Contact Hardening Soft Shadows using Erosion

Andreas Klein
Munich University of Applied Sciences
Lothstrasse 64
80335 Munich, Germany
andreas.klein@hm.edu

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

Paul Obermeier
MBDA Deutschland GmbH
Hagenauer Forst 27
86529 Schrobenhausen, Germany
paul.obermeier@mbda-systems.de

ABSTRACT

In this paper, we present an image based method for computing contact hardening soft shadows by utilizing an erosion operator. Our method is based on shadow mapping and operates in screen space. By using object silhouettes in hard shadows, we estimate the penumbra size and scale an erosion operator to generate the penumbra areas. Furthermore, we present two solutions to generate the shadow factor for the penumbra areas. Our method works best for small penumbras and can be easily integrated into existing shadow mapping based applications.

Keywords

Shadow Mapping, Soft Shadows.

1 INTRODUCTION

Shadows are an important part for the human perception. They give clues about the spatial relationship and the form of objects. Real world shadows can be divided into an umbra and a penumbra. An umbra occurs when a light source is completely occluded and a penumbra when it is partially occluded.

In real-time rendering, a popular method to generate shadows is shadow mapping [Wil78a]. Shadow mapping assumes point light sources and thus, only hard shadows are produced. However, real world light sources are extended, and they generate penumbras, whose size often can be proportional to the light size and the receiver-blocker distance.

Current methods for generating contact hardening soft shadows are not suited for high shadow map resolutions [Lau07a], require a high amount of texture fetches [Fer05a] or the performance decreases with the number of shadow maps [Gum10a].

We present an algorithm to produce contact hardening soft shadows using an erosion operator. Our algorithm is an extension to shadow mapping and operates in screen space. Furthermore, it is suited for high shadow map resolutions as well as multiple shadow maps.

2 RELATED WORK

The rendering of soft shadows has been studied extensively over the last years. Therefore, we focus our review on publications closely related to our work. For an exhaustive survey on other methods see [Eis11a].

Percentage closer filtering (PCF) [Ree87a] is a popular method for generating soft shadows. The idea is to make multiple shadow comparisons within a user defined filter window. The shadowing factor is then built by averaging the result. To generate contact hardening soft shadows with PCF, Fernando [Fer05a] proposed percentage closer soft shadows (PCSS). He introduced a blocker search as a preprocessing step, where he sampled the shadow map to calculate an average blocker depth for each screen space pixel and approximated a penumbra width with a parallel planes approximation. Finally, he used the penumbra width to scale the PCF filter window.

Arvo et al. [Arv04a] estimated the penumbra regions by detecting the edges in hard shadows and propagating a visibility factor using a flood fill algorithm. Rong and Tan [Ron06a] accelerated this method using jump flood fill algorithms. Gumbau et al. [Gum10a] dilated a shadow map to replace the blocker search of PCSS. Furthermore, they replaced the PCF filtering with a separable Gaussian blur.

There are several approaches to calculate soft shadows in screen space. Robison and Shirley [Rob09a] used a screen space distance map to estimate a penumbra

width and blurred a hard shadow map with it. Hanjun and Huali [Han10a] developed an algorithm that propagates a shadow factor using erosion and dilation that is closely related to our work. However, we incorporate contact hardening soft shadows as well as penumbra anti-aliasing. Aguado and Montiel [Agu10a] presented an approach where a penumbra size is propagated using a mipmap flood fill and the penumbra is generated with a Gaussian filter in screen space. However, this approach produces light leaks which can be reduced by using multiple layers. MohammandBagher et al. [Moh10a] used a projected shadow map in screen space to estimate a penumbra size and to blur a hard shadow map

3 ALGORITHM OVERVIEW

Our algorithm computes the penumbra in screen space and is an extension to existing shadow mapping approaches. The algorithm proceeds as follows. First, we render hard shadows with a shadow mapping algorithm. Second, we detect edges in the hard shadows and store the blocker-receiver distance as well as the camera-receiver distance for the edge pixels. Now we can calculate the penumbra width since it is proportional to the blocker-receiver distance and indirect proportional to the camera-receiver distance. Next, we erode the edges with a filter kernel that is scaled according to the penumbra width and thus, estimating the inner and outer penumbra regions for contact hardening soft shadows. In order to generate the final penumbra, we propose two solutions. First, by filtering the shadow map in the penumbra regions with PCF and second, by directly calculating it during erosion.

