# Soft Shadow Maps for Linear Lights

Wolfgang Heidrich Stefan Brabec Hans-Peter Seidel
Max-Planck-Institute for Computer Science
Im Stadtwald
66123 Saarbrücken
Germany
{heidrich,brabec,hpseidel}@mpi-sb.mpg.de

**Abstract.** Soft shadows and penumbra regions generated by extended light sources such as linear and area lights are visual effects that significantly contribute to the realism of a scene. In interactive applications, shadow computations are mostly performed by either the shadow volume or the shadow map algorithm. Variants of these methods for soft shadows exist, but they require a significant number of samples on the light source, thereby dramatically increasing rendering times.
In this paper we present a modification to the shadow map algorithm that allows us to render soft shadows for linear light sources of a high visual fidelity with a very small number of light source samples. This algorithm is well suited for both software and hardware rendering.

## 1 Introduction

Shadows provide important visual cues for the relative position of objects in a scene. Thus, it is not surprising that there has been a lot of work in the computer graphics literature on how to include shadows in interactive applications. Apart from precomputed diffuse shadow textures for each object, and some special purpose solutions like projected geometry [2] for large planar receivers, there are two general purpose shadow algorithms for interactive applications.

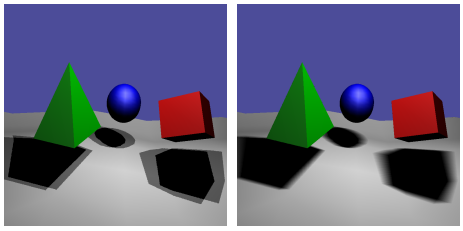
The first of these, shadow volumes [5] is an object-space method, while the second one, shadow maps [23] is a purely sampling based approach that works with depth images of the scene. Both techniques have their specific advantages and disadvantages. For example, while the shadow volume algorithm is generally very stable, it can introduce a large number of boundary polygons that significantly increase the geometric complexity of a scene. On the other hand, the shadow map algorithm has a low geometric complexity, however, numerical problems occur quite frequently. Furthermore, the hardware support for shadow maps has so far been restricted to very high end systems so that developers for the low end were forced to use shadow volumes, since these only require a minimal hardware feature set.

Variants to produce soft shadows for linear and area light sources are known both for the shadow volume and for the shadow map algorithm (see, for example [1, 4]) as well as for other texture-based methods [10]. These work by replacing the linear or area light source with a number of point light sources. In many cases, the light source does not subtend a very large solid angle as seen from any object point in the scene. This means that, especially in scenes with mostly diffuse materials, the local illumination caused by different samples of the light source differs only marginally, and thus a small number of light source samples should be sufficient. Nonetheless, the number of samples often has to be quite significant to obtain smooth penumbra regions. This is due to the fact

that, with $N$ light source samples, one can only obtain $N+1$ different levels of shadow: fully lit, fully shadowed (umbra), as well as $N-1$ levels of penumbra. Thus we will need to have a large number of light source samples for scenes with large penumbra regions, or the quantization into $N-1$ penumbra regions will become apparent.

So, while a small number of samples would be sufficient for the local shading process, we require a large number of samples to establish the correct visibility in the penumbra regions. This significantly increases the computational cost of soft shadows, and makes them infeasible for many interactive applications.

In this paper, we introduce a new soft shadow algorithm based on the shadow map technique. This method is designed to produce high-quality penumbra regions for linear light sources with a very small number of light source samples. It is not an exact method and will produce artifacts if the light source is so severely undersampled that the visibility information is insufficient (i.e. if there are some portions of the scene that should be in the penumbra, but are not seen by any of the light source samples). However, it produces believable soft shadows as long as the sampling is good enough to avoid these problems. Figure 1 gives a first impression of our technique.



(a) (b)

**Fig. 1.** (a): approximating a linear light source with two point lights. (b): our method, also using two light source samples.

The remainder of this paper is organized as follows. In Section 2 we review related work by other researchers. Then, in Section 3, we introduce our soft shadow technique, starting with a simple linear light source with two light source samples, eventually extending the technique to multiple samples.

## 2 Related Work

Since shadows are such an important visual effect, it is not surprising that a host of literature by many researchers is available on this topic (see [25] for a survey of different methods). In ray-tracing, a shadow ray is cast towards the light source to obtain a boolean visibility flag for point and directional light sources. This method can be extended to distribution ray-tracing for the rendering of soft shadows. This sampling approach, however, suffers from the quantization artifacts similar to the ones mentioned above for the shadow map and the shadow volume algorithm. In other words, with $N$ light source samples we can only discriminate $N-1$ levels of penumbra in addition to the umbra and the completely lit regions. However, this quantization is masked by noise if the sampling pattern on the light source is chosen differently for each illuminated surface point.

