size $n + 1$ and $\tau_{cur} + \rho_n$ for every fragment into a screen sized off-screen buffer called the *shadow buffer* for use in the next frame. Now this formula can be evaluated very quickly in a fragment shader and real-time frame rates pose no problem. Note that we also have to store the depth of each fragment to be able to evaluate the test in Equation 4.

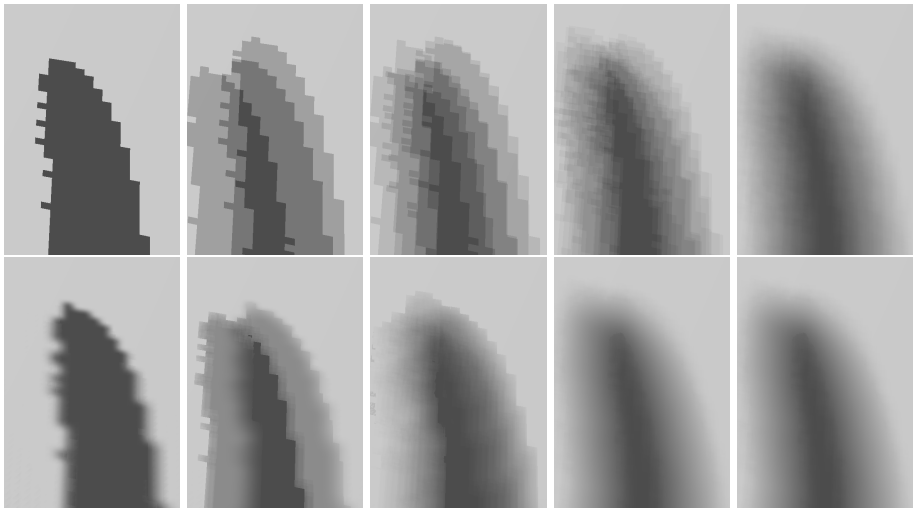

Fig. 2. Convergence after 1,3,7,20 and 256 frames. *Upper Row*: Sampling of the light source one sample per frame (using Equation 5); *Lower Row*: Our new algorithm.

The disadvantage of this approach is that for newly disoccluded fragments (i.e. fragments with a small n) the results have large errors (see also Figure 2). This becomes clearer when we take a look at the example in Table 1: At $n = 2$ we have a standard error of 0.5, which means the real ψ is probably inside of 0.5 ± 0.5, so the quality of $\hat{\psi}_2$ as an estimate is really bad. A second closely connected problem that aggravates the situation even further is discontinuity in time – the difference between $\hat{\psi}_n$ and $\hat{\psi}_{n+1}$ is still large. For instance $\left|\hat{\psi}_1 - \hat{\psi}_2\right| = 0.5$ which means the soft shadow results can be 128 attenuation values apart. This is a

noticeable jump and will cause flickering artifacts. In the next section we will show how to avoid this by using spatial filtering.

3.3 Spatial Filtering

Due to the use of temporal coherence and Equation 5 we have already constructed a soft shadow algorithm that takes little time to evaluate each frame, but two closely related problems remain:

- For a good estimation of ψ with $\hat{\psi}_n$ for a fragment with temporal coherence alone we need the fragment to be visible for many frames.
- During the frames following the disocclusion of a fragment, $\hat{\psi}_n(x, y)$ has a large standard error and may change drastically, resulting in flickering of these fragments (see also Figure 2, upper row).

Our first observation in this respect is that for the first few frames after a fragment has been disoccluded, a single sample soft shadow mapping approach will probably have better quality than using $\hat{\psi}_n$ (a.k.a the sampled approach), which is also suggested by the high variance in this case. So our first improvement over using only Equation 5 is to use a very fast single sample soft shadow approach, PCSS, as a starting point and then refine it by using Equation 5 (see Figure 3). Note that PCSS itself is also a spatial filter.

