Vol. 13, No. 1: 19–29

Robust Soft Shadow Mapping with Backprojection and Depth Peeling

Louis Bavoil, Steven P. Callahan, and Claudio T. Silva
Scientific Computing and Imaging Institute, University of Utah

Abstract. Soft shadow mapping is an attractive solution to the problem of real-time soft shadows because it works with any kind of rasterizable geometry (in particular alpha-transparent textures and hair), it does not require any precomputation, and it is simple to implement on the GPU. However, state-of-the-art approaches have several limitations that prevent them from being practical for all scenes. First, parameter tuning is required to avoid surface acne. Second, gaps between shadow map pixels are either ignored, which results in light bleeding, or handled using gap filling, which results in overshadowing.

We present a more robust soft shadow mapping technique, based on a recent backprojection algorithm, that uses depth peeling to address the problems of surface acne and light bleeding. Our algorithm uses a multi-layer shadow map to reduce light bleeding, and midpoint shadow maps to handle self-shadowing more robustly. It provides high-quality soft shadowing for complex scenes, while still maintaining interactive rendering rates. Source code is available online.

1. Introduction

Shadow mapping [Williams 78] is a real-time algorithm for rendering hard shadows from point lights that maps very well to GPUs. In this classic algorithm, a preliminary pass renders the scene from the viewpoint of the light into a depth buffer (dubbed a *shadow map*). Then, image fragments are transformed to the image space of the light, and their depths are compared to the

1086-7651/06 $0.50 per page

shadow map depths to determine if the fragment is occluded from the light. The advantages of shadow mapping are its speed and simplicity.

A physically-based soft shadow mapping algorithm with self-shadowing was proposed recently by Guennebaud *et al.* [Guennebaud et al. 06], which computes a percentage of light seen from every pixel in the image. This is done by backprojecting the occluders in the shadow map onto the light plane and determining the magnitude of the occlusion. The algorithm addresses many of the issues of previous approaches, but still overshadows, mainly because it extends the occluding samples to prevent light bleeding.

Our algorithm extends this backprojection approach to reduce artifacts. We represent the visibility of an area light using multiple depth layers, similar to [Keating and Max 99] and [Agrawala et al. 00]. We extract our layers using depth peeling [Everitt 02], which can be performed efficiently on graphics hardware. This allows us to remove most light bleeding artifacts without extending the shadow map samples. In addition, by using the midpoints between layers, our algorithm reduces the problem of surface acne.

2. Soft Shadow Mapping with Backprojection

Soft shadows rendered using percentage closer filtering (Percentage Closer Soft Shadows – PCSS) [Fernando 05] usually look plausible and are efficient enough to be used in video games on current graphics hardware. However, PCSS is similar to tracing rays from a point light to the scene rather than from a point to be shaded to the light. Unlike PCSS, the backprojection algorithm [Guennebaud et al. 06] approximates the result of ray tracing from the point to be shaded. To avoid doing actual ray tracing, it considers the samples from a single shadow map as micropatches in world space and computes the amount of occlusion of the micropatches seen from the point to be shaded. Because it is more physically based, backprojection produces soft shadows of higher quality than PCSS. Besides, self-shadowing is easier to handle with backprojection since false occluders tend to backproject outside the light area. We provide a brief summary of the backprojection algorithm here.

Assuming a square area light and square shadow map pixels, the algorithm proceeds as traditional shadow mapping by rendering the scene from the viewpoint of the light to create a shadow map of depth values. Then, the scene is rendered from the viewpoint of the eye and each pixel is shaded based on some shadow map tests. Each shadow map sample is first tested for approximate occlusion using a traditional depth test based on its biased depth value. This test is necessary to discard samples behind the point to be shaded relative to the light. If the sample passes the test, it is backprojected onto the light based on its actual depth value. This backprojected sample clamped to the light area results in a percentage of light seen by the point to be shaded.