To eliminate both quantization and noise, some researchers use a geometrical analysis of the scene to find regions of the scene where the whole light source is visible, the whole light source is occluded (umbra), and regions where part of the light source is visible (penumbra). These methods work in object space by either generating discontinuities on the illuminated objects (discontinuity meshing along the lines of Heckbert [9]), by backprojecting the scene onto the light source (for example Drettakis and Fiume [7] and Stewart and Ghali [21]), or, more recently, by detecting singular points and lines on the light source itself (Ouellette and Fiume [14]).

After the discontinuities have been geometrically analyzed, the actual shading of each point can be performed analytically, or again using sampling. Common to discontinuity meshing and back projection is the large geometric complexity, which makes these approaches ill suited for interactive applications. Recently, there has also been some work by Parker et al. [16] on approximating the penumbra by manipulating the geometry of the occluders, and then assuming a point light source. This work does not yield the exact solution for the penumbra, but results in an approximation of high visual quality. It is similar to our work in that it attempts to render hight quality soft shadows with very few light samples.

Another method of generating soft shadows for off-line rendering is based on convolution, and has recently been introduced by Soler and Sillion [20]. While this method is much faster than the techniques described above, it is still far from interactive.

In the area of interactive computer graphics, we often see special purpose algorithms that are only adequate for very specific situations, such as the projected geometry approach [2], which only works for shadows cast onto large planar objects. In addition to these methods, there has recently been some work on generating shadows from image-based scene representations, for example by Keating and Max [11].

Among the two general purpose algorithms for interactive shadows we find Crow's object space shadow volume method [5]. A variant of this algorithm for graphics hardware using the stencil buffer has later been developed by Diefenbach and Badler [6], and some additional fixes for special situations, where the near plane of the view frustum straddles one of the shadow volumes, have been introduced by Udeshi and Hansen [22]. Soft shadows can be implemented with shadow volumes by sampling the light source with point lights [1], but this bears the above-mentioned sampling problems.

To reduce the geometric complexity of shadow volumes, McCool generates shadow volumes directly from a depth image of the scene in some very recent, unpublished work [13]. He uses an edge detection algorithm to obtain the discontinuities in the depth map, which then act as the boundary polygons of a shadow volume. In this paper, we also use edge detection in the shadow map to obtain discontinuities. However, in contrast to McCool we use this information for rendering penumbra regions rather than extracting shadow volume information.

The other frequently used shadow algorithm in interactive applications are shadow maps [23]. Here, the shadow test is reduced to a comparison of a point's actual depth in the light source coordinate system to a reference value stored in a depth image. In Williams' original paper this reference value is the depth of the visible surface along a ray through the light source position. Reeves et al. improved the shadow map technique by anti-aliasing the shadow boundaries using a technique called *percentage-closer filtering* [17]. Common artifacts of the shadow map algorithm, like self-shadowing of surfaces and missing shadows due to numerical problems in the depth comparison were resolved by Woo [24]. He modified the algorithm to use a shadow map where the reference value is actually a weighted sum of the visible surface and the first surface point behind it. This improved the numerical stability of the shadow map algorithm, because it mostly avoids depth comparisons of very similar values. Finally, Segal et al. [18] introduced hardware support for soft shadows on high-end graphics hardware. Without this dedicated support, shadow maps can be implemented on a fairly standard OpenGL pipeline as described by Brabec et al. [3]. This is the method we use for the hardware implementation of our soft shadow extension to shadow maps.

## 3 Soft Shadow Maps

For the following discussion, we consider a scene with a single linear light source. As discussed in the Section 1, we want to assume that the visibility term can be separated from the local illumination part, and that the latter is smooth enough to be represented by very few light source samples. The task is then to reconstruct the visibility term with a high quality, while only using a small number of light source samples. For the moment we restrict ourselves to the simplest case where we use only two samples residing at the vertices of the linear light.

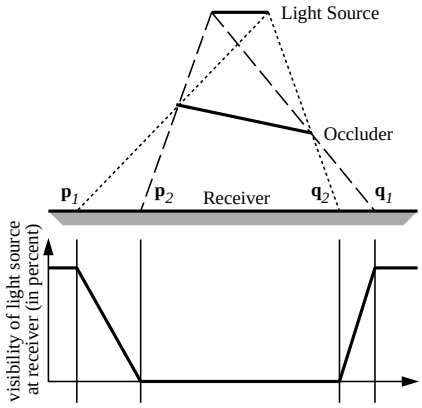



**Fig. 2.** Top: a simple scene with a linear light source, an occluder and a receiver polygon. Bottom: the percentage of visible portions of the light source as a function of the location on the receiver.