For the refinement, we have to initialize n and ρ_n for the sample generated using PCSS. The PCSS shadow test will give us a result in the range [0..1]. The most natural choice is here to use this result directly as ρ_n and set $n = 1$. Note that it can make sense to use higher values for n, meaning that the PCSS sample will be of greater weight than the following "normal" samples. Using bigger values for n can lead to faster convergence if the PCSS result is near to the correct solution (see Figure 3). This approach can considerably shorten the period a fragment has to be visible to achieve a good estimation of ψ with $\hat{\psi}_n$. Please note that other single sample soft shadow approaches could also be used together with our sampling approach. We have chosen PCSS mainly for its speed.

For small n, each new sample potentially causes drastic changes to the estimated soft shadow solution. On the one hand these changes need to happen to guarantee a swift convergence, but on the other hand we want to avoid the resulting flickering artifacts in the rendered shadows. Therefore we propose to introduce an additional smoothing by using a *neighborhood filter* in screen space just for the rendered shadows, without any impact on the soft shadow information (namely $n+1$ and ρ_{n+1}) stored in the shadow buffer for the next frame (see Figure 2, lower row for results). Section 3.3 will describe in detail how this filter is constructed.

We still have to decide when to apply the neighborhood filtering. For this we use the standard error of the variance estimator of Equation 3. This gives us an estimated error for our sampling approach. We choose an error threshold t down to which neighborhood filtering will be used. If the error is smaller, only sampling will be applied.

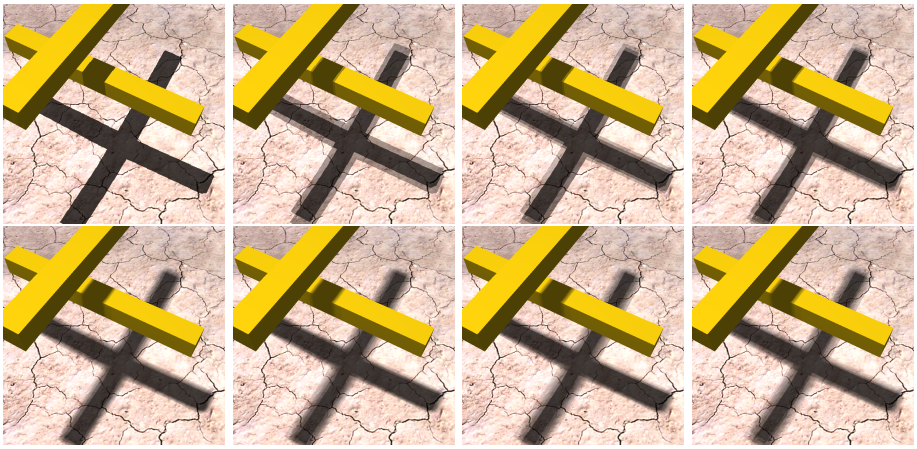

Fig. 3. Convergence for the first 4 frames when sampling the light source one sample per frame (*upper row*) can be greatly increased by using PCSS as its first sample (*lower row*). Here we have used $n = 4$ for the PCSS sample, so it is weighted like 4 normal samples.

Neighborhood Filter The neighborhood filter should remove noise and flickering artifacts in the resulting shadow by smoothing. We use a box filter and only include samples within a certain depth range of the current fragment, since those are likely to have a similar $\hat{\psi}_n(x, y)$. The main question is how to set the filter width to achieve the correct amount of smoothing. Since we want to render soft shadows, a good filter size is given by the penumbra width.

Although we don't know the exact penumbra width, we can estimate it efficiently. We base our estimation on the one used in Fernando [4] because it is fast and simple. It assumes blocker, receiver and light source are parallel and is given by

$$pw = \frac{near}{depth_{eye}} \frac{receiver - avg(blocker)}{avg(blocker)} light_{size} \tag{6}$$

where pw denotes the penumbra width (projected to screen space) we want to estimate, $receiver$ is the depth of the current fragment and $light_{size}$ is the size of the light source. $near$ is the near plane distance and $depth_{eye}$ the fragment depth for the projection.

