The Soft Shadow Occlusion Camera

Qi Mo
University of Iowa
qmo@cs.uiowa.edu

Voicu Popescu
Purdue University
popescu@cs.purdue.edu

Chris Wyman
University of Iowa
cwyman@cs.uiowa.edu

Abstract

A fundamental challenge for existing shadow map based algorithms is dealing with partially illuminated surfaces. A conventional shadow map built with a pinhole camera only determines a binary light visibility at each point, and this all-or-nothing approach to visibility does not capture penumbral regions. We present an interactive soft shadow algorithm based on a variant of the depth discontinuity occlusion camera, a non-pinhole camera with rays that reach around blockers to sample normally hidden surfaces. Our soft shadow occlusion camera (SSOC) classifies a fragment on a continuum from fully visible to fully hidden, as seen from the light. The SSOC is used directly in fragment illumination computation without building an explicit "soft shadow map." This method renders plausible soft shadows at interactive speeds under fully dynamic conditions.

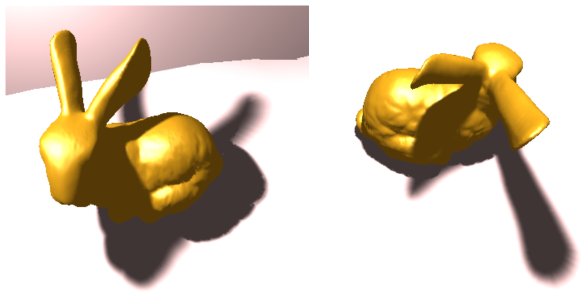


Figure 1. Interactive soft shadows using the soft shadow occlusion camera run at 85 fps for this scene. Note the self-shadowing and the varying size of overlapping penumbrae.

1 Introduction

Soft shadows play an important role in rendering, as they provide an increased sense of realism, produce object-object contact cues [19], and improve spatial perception [22, 33]. However efficient soft shadow rendering proves difficult, as computation involves visibility queries from every point in the scene to potentially complex light sources. Thus, most interactive applications today rely on *hard shadows* arising from a simple point light. For such lights, only a single binary visibility query is necessary. While this enables quick renderings, the resulting shadows lack *penumbra*, the regions of transition between full shadow and full illumination.

As real-world lights have a non-negligible area, realistic shadow renderings must compute how much of the light is visible, leading to regions of full illumination, partially illuminated penumbra, and fully shadowed *umbra*. Generally, these computations cannot be solved analytically or determined via a single visibility query. Offline renderers often generate soft shadows by numerically integrating visibility by averaging over many binary queries.

While interactive applications can repeatedly sample the light [1, 16, 18], the costs increase linearly with the sampling rate, quickly limiting shadow quality and scene complexity. Thus, a variety of researchers have sought to approximate the soft shadow computation using a single sample [8, 14, 27, 36]. These techniques add penumbral regions to an existing hard shadow. As the light grows, these techniques exhibit artifacts due to the fixed umbra size. Cai et al. [7] extended these methods to reduce this problem, at the cost of storage for more map layers.

This paper presents a new soft shadow approach inspired by a non-pinhole camera introduced by Popescu and Aliaga [28], the *depth discontinuity occlusion camera* (DDOC). A DDOC camera bends light around depth discontinuities to allow "nearly visible" points to appear in the final rendering. When positioned at the light, this camera allows both fully and partially illuminated points to be visible in a single rendering.

We introduce the *soft shadow occlusion camera* (SSOC), a non-pinhole camera constructed from the center of the light that efficiently provides a quality penumbral approximation. The partial visibility queries for these penumbrae are answered directly, bypassing the need of costly aggregation of multiple point-to-point visibility queries and avoiding construction of a reference image to serve as a soft shadow map. The SSOC enables rendering realistic soft shadows at interactive rates (see Figure 1) and requires no precomputation, thus supporting fully dynamic scenes.

2 Previous Work

Efficiently rendering shadows has long been an active area of research. We refer readers to comprehensive surveys on shadows [35] and soft shadows [15] for detailed discussion of prior work. We focus our discussion on non-pinhole cameras and soft shadow methods most relevant to our method.