Also suppose for the moment that we have an efficient way of computing for each point in the scene the percentage of the light source visible from that point (Sections 3.1 and 3.2 describe an efficient method for computing an approximation of this term). An example for such a visibility function in a simple 2D scene is shown in Figure 2. In order to render soft shadows in this setting efficiently, we can extend the shadow map algorithm as follows.

First, generate a shadow map for each of the two light samples, considering these as two distinct point lights. Each of the texels in such a shadow map corresponds to one surface point that is visible from the respective light source sample. Now we add a second channel to each of the two shadow maps. This channel describes the percentage visibility of the whole light source for each of these object points. More precisely, this channel has the following properties:

- If a surface point is seen by exactly one of the light samples, the corresponding visibility entry in that shadow map reflects the percentage of the linear light source visible from that point (or an approximation thereof).
- If a surface point is seen by both samples, the sum of the corresponding visibility channels represents the percentage visibility. In the algorithm presented in Section 3.2 we assume that the whole light source is unobstructed if both samples are visible, that is, the visibility is 100%. In our algorithm this means that both maps will get a value of 50%. Note that it would be possible to drop this assumption if a different way of computing the visibility channel was used.

- Surface points not seen by any of the two light samples are not represented in any of the two maps. Therefore these are assumed to be in the umbra.

Put differently, the shadow map not only contains information about *which* object points are visible from a given point light, but also a percentage value that describes *how much* of the whole linear light source can be seen by that point. For any given object point, the sum of these visibility terms from the two point lights should then result in the value of the function plotted at the bottom of Figure 2.

Based on these two-channel shadow maps, we can now formulate a variant of the shadow map algorithm for soft shadows. Let $S_1$ and $S_2$ be two shadow maps including such visibility channels $V_1$ and $V_2$, one for each of the two point light sources $L_1$ and $L_2$. The shading of a particular point $\mathbf{p}$ in the scene then proceeds according to the following algorithm:

```
shade( p ) {
  if( depth1(p)> S1[p] )
    l1= 0;
  else
    l1= V1[p] * localIllum(p,L1);

  if( depth2(p)> S2[p] )
    l2= 0;
  else
    l2= V2[p] * localIllum(p,L2);

  return l1+l2;
}
```

In this piece of pseudo code $S_i[\mathbf{p}]$ means looking up the reference depth value corresponding to $\mathbf{p}$ in shadow map $i$. Similarly $V_i[\mathbf{p}]$ means looking up the visibility value for $\mathbf{p}$. The depth of a point $\mathbf{p}$ in the respective light coordinate system is given as $\mathtt{depth}_i(\mathbf{p})$. From this code it is obvious that the proposed method is only marginally slower than shadow mapping with two point lights, assuming that the shadow maps including the visibility channels are provided.

The problem now is to determine how to generate these visibility channels in the first place. In principle, we could use any known object-space algorithm for this, including both analytical methods and sampling. While this may be a feasible approach for static scenes, dynamic environments require faster techniques. For this latter case we chose to use a a linear approximation for the transition in the penumbra regions. In the following we first motivate this approximation, before we describe the details of the method in Section 3.2.

### 3.1 Linear Interpolation of Visibility

One property of linear lights is that object edges parallel to it do not have a penumbra region. In other words, there is a sharp transition from umbra to fully lit regions for these edges. Furthermore, linear light sources have the advantage that the visibility considerations of a 3D scene can be reduced to 2D scenes. Consider the intersection of the scene with a plane containing the light source. If we can solve the visibility problem for all such planes, i.e. for the whole bundle of planes having the light source as a common line, then we know the visibility of the light source for all 3D points in the scene.

For a motivation of our algorithm for generating the visibility channels, consider the configuration in Figure 2, which contains a linear light source at the top, an occluder and a receiver polygon. In order to compute the correct penumbra, we have to determine for each point on the receiver, which percentage of the linear light source is visible from that point. This percentage is plotted as a function of the surface location at the bottom of Figure 2.

In this simple configuration, it is clear that we have two penumbra regions, one where the visibility varies from 100% at $\mathbf{p}_1$ to 0% at $\mathbf{p}_2$, and similarly from 100% at $\mathbf{q}_1$ to 0% at $\mathbf{q}_2$. In general, the transition from fully visible to fully occluded is a rational function, which becomes obvious by considering the simple case of a single occluder edge, as depicted in Figure 3.