Rendering Hard Shadows

The first step in the algorithm is to render hard shadows and auxiliary buffers. We assume that a shadow map has already been rendered. The scene is rendered from the observer and we perform a standard shadow comparison for each screen space pixel. We store a one in a screen space hard shadow buffer for each lit pixel and a zero for each shadowed pixel. Additionally, the depth difference between the blocker and receiver as well as the distance to the camera is stored. Furthermore, we store the diffuse color in a separate texture to calculate the final shading in the last pass.

Edge Detection

The second step is to estimate the penumbra regions by detecting the edges in the screen space hard shadow buffer. As the hard shadow buffer is a binary image, we can easily detect the edges with a 3 x 3 Laplacian filter, which needs five texture fetches:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

We calculate the penumbra width for each edge pixel by using the parallel planes approximation of Fernando [Fer05a]. Additionally, the kernel size is scaled based on the camera-receiver distance [Gum10a]:

$$\omega_{penumbra} = \frac{(d_{receiver} - d_{blocker})\omega_{light}}{d_{blocker} d_{observer}}$$

where ω_{light} is the light dimension. We store the penumbra width in the second texture channel.

Erosion

After the edges have been detected, they are eroded with a variable sized erosion filter[1]. We estimate the erosion by using a min-max mipmap [Isi06a, Dmi07a] (Figure 1). Recall that the edge texture stores a zero for each boundary pixel in the first channel and the penumbra size in the second channel. First we generate the min-max mipmap hierarchy for the edge texture by performing a min operation in the first channel and a max operation in the second texture channel. During erosion we calculate a maximum search radius for the given light dimension and the distance to the observer in order to find the closest edge, as the penumbra widths are only stored in the edge pixels. We choose a mipmap level based on the maximum search radius and query the min-max mipmap hierarchy. If the first channel contains no boundary pixel, we immediately terminate the erosion. Otherwise, we read the penumbra width from the second texture channel and choose again a mipmap level. Finally, we access the mipmap hierarchy in this mipmap level and test for boundary pixels. If a boundary pixel is found, we store the penumbra width in the result texture. Otherwise we discard the pixel.

Determine the Shadow Factor

In the final pass, we use PCF to determine the shadow factor for each pixel. We scale the PCF filter based on the penumbra width and combine the result with the hard shadow map rendered in the first pass.

4 ESTIMATE A SHADOW FACTOR WITH EROSION

A second solution is to estimate a shadow factor directly during erosion. In order to realize this idea, some changes in the algorithm are necessary.

As the screen space hard shadow buffer is rendered from the observer's viewpoint, objects may occlude shadowed areas and thus, the edge detector will produce edges which do not belong to penumbra regions. As Arvo et al. [Arv04a] pointed out, this issue can be solved by testing the shadow map for silhouettes on

[1] In terms of image processing this operation is an erosion, as the zeros in the edge texture are propagated.

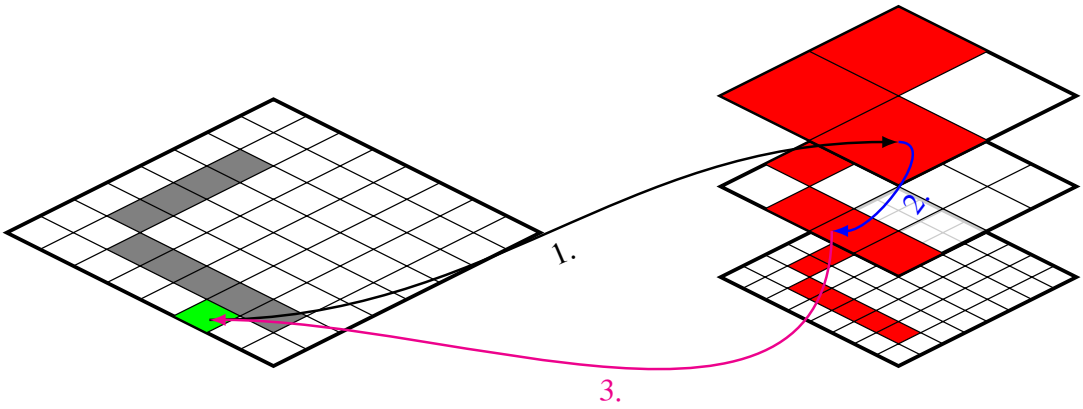

Figure 1: Erosion of the edges using a min-max mipmap. First, we choose mipmap level based on a maximum search radius and read the penumbra size from the max channel of the mipmap. Second, we calculate a mipmap level based on the penumbra size and read the edge value stored in min channel. Finally, we output the penumbra size if the edge value can be classified as a boundary pixel.