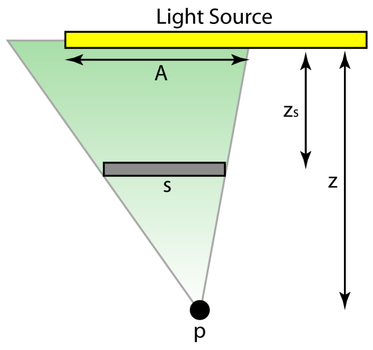

Figure 1. Backprojection works by computing the projected area A on the light source of a shadow map sample s from a shaded point p.

The backprojection algorithm is based on accumulating the percentage of the light source that each shadow-map sample is occluding (see Figure 1). A shadow-map sample is a depth value z_s in eye space, representing the distance between the shadow map plane and the sample. Based on this distance, the width of the sample is proportional to z_s. Then, given the depth z of the point to be shaded p and the difference between z and z_s, the backprojected size of the sample is a scale operation. The backprojected sizes of the sample in the x and y directions are computed, and clamped to the light size.

Concretely, the normalized coordinates B of the backprojection bounds of a sample of depth z_s seen from a point to be shaded p of depth z is:

$$B = \begin{bmatrix} b_{left} \\ b_{right} \\ b_{bottom} \\ b_{top} \end{bmatrix} = \begin{matrix} (du - 0.5) \\ (du + 0.5) \\ (dv - 0.5) \\ (dv + 0.5) \end{matrix} \ (\frac{w}{n\ r}\ z_s)(\frac{z}{z - z_s})\frac{1}{l} \qquad (1)$$

where (u_s, v_s) and (u, v) are respectively the shadow map coordinates of the sample and the point to be shaded (in pixels), $du = u_s - u$, $dv = v_s - v$, $w_s = \frac{w}{n\ r}\ z_s$ is the size of the sample (in world-space), w and n are respectively the width and depth of the light plane (in world-space), r is the width of the shadow map (in pixels), z is the depth of the point to be shaded (in world-space), and l is the width of the light (in world-space). The intersection of the backprojected sample with the light is performed by clamping B by [-0.5,0.5]. The shadow contribution of a sample is the area of the clamped backprojection area $A = (b_{right} - b_{left})(b_{top} - b_{bottom})$. Note that the depths z and z_s must be view-space depths (linear), not post-projection depths from a traditional shadow map (non linear).

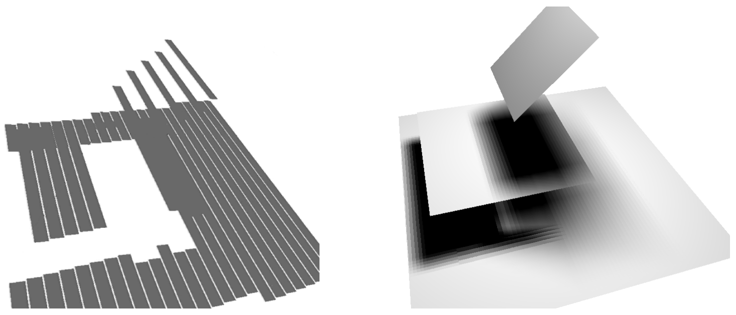
(a) (b)

Figure 2. Light bleeding is caused by gaps between shadow map pixels as seen from an area light. Here we show the errors that a single shadow map (a) produces in a simple scene with soft shadows (b).

This algorithm assumes that every sample of the shadow map with a depth less than z_s is a potential occluder. To reduce this set of samples, a conservative kernel is the intersection of the near plane ($z = n$) of the shadow map with the shadow frustum defined by the point to be shaded and the square light. The center of the search region is given by projecting the point to be shaded in light space, as in regular shadow mapping. The width (in pixels) of the kernel is given by $w_0 = r\ l\ \frac{n}{w}\ (\frac{1}{n} - \frac{1}{z})$. The search region can be reduced by fetching a local minimum depth from a hierarchical shadow map [Guennebaud et al. 06, Guennebaud et al. 07]. Another optimization is to perform adaptive screen-space shadow sampling [Guennebaud et al. 07] to avoid oversampling large low-frequency penumbra. Such sampling optimizations are orthogonal to our approach of using a multi-layer shadow map.