The calculation of the average blocker depth $avg(blocker)$ is one of the costliest steps in this algorithm. It works by averaging the depths of the k nearest texels in the shadow map for each fragment each frame. In our approach, we can avoid doing this by using temporal coherence again: When a fragment first becomes visible we perform PCSS, including blocker depth estimation, anyway. We save this value $avg(blocker)$, generated from k depth samples, as a starting value, and in each successive frame we refine it using one additional depth

sample $depth_i$ from the new shadow map:

$$avg(blocker)_i = \frac{depth_i + avg(blocker)_{i-1}(i-1+k)}{i+k} \tag{7}$$

3.4 Accounting for Moving Objects

The original temporal reprojection paper by Scherzer et al. [12] does not account for moving objects and up to now we have only accounted for camera movement by reprojection. If we want to be able to display moving objects that cast and receive soft shadows, we have to investigate how these influence the evaluation of our algorithm. Moving objects will appear in the shadow map as potential casters of dynamic shadows and also in the scene as dynamic objects where shadows are cast upon. With moving objects disocclusions become very frequent and therefore temporal reprojection does not work as well.

Our approach is therefore to first identify moving objects and shadows that are cast by moving objects in order to handle these cases. First we change the generation of the shadow map by including for each texel the information whether it belongs to a moving object or not. Because the depth in the shadow map is always positive we can store this inside the depth by using the sign, therefore generating no additional memory cost. A negative sign means that this fragment is from a moving object (and therefore a potential dynamic shadow caster). We can now retrieve this information for each screen-space fragment when we do the shadow map test. If we have a negative depth we know that the shadow that falls on this object is cast by a moving object.

If we detect such a case we must assume that the data stored in the shadow buffer for the current fragment is probably invalid and we therefore handle it like a normal disocclusion and apply PCSS. This alone would lead to unsatisfactory results because we generate a different shadow map each frame on a different position on the light source. This would cause the PCSS shadow to jump around each frame. Therefore we additionally apply the neighborhood filter from Section 3.3 to smooth out any jumps and decrease discontinuities between the primarily PCSS based moving shadows with the sampled static shadow.

4 Implementation and Results

We implemented the algorithm in 3 passes: first, render shadow map, second, render into new shadow buffer (applying algorithm) and final color buffer (use shadow buffer from previous frame as input) and third copy final color buffer to framebuffer.

For our tests, we used an Intel Core 2 Duo E6600 CPU with an NVIDIA 280GTX graphics card and DirectX 10. All images were taken using shadow maps with a resolution of 1024^2. For selecting the sample position on the area light we used a Halton sequence. Other quasi-random sequences showed similar behavior. The screen space neighborhood filter uses a Poisson disk centered at the current fragment with a fixed sample size (16 samples).

The shadow maps were rendered using standard uniform shadow mapping, a 32bit floating point texture and linear depth. Hardware 2x2 PCF filtering was not used. We used the multiple render target functionality for the second pass to render into the shadow buffer and into an 8bit RGB buffer. The shadow buffer is a 4-channel 16bit floating point texture. It contains ρ_n, n, the linear depth of the fragment and the average blocker depth $avg(blocker)$. To have meaningful neighborhood texels at the frame buffer borders, we have chosen the resolution of the shadow buffer about 5% larger than the framebuffer, so if we assume a 800x600 framebuffer, the shadow buffer would be 840x630. Please note that 5% was sufficient for our movement speeds. If faster movements occur, a larger "overscan" should be choosen. Two instances of the shadow buffer are required (one for reading and one for writing), resulting in an additional memory requirement of $2 \times 840 \times 630 \times 4 \times 2 \approx 8MB$. We used an error threshold of $t = 1/50$.