each detected edge pixel. We use the same 3 x 3 Laplacian filter and compare the result against a threshold. If the result is within the threshold, the edge pixel is valid and will be used in the next step. Otherwise, we discard it.

In order to compute the shadow factor, we implemented the variable sized erosion in a gathering approach. First, we determine a maximum kernel size and search for edge pixels within this area. If an edge pixel is found, we calculate its penumbra width and check, whether the current pixel is within its range. We continue until we found the edge pixel with the smallest distance to the current pixel. The shadow factor of the outer penumbra can then be directly calculated:

$$s_{outer} = \frac{\omega_{penumbra} - d_{min}}{2\omega_{penumbra}}$$

where d_{min} is the minimum distance to the edge and $\omega_{penumbra}$ the penumbra size. The inner penumbra is simply calculated with $s_{inner} = 1 - s_{outer}$.

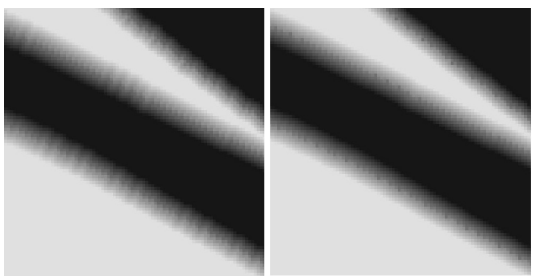

Figure 2: Left: Artifacts due to aliasing in hard shadows. Right: Result after the distance correction. Note that there are still some incorrect pixels at the transition from the outer to the inner penumbra.

Due to aliasing in the screen space hard shadow buffer, this method replicates the aliasing in the penumbra (Figure 2).

To increase the visual quality, we search for a best fit straight line by a least square method in a discrete environment around each edge pixel prior to the erosion and store the line parameters in an auxiliary texture. During erosion, we read the line parameters from the texture and calculate the vector v_{line} from the edge pixel's center to the line. Finally, we add v_{line} to the vector from the erosion point to the edge, calculate the distance and use it during erosion (Figure 3).

This compensates parts of the aliasing in the screen space hard shadow buffer and increases the visual quality (Figure 2). However, there are still some incorrect pixels at the transition from the outer to the inner penumbra. We will try to solve this issue in future work.

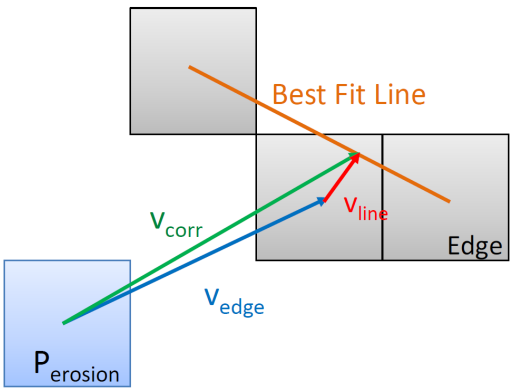

Figure 3: In order to improve the visual quality, we search for a best fit line and offset the vector to the edge with the vector from the edge to the line. Finally, we use the distance of the resulting vector during erosion.

5 RESULTS

We compared our algorithm against a PCSS implementation and a reference solution. This PCSS implementation uses a Poisson disk for the blocker search and PCF filtering. Both methods use 32 samples for the final PCF. The reference solution is realized by approximating the area light source with 512 point light sources. The screen resolution was 1920 x 1080 pixels and the shadow map size was 2048 x 2048. Figure 4 and 5 shows the resulting images and Table 1 the performance results. Table 2 shows the duration of the algorithm steps in the buddha dataset. The performance results were obtained on an Intel Xeon E5620 CPU with 2.4 GHz, 8 GB RAM and a NVIDIA GeForce GTX 680 graphics card with 2048 MB memory.