Though backprojection is more accurate than PCSS, there are still a few artifacts that need to be addressed. First, the algorithm suffers from surface acne artifacts that are currently reduced using a hand-tuned depth bias. This problem is common to most algorithms based on image-based representations of a scene. Second, the original backprojection algorithm suffers from overshadowing.

3. Soft Shadow Mapping Artifacts

Due to the discrete nature of shadow maps, small overlaps and gaps may occur between shadow map pixels seen from a given point to be shaded. This problem affects all the soft shadow mapping algorithms that approximate the original geometry by a set of shadow-map samples unprojected into world space. Gaps result in light bleeding, whereas overlaps result in overshadowing.

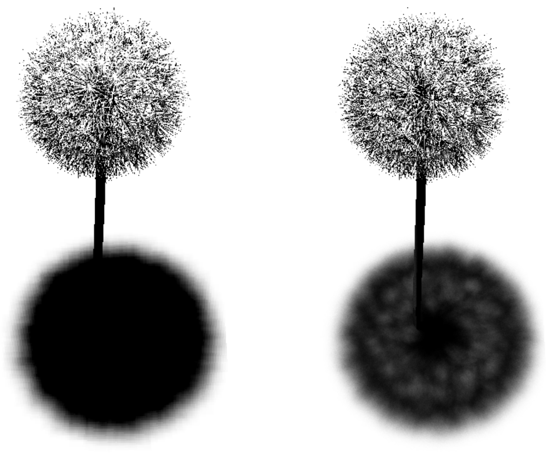

Figure 3. (a) The dandelion scene shows that gap filling [Guennebaud et al. 06] can result in shadows that are too dark. (b) A ray-traced image of the dandelion scene shows the correct shadowing.

In addition, surface acne appears because the points seen from the eye do not correspond exactly to the points seen from the light. We do not address the problem of removing occlusion overlaps. However, we do address the issues of light bleeding and surface acne.

3.1. Light Bleeding

Light bleeding occurs when there is a gap between shadow map pixels seen from the point to be shaded. Thus, surfaces that are in the penumbra of the first object, as seen from the light, will not get shadows from other objects that may fully occlude the surface. This is a significant problem for any scene with multiple overlapping shadows and is magnified with higher depth ranges. Figure 2 shows an example of this case where objects closest to the light interfere with the shadows cast from objects closer to the shadow. One approach to resolving the problem of light bleeding is to overestimate the shadow by extending the shadow map pixels, as in the original backprojection algorithm. This *gap filling* approach does not work properly for shadows of thin objects such as hair (see Figure 3).

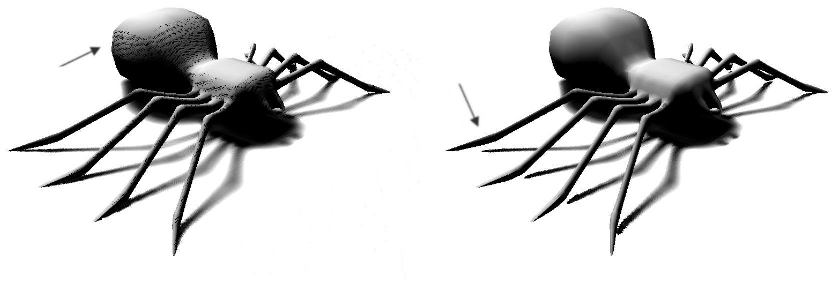

Figure 4. Self-shadowing issues, using a uniform depth test. (a) A depth bias too small results in surface acne. (b) A depth bias too large results in the incorrect placement of shadows.

3.2. Surface Acne

Self-shadowing is traditionally handled by computing a depth for the point to be shaded (z) and for the shadow map sample (z_s). A point is considered in shadow if and only if $z_s < z$. However, due the discrete nature of the shadow map, *surface acne* (false self-shadowing) appears in the final image because shadow-map samples behind the visible surfaces are peaking through. To counter this, a depth bias is usually applied to move the occluders underneath the surface farther away from the point to be shaded. This bias is usually based on the slope of the shadow map (`glPolygonOffset`). The shadow test then becomes $z_s + bias(z_s) < z$. A depth bias that is too small results in aliasing, while a depth bias that is too large results in incorrectly placed shadows. Unfortunately, the depth bias often requires manual tuning to achieve good image quality. Figure 4 shows the errors that occur when using a uniform depth bias to correct surface acne.