Without loss of generality, the occluder edge is located at the origin (the slope of the occluder is not of importance), the light source is given by the formula $y_1 := mx_1 + t$, and the intersection of the receiver with the 2D plane in consideration is given by $y_2 := nx_2 + s$. From the constraint $x_1/x_2 = y_1/y_2$, which characterizes the point $(x_1, y_1)$ on the light source that is just visible from $(x_2, y_2)$, it follows that

$$x_1 = \frac{x_2 t}{nx_2 - mx_2 + s}. \qquad (1)$$

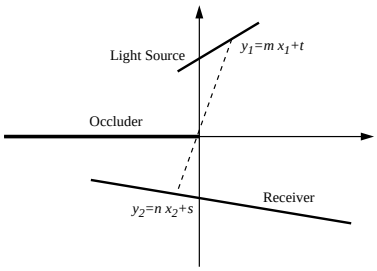



**Fig. 3.** A simple scene with a single occluder edge that can be used to characterize the change of visibility across a planar receiver that is not parallel to the linear light source.

This rational function simplifies to a linear one if the slopes $m$ of the light source and $n$ of the receiver are identical, i.e. if light and receiver are parallel as in Figure 2.

If light source and receiver are not parallel, the rational function has a singularity at the point where the receiver polygon intersects the line on which the linear light source resides. However, this is an area where the penumbra region collapses to zero size anyway. On the other hand, the regions for which we expect large penumbra regions are far away from this singularity, and there the rational function from Equation 1 behaves almost like a linear function.

### 3.2 Generating the Visibility Map

With this observation we can now formulate an algorithm for generating the visibility channels for the two shadow maps. The object points in one of the penumbra regions are of particular interest. In our simple setting, these are the object points seen by one of the two point lights, but not by both.

Now imagine we take the shadow map from the right sample point, triangulate all the depth samples, and warp all the resulting triangles into the view of the left point light, thereby using the depth buffer to resolve visibility conflicts. This is similar to an image based rendering algorithm along the lines of post-rendering 3D warping [12]. The resulting image will consist of two kinds of polygons: those corresponding to the real geometry in the scene, and "phantom polygons", sometimes also called "skins", which result from triangulating across depth continuities. Both types are depicted in Figure 4. The skins are shown as gray lines; the original surfaces are colored black.

While the original polygons are the desired result in image-based rendering and the skins are an artifact, it is the skins that are of particular interest to us. Wherever they are visible in the destination image (i.e. in the image corresponding to the left point light), a penumbra region is located! What is more, we know qualitatively what the visibility value should be for points in this region. Since the skins are generated by depth discontinuities in the source shadow map, they always connect an occluder polygon and a receiver polygon. Points in the penumbra region that are closer to the occluder in the reprojected image see less of the linear light than points closer to the receiver polygon.

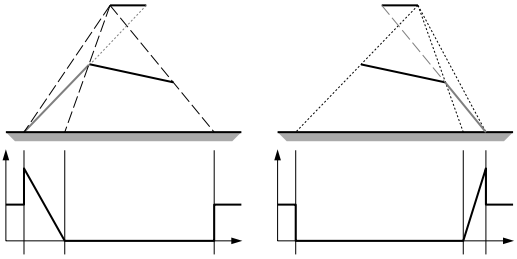
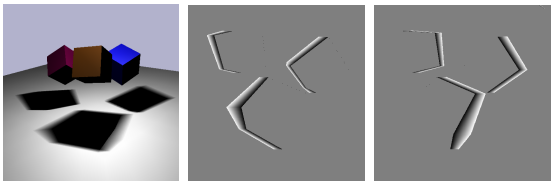


**Fig. 4.** Top: skin polygons warped from one depth map into the other. Bottom: visibility contributions for both point lights at each point on the receiver, using the presented method to generate the visibility channel of the shadow maps.

If we assume a linear transition between fully visible and fully occluded, as argued in the previous section, then we can generate the visibility channel as follows: First, we need to find the depth discontinuities in the shadow map of the right point light, which can be done using standard image processing techniques [8], and can be performed at interactive speed. The resulting skins then need to be reprojected and rendered into the visibility channel of the left shadow map. During rendering we Gouraud-shade the skin polygons by assigning the value 0 to vertices on the occluder and the value 1 to vertices on the receiver. This can be done either using a software renderer, or using computer graphics hardware and a depth buffer algorithm. In the latter case, it is possible to generate the visibility channel at interactive frame rates. We repeat the whole procedure to project the discontinuities from the depth buffer of the left point light to the right shadow map.