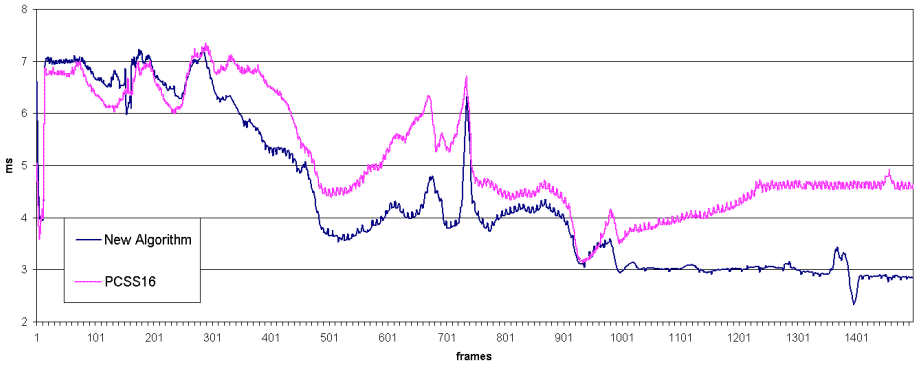

Fig. 4. A sample walkthrough in one of our test scenes with our new method and with PCSS using 16/16 samples for blocker/PCF lookup.

Our goal was to develop a soft shadow approach that should be faster than PCSS, but provide at the same time better quality, therefore we compared to a fast PCSS version using only 16 texture lookups for the blocker search and 16 texture lookups for the PCF kernel. Figure 4 shows a benchmark of a typical walkthrough of one of our test scenes, using a viewport of 1024x768 pixels. Timings are given in ms. Our algorithm tends to be slower if there are many disocclusions, because here it has to perform the blocker search that also PCSS has to perform. Our shader is more complex (more ifs) than the PCSS shader so we can be slower then PCSS in such circumstances. The used PCSS16 always performs 16+16 lookups, while our shader only has to do those 16+16 lookups for disocclusions. Our shader performs at least one shadow map lookup and one shadow buffer lookup every frame. 16 lookups are used for the neighborhood filter, which is the case when the standard error is higher then the threshold t for

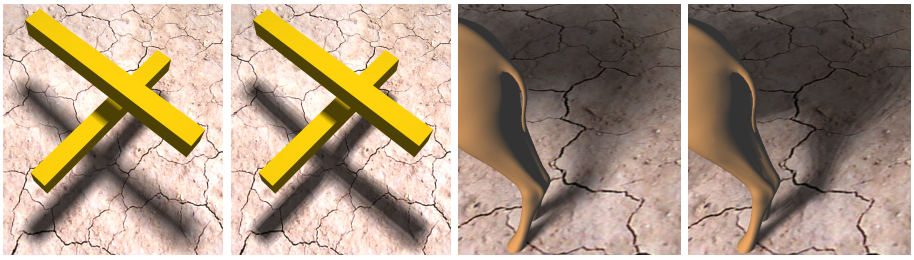

Fig. 5. *From left to right*: Overlapping occluders (our Method, PCSS 16/16) and bands in big penumbras (our Method, PCSS 16/16) are known problem cases for single sample approaches.

every fragment were the single sample soft shadow approach is active. Figure 5 shows typical problems of PCSS that are solved with our approach. A better comparison can be seen in the accompanying videos. We also did a comparison with a more elaborate PCSS with 32 lookups for the occluder search and 32 lookups for the PCF kernel, which was considerably slower than our algorithm.

4.1 Limitations

Although we presented a method to handle moving objects, there is still room for improvements. Especially when dynamic shadows overlap with static shadows with large penumbrae, some flickering artifacts remain (see the accompanying videos).

5 Conclusions and Future Work

We presented a very fast soft shadow approach based on shadow maps that uses temporal reprojection for achieving physical correctness. Where temporal reprojection is insufficient we use spatial filtering to allow for soft shadows on recently disoccluded fragments.