4. Removing the Artifacts

To minimize the problems of light bleeding, overshadowing, and surface acne, we introduce two enhancements to the original backprojection algorithm—multi-layer shadow maps for reducing gaps causing light bleeding, and multi-layer midpoint shadow maps for reducing surface acne. These modifications are described in detail in this section.

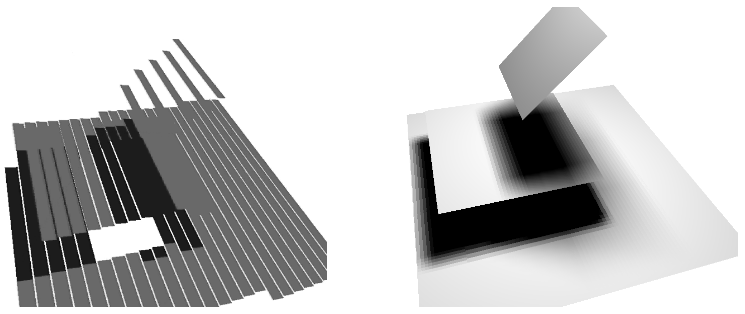

Figure 5. Multiple layers of shadow maps reduces light bleeding artifacts by filling gaps seen from the area light. Here we show a shadow map using two layers distinguished by different shades of gray (a) and the resulting soft shadows (b).

4.1. Multi-Layer Shadow Maps

To create a multi-layer shadow map, for every frame we render the entire scene multiple times from the light using depth peeling [Everitt 02] with a fixed number of layers (see Figure 5). The idea behind depth peeling is to capture one layer of depth for each rendering pass, starting with the nearest fragments at each pixel, the second nearest, the third nearest, and so on. Everitt [Everitt 02] introduced a simple approach to depth peeling on commodity hardware. The first pass renders the scene normally and results in depths for the nearest surface. In subsequent passes, the depth buffer computed in the previous pass is used to peel away depths less than or equal to those already captured in previous passes. Thus, each pass i generates a depth buffer for the ith nearest surface. For shadow mapping, this depth peeling results in multiple layers, each containing the occluders' world-space distances to the light plane.

After the shadow map passes, the shadow intensity of each pixel in the image is computed as before, by projecting the pixel onto the shadow map and testing shadow map samples for occlusion using a conservative search region. Each sample coordinate in the shadow map corresponds to a beam with the origin at the center of the light, going through the shadow map sample (see Figure 5). By sampling the depth values of the first k samples for a given shadow map sample coordinate, we effectively take the first k hits along the corresponding light beam. We efficiently reduce light bleeding by computing the occlusion of every sample along a light beam and using the furthest occluding sample from the light. In practice, we found that by using only three layers we can remove most light bleeding artifacts, as was noted in [Agrawala et al. 00].

4.2. Midpoint Shadow Maps

As described above, surface acne results from an incorrect depth bias. Several approaches have been developed to adaptively compute the depth bias. One common method is to use the `glPolygonOffset` function in OpenGL. This function offsets the depth of the rasterized fragments by a constant epsilon and a coefficient proportional to the maximum slope of the depth values around the pixel. This method requires tuning the slope parameter to remove surface acne, which can be difficult for complex scenes.

The midpoint shadow map algorithm [Woo 92] uses the average depth of the first two surfaces encountered from the viewpoint of the light for the self-shadowing test. This algorithm works for both closed and non-closed objects. Using a midpoint shadow map removes most of the issues with self-shadowing. However, it too has minor issues. First, light bleeding may appear when the midpoint and the original point are too far away. This can be fixed using a maximum depth bias [Weiskopf and Ertl 03]. Second, surface acne may remain at corners where the midpoint and the original surface meet. We found that these artifacts are minor in relation to the artifacts that are inherent to a limited-resolution shadow map, thus we use an extension of midpoint shadow mapping to handle self-shadowing.