A final consideration for the generation of the visibility channel is the treatment of completely lit and completely shadowed object points. The latter case is simple. Since points in the umbra are not seen by any of the two light sources, they will fail the shadow map tests for both point light sources, and therefore be rendered black (or with an ambient color only). Completely lit points on the other hand, are seen by both point lights, and in this case we are going to assume that the whole linear light is visible. Thus, the visibilities for both lights need to sum up to 1. One way of doing this is to give these points a visibility of 0.5 in both shadow maps. This can easily be implemented by initializing all entries in the visibility channel to 0.5 before starting to warp the skin polygons.



(a) (b) (c)

**Fig. 5.** (b) and (c): the visibility channel for the two point lights for the scene in (a).

Figure 4 shows for both samples on the light source the visibility contribution to each point on the receiver of the 2D scene from Figure 2. These contributions can be

generated using the just described algorithm to generate the visibility channels. Furthermore, Figure 5 (b) and (c) show the visibility channels for a linear light source in a 3D scene. The scene itself is depicted in Figure 5 (a).

### 3.3 Linear Light Sources With More Samples

The restriction of this algorithm for generating the visibility map is that object points seeing portions of the linear light source, but none of the two point lights at its ends, will appear to lie in the umbra. Moreover, there are situations where this results in discontinuities, as depicted in Figure 6. These artifacts result from a severe undersampling of the light source, with the consequence that important visibility information is available in neither of the two shadow maps.

The consequence from this observation is to increase the sampling rate by adding in one or more additional point samples along the linear light source. For example, if we add in a third point light in the center of the linear light, we have effectively subdivided the linear light into two smaller linear lights that distribute only half the energy of the original one. If we treat these two linear light segments independently with the algorithm described in the previous section, we get the situation depicted in Figure 7 for the same geometric setup as in Figures 2 and 4.

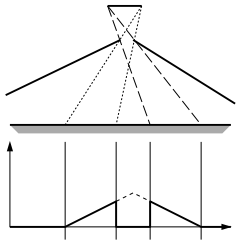


**Fig. 6.** An example of failure due to undersampling of the light source which causes some portions of the penumbra to end up in full shadow. These artifacts can only be resolved by increasing the sampling rate on the light source.

The top row of the figure corresponds to the rendering of the left half, while the bottom row corresponds to the right half of the linear light. Note that the light source on the right side of the top row, and the one on the left side of the bottom row correspond to the same point light, namely the one inserted at the center of the linear light. Therefore it is possible to combine these two point lights into a single one with twice the brightness, by summing together the visibility channels (the depth channels are identical anyway!).

With this general approach we can add in as many additional sample points on the linear light source as are required to avoid the problems of points in the penumbra that are not seen by any light source sample. To generate the visibility channel for one of the sample points, we need to consider only the depth discontinuities (skins) of those samples directly adjacent to this point. For example, in Figure 7, the discontinuities from the rightmost sample do not play a role for the visibility map of the leftmost sample and vice versa.

## 4 Results

We have implemented the approaches described in this paper for two different versions. Firstly, we have a software implementation of the method in a ray-tracer, and secondly we also have an implementation that utilizes OpenGL graphics hardware for the rendering. This latter method is an extension of an OpenGL implementation of the standard shadow map algorithm described in [3]. For this hardware-based technique, we use SGI workstations (SGI Octane, O2 and Visual Workstation), which all have support for the

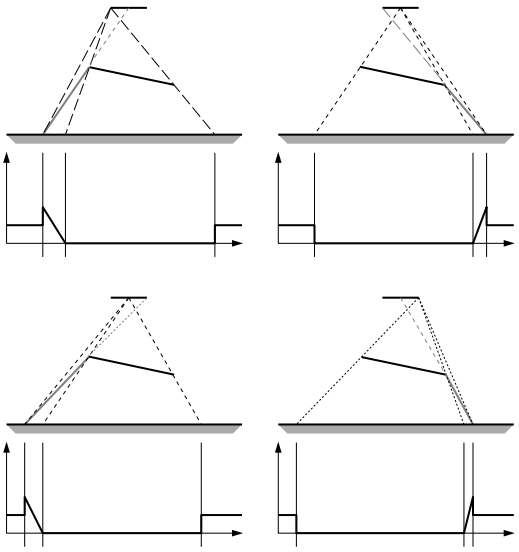


**Fig. 7.** By inserting an additional point light in the center, we have effectively reduced the problem to two linear lights of half the length and intensity. Top two: left half of linear light. Bottom two: right half.

OpenGL imaging subset [19].

This imaging subset, which allows to perform convolutions of images, is not required for the rendering using an existing shadow map, but it is helpful for generating the visibility channels on the fly in dynamic environments. Remember that this task requires us to find discontinuities in the depth channel of the shadow map in order to determine the skin polygons. Using a convolution with a $3 \times 3$ Laplacian-of-Gaussian (LoG) edge detection filter helps us to perform this task very efficiently.