	Cactus (188K tris)	Hairball (2.88M tris)	Buddha (1.08M tris)
Erosion	1.62	12.56	5.77
PCSS 8	1.58	11.51	4.68
PCSS 32	1.60	11.89	5.11
PCSS 64	1.66	12.44	5.69

Table 1: Performance results in comparison with PCSS using 8, 32 and 64 blocker search samples. The screen resolution is 1080p.

Step	Time [ms]
Shadow Map	2.20
Hard Shadows	2.52
Edge Detection	0.31
Erosion	0.30
Shading	0.44

Table 2: Duration of the algorithm steps using the buddha dataset.

6 DISCUSSION

The bottleneck of PCSS [Fer05a] is the blocker search that is performed for each screen space pixel. Our motivation is to replace the blocker search per pixel by operating only on silhouettes of hard shadows. However, the rendering chain of our method is more complex. Thus, a speedup is only achieved when the blocker search is performed with 64 samples per pixel and the penumbra area is small. In contrast to PCSS, our method does not produce artifacts resulting from a small number of blocker search samples when rendering fine structured geometry, such as in Figure 4.

One possible issue in the method of [Gum10a] is that the algorithm operates on shadow maps. In contrast our algorithm operates on a screen space hard shadow buffer, which makes it attractive for applications with multiple shadow maps.

Compared to [Han10a] we incorporated variable sized penumbras and increased the visual quality by calculating the distance to a best fit straight line. Furthermore, we implemented a second solution for generating the shadow factor with a PCF filter, which results in a superior image quality.

Nevertheless, this technique has limitations. As our method is based on PCSS, it has the same limitations, such as overestimating the penumbra size. Furthermore, the erosion size is bounded and thus, we may miss relevant occluding information. This could result in visible artifacts. Due to mipmap erosion and the scaling of the penumbra width based on the distance to the camera, our method introduces aliasing when the camera is moved. Another limitation is that the visual quality is strongly dependent on the quality of the hard shadows. Consequently, aliasing reduction algorithms such as cascaded shadow mapping (CSM) [Eng06a] and light space perspective shadow maps (LiSPSM) [Wim04a] should be used.

7 CONCLUSIONS AND FUTURE WORK

We proposed a method for generating contact hardening soft shadows in screen space. As with all image based methods, this technique works best for small penumbras and can be used to extend shadow mapping based applications. Furthermore, we presented two solutions to generate a shadow factor for the penumbra. While the mipmap erosion is fast and produces results comparable to PCSS, the calculation of the shadow factor during erosion still produces some artifacts.

For future work, we wish to explore the possibility to replace the least square line fitting with a low pass filter and try to reduce the remaining artifacts.

8 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their useful comments. The work of A. Klein is funded by MBDA Deutschland GmbH.

9 REFERENCES

[Agu10a] Aguado, A. and Montiel, E. MipMapped Screen Space Soft Shadows. In GPU Pro 2. 2010

[Arv04a] Arvo J., Hirvikorpi M., Tyystjarvi J. Approximate Soft Shadows with an Image-Space Flood-Fill Algorithm. Computer Graphics Forum 23, 271-279, 2004.

[Dmi07a] Dmitriev K., Uralsky Y. Soft shadows using hierarchical min-max shadowmap. GDC 2007, 2007.

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

[Eng06a] Engel W. Cascaded shadow maps. In Shader X5, Engel W., (Ed.). Chares River Media, 197-206, 2006.