For a given sample coordinate in the shadow map, we sample a fixed number of depths in our multi-layer shadow map. We then compute biased depth values corresponding to the midpoints between the layers. We use the biased depth values for handling self-shadowing and the original depths for computing the actual shadow contributions. Then, the samples are processed starting from the deepest shadow map layer and the first non-zero shadow contribution is selected and added to the shadow.

5. Examples and Discussion

Apart from the shadow map resolution, the main quality parameter of our algorithm is a maximum number of samples per pixel *max nspp*. We compute the width of the square search region as described in the backprojection algorithm. If the number of samples in the search region is less than *max nspp*, we process all the sample coordinates in the search region. Otherwise, we use an adaptive step so that the search area is sampled uniformly with *max nspp* samples. We compute $step = (region\ nspp)/(max\ nspp)$, and we scale the world-space size of the samples w_s by *step* when backprojecting the samples. This uniform sampling method can be combined with light-space optimizations to reduce the size of the search region conservatively [Guennebaud et al. 07, Schwarz and Stamminger 07]. Note that it may introduce minor artifacts because the sampling pattern shifts with the point to be shaded.

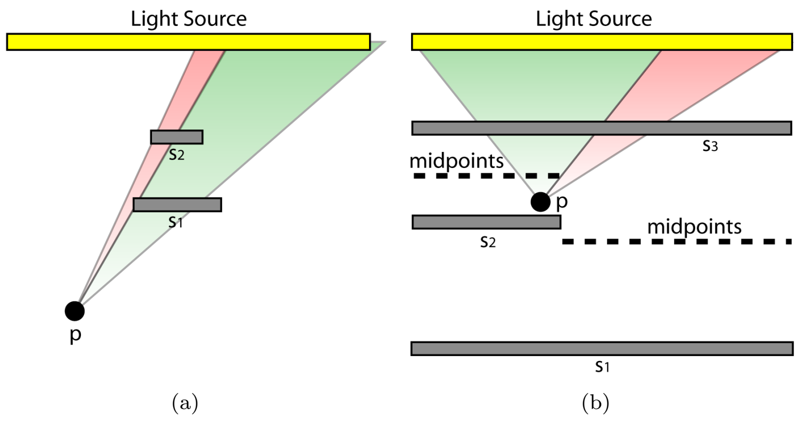

Figure 6. Though multiple layer shadow mapping reduces light bleeding substantially, some rare failure cases remain. (a) In this scene containing two micropatches, our algorithm would only count the contribution of the nearest micropatch (s_1) to the shaded point. However, the partial contribution of the second patch (s_2) should be added. (b) Midpoints can introduce light bleeding. This case can be handled by limiting the difference between the original depth and the midpoint depth.

Since the main bottleneck in the algorithm is computing per-pixel shadows, we use a deferred shadowing pipeline which computes the shadows only for the visible fragments. We use a predicate in the shadow shader to avoid computing shadows of background pixels or pixels with black diffuse colors. Thus, the speed of the algorithm depends on the number of non-background and non-black pixels, the resolution of the shadow map, the maximum number of samples per pixel, and the number of depth layers. For all our scenes, we used three shadow map layers and achieved images that look similar to ray tracing (see Figure 7). Our technique is useful for reducing light bleeding when using backprojection and gap filling renders unacceptable shadows, such as for hair-like objects.

Though the algorithm we have introduced removes most of the light bleeding and surface acne artifacts that we describe in Section 3, some issues may still arise in rare cases. First, our solution to light bleeding is not guaranteed to remove all light bleeding since it uses a fixed number of layers. For instance, for a simple scene such as an oblique quad, light may still bleed though. Also, our technique may incorrectly ignore the contribution of some micropatches, leaving some light bleeding (see Figure 6). In practice, we have not noticed this effect, as it is probably compensated by overshadowing from overlapping backprojections. Second, although the algorithm uses midpoint shadow mapping and removes much of the surface acne without parameter tuning, there are still rare cases when the acne appears, such as when the resolution of the shadow map is too low or the layers are very close together. One solution to this problem would be to take the second samples from the point to be shaded instead of the midpoints, but this tends to result in light bleeding, especially for non-closed objects. Third, because the backprojections of shadow map samples may overlap, the algorithm tends to slightly overshadow (see Figure 7(k)). Though it is a substantial improvement over the original backprojection algorithm, as the resolution of the shadow map or the number of samples increases, this problem becomes more pronounced.