A comparison of the different variants of our soft shadowing algorithm is depicted in Figure 8. The top row shows the images that would be generated by simply approximating a linear or area light source with a number of point lights. The bottom row shows results from our algorithm with the same number of light source samples. It can be seen that the quality of the penumbra regions is much higher in all cases. The left column shows the result from approximating a linear light source with two samples. It has been chosen such that overlapping penumbra regions exhibit the undersampling artifacts described in Section 3.3. These artifacts disappear as a third light source sample is inserted, as shown in the center column.

Finally, in the right column, we have experimented with a triangular area light source, which we have approximated by three linear lights corresponding to the edges of the triangle. As we have shown in Section 3.1, the linear transition of visibility in the penumbra region was an approximation of the true rational function for the case of linear light sources. For area light sources, this transition is, in general, a quadratic rational function, so that the linear approximation of our algorithm is a really crude approximation. Nonetheless, the results seem to indicate that it may still be useful for certain applications. The problem in finding a better approximation is that the shape of the quadratic rational function depends on the shape of the triangle and on the relative orientation of light source and occluder edges. Taking these into account during the

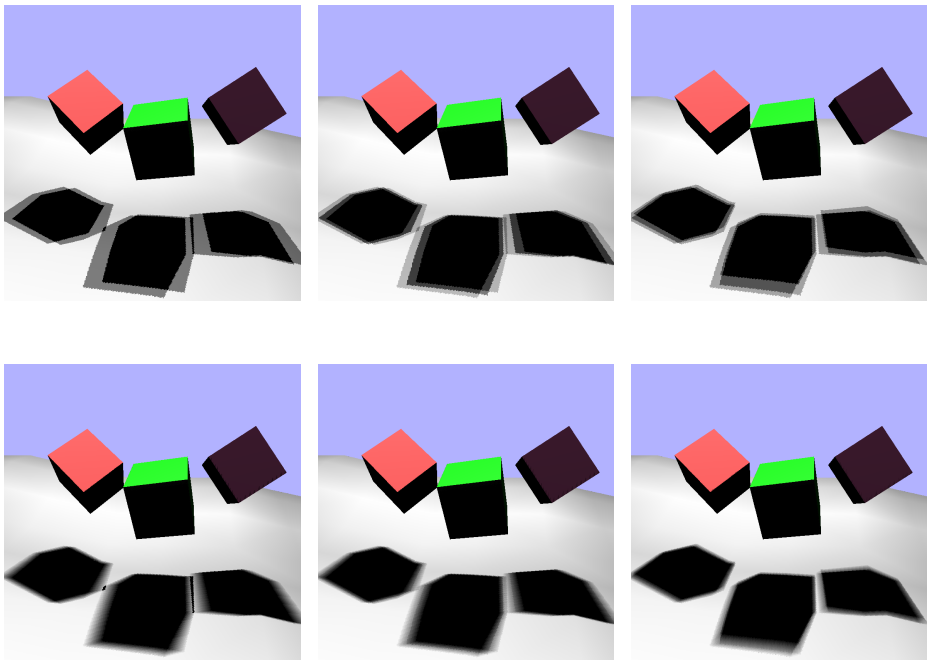


**Fig. 8.** Comparison of different soft shadow techniques. Top row: simple approximation of the light source by several point lights. Bottom row: the method proposed in this paper. Left column: a linear light source approximated with two samples. Note the artifact introduced where the soft shadows overlap. This is due to undersampling (see Section 3.3). Center: The artifacts disappear as a third light source sample is introduced. Right: Applying the technique to a triangular area light (see text for details).

generation of the visibility channel would slow down the algorithm considerably.

Figure 9 in the color plates compares a high-quality solution for the visibility of a scene with one linear light source, one blocker and one receiver with our method. Figure 9a shows the solution of a ray-tracer using 200 light source samples to determine the visibility in every point on the receiver. Figure 9b depicts the result of a software implementation of our method. In contrast to the OpenGL implementation, the software implementation allows for having the same per-pixel shading as in the ray-traced image. Figure 9c shows a ray-traced solution with 10 uniformly spaced samples for comparison. With a shadow map resolution of $500 \times 500$, our method including map generation and rendering of $2 \times 1700$ skin polygons takes about as long as ray-tracing with 6 samples.