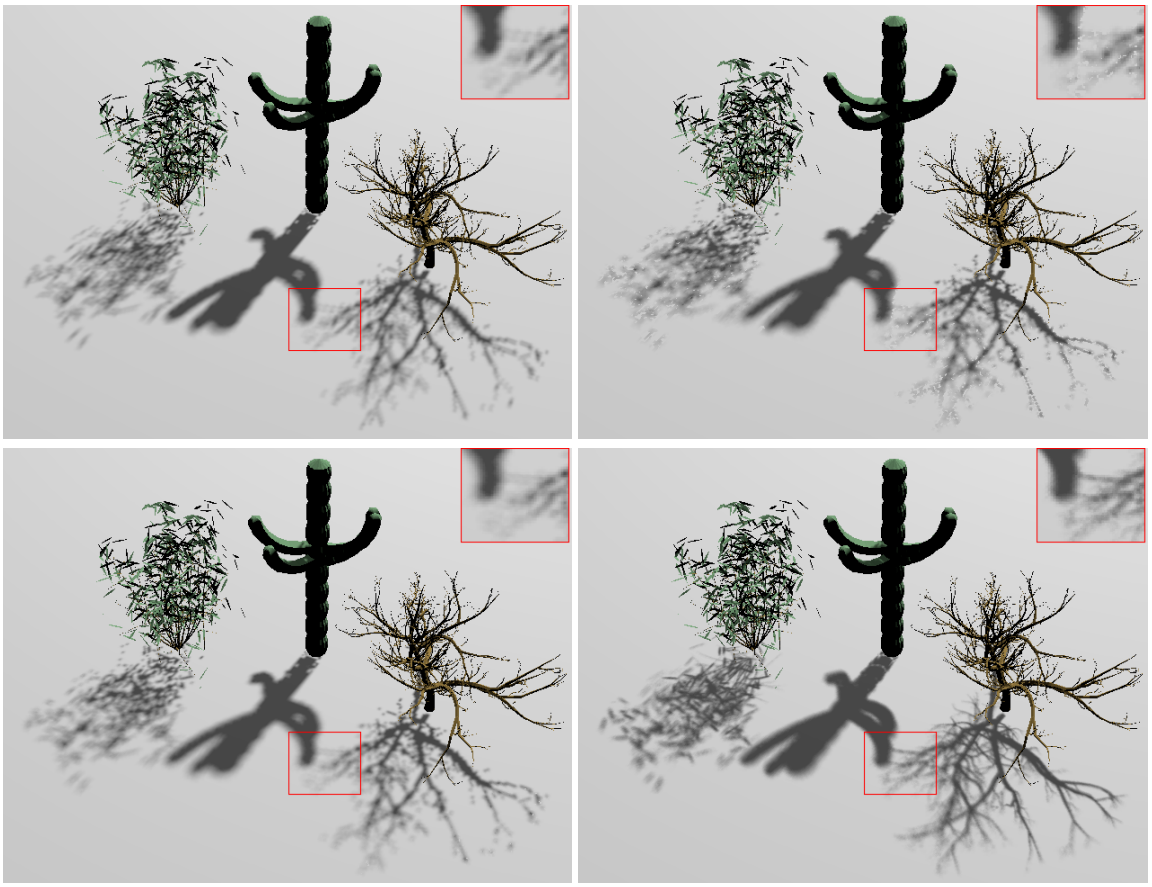

Figure 4: Visual results in the cactus dataset. Top Left: Our algorithm. Top Right: PCSS with 8 blocker search samples. Notice the artifacts resulting from missed blockers due to the small number of blocker samples. Bottom Left: PCSS with 64 blocker search samples. Bottom Right: reference solution.

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

[Gum10a] Gumbau J., Chover M., Sbert M. Screen space soft shadows. In GPU Pro - Advanced Rendering Techniques, Engel W., (Ed.). A.K. Peters, 2010.

[Han10a] Hanjun J., Huali S. Rendering fake soft shadows based on the erosion and dilation. 2nd International Conference on Computer Engineering and Technology, 234-236, 2010.

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

[Lau07a] Lauritzen A. Summed-area variance shadow maps. In GPU Gems 3, Nguyen H., (Ed.). Addison-Wesley, 157-182, 2007

[Moh10a] MohammadBagher M., Kautz J., Holzschuch N. and Soler, C. Screen-space percentage-closer soft shadows. ACM SIGGRAPH 2010 Posters, 2010.

[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.

[Rob09a] Robison A. and Shirley P. Image space gathering. In Conf.proc. High Performance Graphics 2009, 91-98, 2009.

[Ron06a] Rong G., Tan T.-S. Utilizing jump flooding in image-based soft shadows. In Conf.proc. VRST '06, ACM, 173-180, 2006.

[Wil78a] Williams L. Casting curved shadows on curved surfaces. In Conf.proc. SIGGRAPH '78, ACM, 270-274, 1978.