The techniques we have described in this work significantly improve overall quality of real-time soft shadow mapping. Furthermore, in our experiments, we found that the algorithm worked robustly without the usual parameter tuning that is required with state-of-the-art techniques. Because the results approach those of ray tracing in quality, the algorithm could be efficiently used as a preview tool in a production setting for scenes with complex geometry and soft shadows.

Acknowledgments. We thank Sony Computer Entertainment America R&D for their support, and Gael Guennebaud for helpful conversations. Louis Bavoil, Steven P. Callahan, and Cláudio T. Silva are funded by the National Science Foundation (grants CCF-0401498, EIA-0323604, OISE-0405402, IIS-0513692, and CCF-0528201), the Department of Energy, an IBM Faculty Award, Sandia National Laboratories, Lawrence Livermore National Laboratory, the Army Research Office, and a University of Utah Seed Grant.

References

[Agrawala et al. 00] Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laurent Moll. "Efficient Image-Based Methods for Rendering Soft Shadows." In *Proceedings of ACM SIGGRAPH 2000*, *Computer Graphics Proceedings, Annual Conference Series*, pp. 375–384, 2000.

[Everitt 02] Cass Everitt. "Interactive Order-Independent Transparency." Technical report, NVIDIA Corporation, 2002.

[Fernando 05] Randima Fernando. "Percentage-closer soft shadows." In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, p. 35, 2005.

[Guennebaud et al. 06] Gaël Guennebaud, Loïc Barthe, and Mathias Paulin. "Real-time soft shadow mapping by backprojection." In *Eurographics Symposium on Rendering*, pp. 227–234, 2006.

[Guennebaud et al. 07] Gaël Guennebaud, Loïc Barthe, and Mathias Paulin. "High-Quality Adaptive Soft Shadow Mapping." In *Eurographics*, 2007.

[Keating and Max 99] Brett Keating and Nelson Max. "Shadow Penumbras for Complex Objects by Depth-Dependent Filtering of Multi-Layer Depth Images." In *Eurographics Workshop on Rendering*, pp. 205–220, 1999.

[Schwarz and Stamminger 07] Michael Schwarz and Marc Stamminger. "Bitmask Soft Shadows." *Computer Graphics Forum* 26:3 (2007), 515–524.

[Weiskopf and Ertl 03] Daniel Weiskopf and Thomas Ertl. "Shadow Mapping Based on Dual Depth Layers." In *Proceedings of Eurographics*, pp. 53–60, 2003.

[Williams 78] Lance Williams. "Casting Curved Shadows on Curved Surfaces." In *Computer Graphics (Proceedings of SIGGRAPH 78)*, 12, 12, pp. 270–274, 1978.

[Woo 92] Andrew Woo. "The shadow depth map revisited." In *Graphics Gems III*, pp. 338–342. San Diego, CA, USA: Academic Press Professional, Inc., 1992.

Web Information:

Louis Bavoil, Scientific Computing and Imaging Institute, University of Utah,
50 S. Central Campus Drive, Room 3490, Salt Lake City, UT 84112, USA
bavoil@sci.utah.edu

Steven P. Callahan, Scientific Computing and Imaging Institute, University of Utah,
50 S. Central Campus Drive, Room 3490, Salt Lake City, UT 84112, USA
stevec@sci.utah.edu

Cláudio T. Silva, Scientific Computing and Imaging Institute, University of Utah,
50 S. Central Campus Drive, Room 3490, Salt Lake City, UT 84112, USA
csilva@sci.utah.edu

Received August 8, 2006; accepted in revised form January 18, 2008.