Figure 10 on the color page shows some more complex scenes rendered with the OpenGL-based implementation. Once the shadow maps are computed, the rendering times using our soft shadow algorithm are identical to those for rendering hard shadows with the same number of point lights. This is true for all scenes. Therefore, our algorithm can be used for interactive walkthroughs with no additional cost. The scenes in Figure 10 can be rendered at about 20 fps, provided the shadow maps do not have to be regenerated for every frame.

Building the shadow maps in a dynamic environment is obviously more expensive

for our algorithm, since the visibility channels need to be generated as well. This requires edge detection within the depth maps, as well as rendering a potentially large number of skin polygons. The cost of generating the shadow maps therefore depends on the scene geometry. It varies from $< 1/20$ seconds for the simple scene in Figure 8 to about 2 seconds for the area light source in the jack-in-a-box scene in Figure 10. These numbers include the time for rendering the scene to generate the depth maps.

## 5 Conclusions

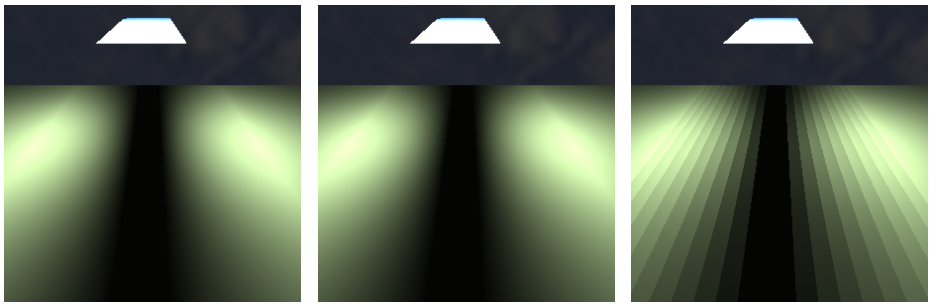
In this paper we presented a new soft shadow algorithm based on the shadow map method. It is designed to produce high-quality penumbra regions for linear light sources with a small number of light source samples. We demonstrated that the method works efficiently and produces high-quality penumbrae for non-trivial scenes. The method can be applied both to software and hardware rendering, and we have demonstrated that it is possible to achieve interactive frame rates in the latter case.

It remains an open research problem to determine the best place to insert samples into a linear light source. Recent work by Ouellette and Fiume [15] seems to be a promising starting point for determining those locations where a new sample point would improve the overall quality of the penumbra regions the most. In the future, it would be interesting to extend the method to area light sources. The key problem there is that a linear visibility transition, as used in this paper, is usually not a good assumption in this case.

## References


1. P. Bergeron. A general version of crow's shadow volumes. *IEEE Computer Graphics and Applications*, 6(9):17–28, 1986.
2. James F. Blinn. Jim Blinn's corner: Me and my (fake) shadow. *IEEE Computer Graphics and Applications*, 8(1):82–86, January 1988.
3. Stefan Brabec, Wolfgang Heidrich, and Hans-Peter Seidel. OpenGL shadow maps. Technical Report TR 2000-4-002, Max-Planck-Institute for Computer Science, March 2000.
4. L. S. Brotman and N. I. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics and Applications*, 4(10):71–81, October 1984.
5. Franklin C. Crow. Shadow algorithms for computer graphics. In *Computer Graphics (SIGGRAPH '77 Proceedings)*, pages 242–248, July 1977.
6. Paul J. Diefenbach and Norman Badler. Pipeline Rendering: Interactive refractions, reflections and shadows. *Displays: Special Issue on Interactive Computer Graphics*, 15(3):173–180, 1994.
7. George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In *"Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 223–230, July 1994.
8. Rafael Gonzalez and Richard Woods. *Digital Image Processing*. Addison-Wesley, 1992.
9. Paul Heckbert. Discontinuity meshing for radiosity. In *Rendering Techniques '92 (Proc. of Eurographics Rendering Workshop)*, pages 203–226, May 1992.
10. Paul Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, Carnegie Mellon University, January 1997.
11. Brett Keating and Nelson Max. Shadow penumbras for complex objects by depth-dependent filtering of multi-layer depth images. In *Rendering Techniques '99 (Proc. of Eurographics Rendering Workshop)*, pages 197–212, June 1999.
12. William Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3D warping. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 7–16, April 1997.