As a future direction of our research we would like to investigate multiple light sources because they lend themselves naturally to this approach: nowhere do we assume that the soft shadow data in the shadow buffer comes from the same light source, so we can extend the approach to multiple light sources simply by calculating a shadow map for each light source each frame. The values rho_n and n can then accumulate contributions from all the light sources.

Moving light sources could also be possible. We think that an approach that weights older light source samples less, together with age factors for shadow buffer fragments, could work. Moving objects would benefit from calculating each frame a second shadow map in the center of the light source. Maybe the two shadow maps could be calculated in a combined fashion. Our statistical approach is now based on uniform distributions of samples. Maybe non-uniform distributions could improve convergence.

Acknowledgements

This work was supported by the Austrian Science Fund (FWF), project number P21130-N13.

References

1. Hasenfratz, J.M., Lapierre, M., Holzschuch, N., Sillion, F.: A survey of real-time soft shadows algorithms. Computer Graphics Forum **22** (2003) 753–774
2. Schwarz, M., Stamminger, M.: Quality scalability of soft shadow mapping. In: GI '08: Proceedings of graphics interface 2008, Toronto, Ont., Canada, Canada, Canadian Information Processing Society (2008) 147–154
3. Heckbert, P.S., Herf, M.: Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, CS Dept., Carnegie Mellon U. (1997) CMU-CS-97-104, http://www.cs.cmu.edu/ ph.
4. Fernando, R.: Percentage-closer soft shadows. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches, New York, NY, USA, ACM (2005) 35
5. Annen, T., Mertens, T., Bekaert, P., Seidel, H.P., Kautz, J.: Convolution shadow maps. In Kautz, J., Pattanaik, S., eds.: Rendering Techniques 2007: Eurographics Symposium on Rendering. Volume 18 of Eurographics / ACM SIGGRAPH Symposium Proceedings., Grenoble, France, Eurographics (2007) 51–60
6. Annen, T., Dong, Z., Mertens, T., Bekaert, P., Seidel, H.P., Kautz, J.: Real-time, all-frequency shadows in dynamic scenes. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers, New York, NY, USA, ACM (2008) 1–8
7. Guennebaud, G., Barthe, L., Paulin, M.: Real-time soft shadow mapping by backprojection. In: Eurographics Symposium on Rendering (EGSR), Nicosia, Cyprus, Eurographics (2006) 227–234
8. Guennebaud, G., Barthe, L., Paulin, M.: High-Quality Adaptive Soft Shadow Mapping. Proceedings of Computer Graphics Forum, Eurographics 2007 **26** (2007) 525–534
9. Schwarz, M., Stamminger, M.: Microquad soft shadow mapping revisited. In: Eurographics 2008 Annex to the Conference Proceedings: Short Papers. (2008) 295–298
10. Agrawala, M., Ramamoorthi, R., Heirich, A., Moll, L.: Efficient image-based methods for rendering soft shadows. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (2000) 375–384
11. St-Amour, J.F., Paquette, E., Poulin, P.: Soft shadows from extended light sources with penumbra deep shadow maps. In: Graphics Interface 2005 Conference Proceedings. (2005) 105–112
12. Scherzer, D., Jeschke, S., Wimmer, M.: Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In Kautz, J., Pattanaik, S., eds.: Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering), Eurographics, Eurographics Association (2007) 45–50
13. Velázquez-Armendáriz, E., Lee, E., Bala, K., Walter, B.: Implementing the render cache and the edge-and-point image on graphics hardware. In: GI '06: Proceedings of Graphics Interface 2006, Toronto, Ont., Canada, Canada, Canadian Information Processing Society (2006) 211–217
14. Nehab, D., Sander, P.V., Lawrence, J., Tatarchuk, N., Isidoro, J.R.: Accelerating real-time shading with reverse reprojection caching. In: Graphics Hardware. (2007)