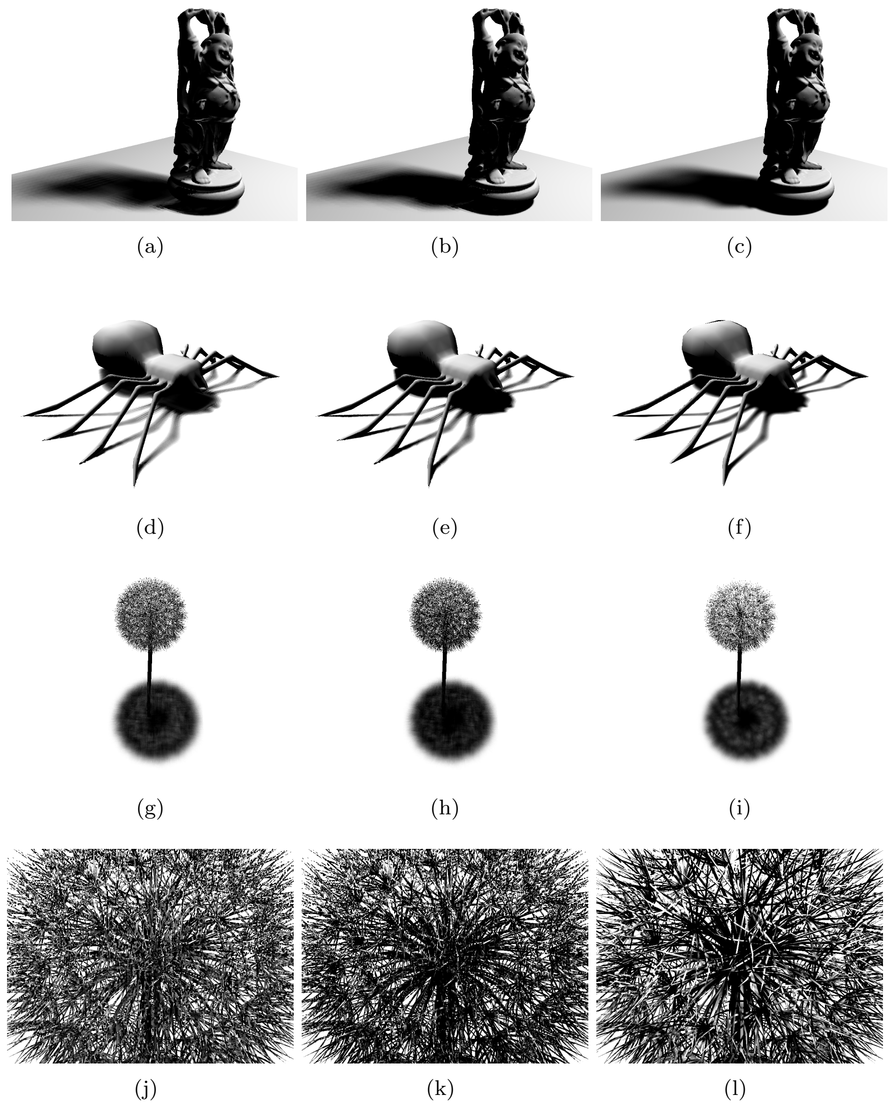

Figure 7. Adding shadow map layers. Our algorithm with uniform sampling (*max nspp* = 121) compared to ray tracing with 1,000 samples per pixel. Image Resolution: 800×600. Shadow Map Resolution: 1024^2. GPU: GeForce 7800 GTX, CPU: AMD Opteron Processor 275 @ 2.2 GHz. **Left:** Our algorithm with 1 layer. **Middle:** Our algorithm with 3 layers. **Right:** Ray tracing. **First row:** Happy Buddha (293,264 triangles). (a) 10.8 fps (b) 7.6 fps (c) 20 min. **Second row:** Spider (3,316 triangles). (d) 13.4 fps (e) 8.7 fps (f) 7 min. **Third row:** Dandelion (35,107 triangles). (j) 18.5 fps (k) 13.0 fps (l) 16 min. **Forth row:** Close-up of the dandelion. (j) 11.5 fps (k) 7.6 fps (l) 50 min.