13. Michael McCool. Shadow volume reconstruction. Technical Report CS-98-06, University of Waterloo, 1998. Available from http://www.cgl.uwaterloo.ca/~mmccool/.
14. Marc Ouellette and Eugene Fiume. Approximating the location of integrand discontinuities for penumbral illumination computation with area light sources. In *Rendering Techniques '99 (Proc. of Eurographics Rendering Workshop)*, pages 213–224, June 1999.
15. Marc Ouellette and Eugene Fiume. Approximating the location of lntegrand discontinuities for penumbral illumination with linear light sources. In *Graphics Interface '99*, pages 66–75, June 1999.
16. Steven Parker, Peter Shirley, and Brian Smits. Single sample soft shadows. Technical Report UUCS-98-019, Computer Science Department, University of Utah, 1998. Available from http://www.cs.utah.edu/vissim/bibliography/.
17. William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 283–291, July 1987.
18. Marc Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadow and lighting effects using texture mapping. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 249–252, July 1992.
19. Mark Segal and Kurt Akeley. *The OpenGL Graphics System: A Specification (Version 1.2)*, 1998.
20. Cyril Soler and François X. Sillion. Fast calculation of soft shadow textures using convolution. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 321–332, July 1998.
21. James Stewart and Sherif Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 231–238, July 1994.
22. Tushar Udeshi and Charles Hansen. Towards interactive, photorealistic rendering of indoor scenes: A hybrid approach. In *Rendering Techniques '99 (Proc. of Eurographics Rendering Workshop)*, pages 63–76, June 1999.
23. Lance Williams. Casting curved shadows on curved surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, pages 270–274, August 1978.
24. Andrew Woo. *Graphics Gems III*, chapter The Shadow Depth Map Revisited, pages 338–342. Academic Press, 1992.
25. Andrew Woo, Pierre Poulin, and Allain Fornier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.



(a) ray traced, 200 samples (b) our method, 2 samples (c) ray traced, 10 samples

**Fig. 9.** A comparison of ray-traced images and our method for a scene with one blocker and one linear light (not visible).

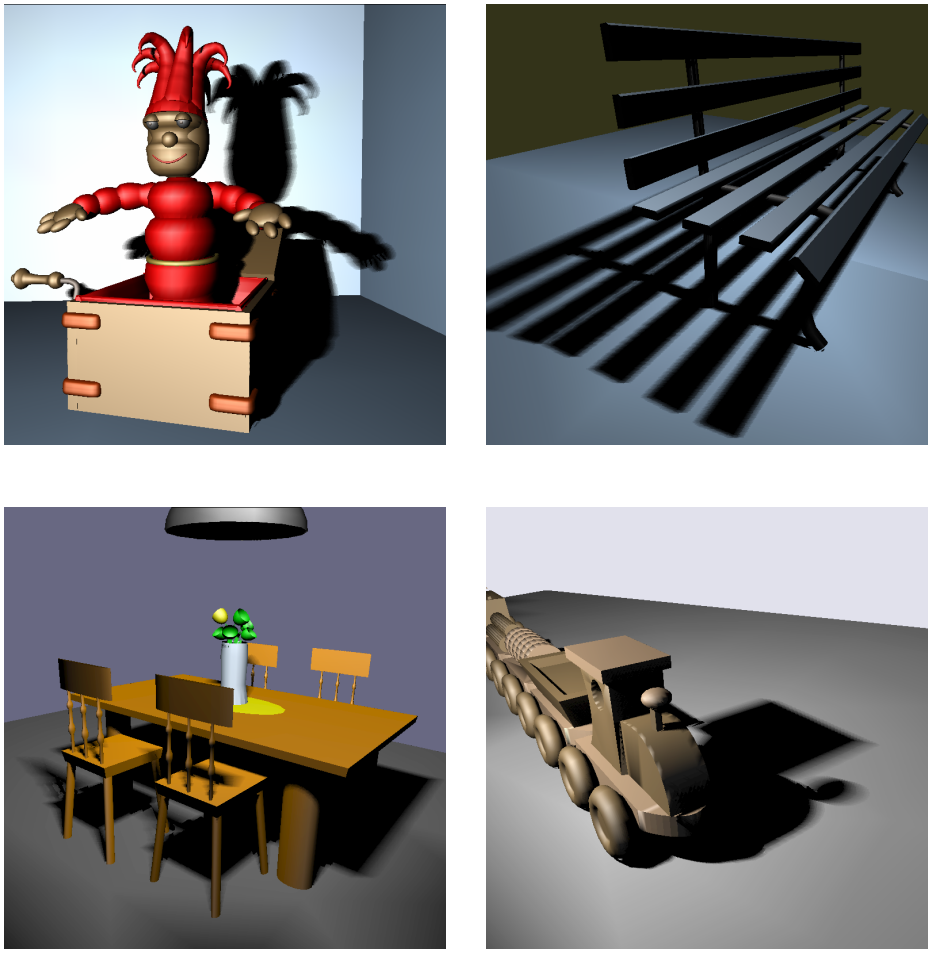


**Fig. 10.** Some more examples of our method with two samples per light.