[Wim04a] Wimmer M., Scherzer D., Purgathofer W. Light space perspective shadow maps. In Conf.Proc. Eurographics Symposium on Rendering, 2004.

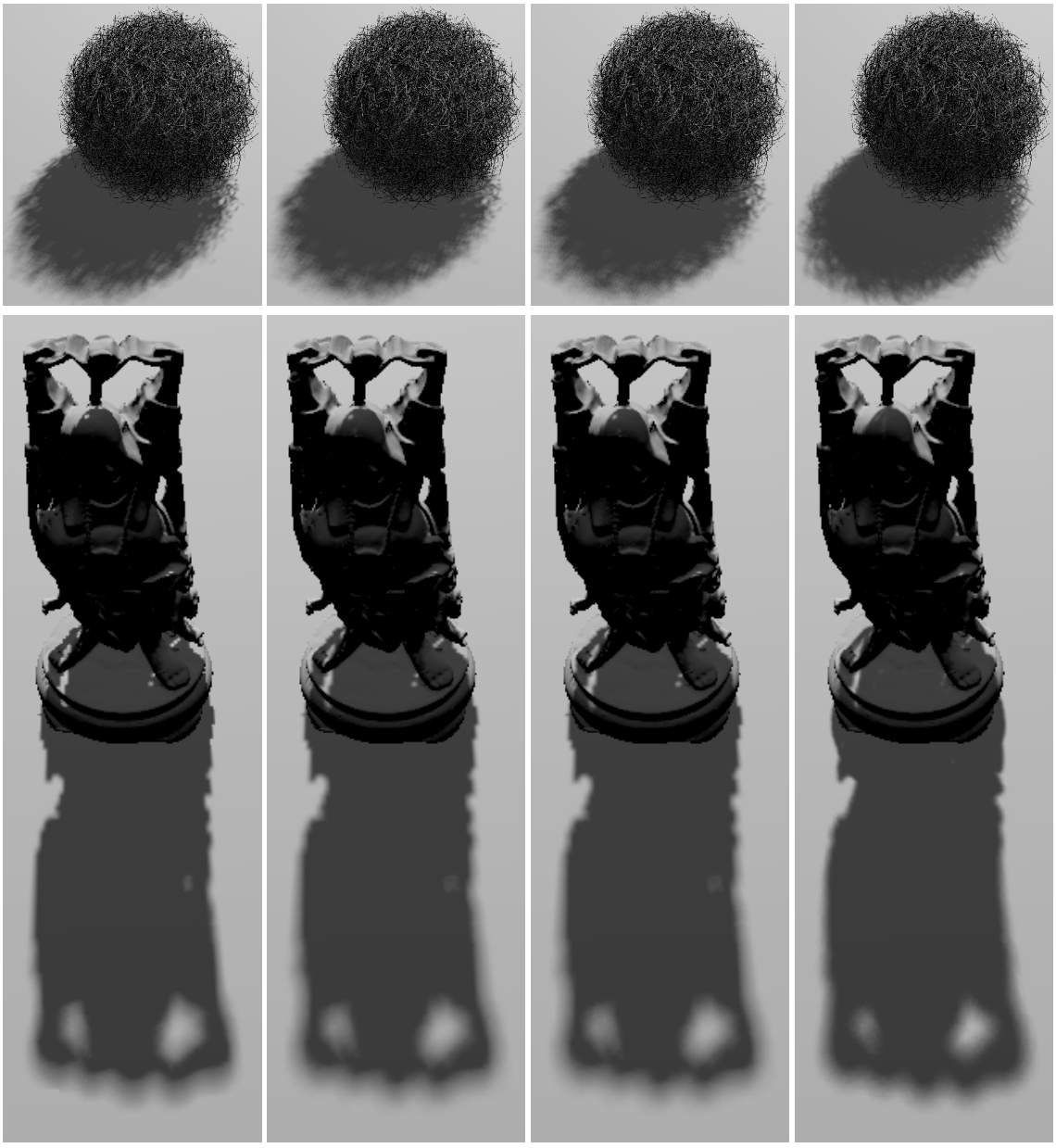

Figure 5: Resulting shadows from the hairball and buddha datasets. Note that the shadow softness increases with the blocker-receiver distance. From left to right: Our method, PCSS with 32 blocker search samples, PCSS with 64 blocker search samples and reference solution.