2.1 Interactive Shadows

Today's applications rely on either shadow maps [17,34] or shadow volumes [10,30] to provide interactive shadows. Shadow mapping requires a depth map rendered from the light position, and scene points are projected onto this depth map to decide visibility from the light. Shadow volumes describe virtual geometry that bounds shadowed regions; objects inside these volumes are shadowed while those outside the volumes are illuminated.

Interactive soft shadow techniques typically build on one, or both, of these methods. Penumbra wedges [3,4] extend shadow volumes by replacing each virtual shadow quad with a penumbra wedge that provides a gradual change from fully illuminated to fully shadowed regions. While this method is the most realistic interactive soft shadow algorithm, its cost grows linearly with the number of wedges and becomes fill-rate limited as wedge geometry increases. Clipping and culling techniques can significantly reduce the necessary fill rate [2,9,23], though perhaps not enough for complex scenes.

Another possibility is to sample the light at multiple points, creating a shadow map for each, and to average over queries from all maps [16]. Alternatively, multiple light samples could be merged into a layered shadow map [1,20,32] to reduce per-pixel lookup costs. In return, however, this method adds an expensive preprocess that precludes dynamic scenes. Both methods produce realistic soft shadows, but usually at a prohibitive cost.

Projecting occluders onto the light surface allows analytically determining light visibility. While accurate, this backprojection [11] process becomes quite costly in complex scenes. Recent work [5,6,13] speeds backprojection by treating shadow map texels as the occluders, instead of using explicit geometry. This significantly increases backprojection speed, but is prone to cracking artifacts.

Other approaches [8,14,36] augment the shadow map's hard shadow with a plausible looking penumbra. These methods observe that objects visible from the light center fall into two categories: fully illuminated or partially shadowed. By augmenting the shadow map with a second map (a *smoothie buffer* or *penumbra map*), plausible soft shadows can be rendered by identifying completely shadowed points and consulting the secondary map to determine the light contribution for all other points. Unfortunately, these techniques break as the light size increases, when the umbra should shrink. Cai et al. [7] note that for complex scenes a single map is not sufficient to store all penumbral regions, so they propose a multi-layered approach.

2.2 Non-Pinhole Cameras

Although camera models are essential components of the graphics pipeline, relatively little research effort has explored alternatives to the pinhole camera. Exceptions include non-pinhole cameras developed in the context of image-based modeling and rendering, such as light fields [12,24], which are 2D arrays of planar pinhole cameras; layered depth images [31], which are planar pinhole cameras that store a variable number of samples along each ray; and multiple center of projection images [29], which are obtained by moving a slit camera along a path.

These camera models capture more than the surface samples visible from a single viewpoint and therefore could be used, in principle, in the context of soft shadows. However, rendering with such cameras implies a large number of feed-forward passes, making them too slow for the dynamic scenes in interactive applications. For example, an LDI constructed from the center of the light source provides hidden samples from the inner penumbra, but construction requires rendering from several viewpoints and merging the results. Unlike earlier non-pinhole cameras, the depth discontinuity occlusion camera [28] provides fast, unambiguous projection, which allows rendering using the feed-forward graphics pipeline. The soft shadow occlusion camera described in Section 3.2 adapts the DDOC to the context of soft shadows.

3 Occlusion Camera Soft Shadows

Standard depth maps, as used in shadow mapping, store a concise representation of a scene as an image—every texel stores the distance from the light to the nearest object along the texel ray. This image-based representation typically scales better with geometric complexity than object-space representations. But this approach implicitly assumes the use of a point light source, as traditional hardware-accelerated rendering relies on a pinhole camera to render the depth map.

Previous shadow map based soft shadows attempt to overcome the point light limitation in one of two ways: by augmenting a shadow map with additional maps storing penumbra information or by backprojecting shadow map texels onto the light. Neither approach truly solves the problem, namely the need for scene information missing due to occlusions in the shadow map.

Interestingly, image-based rendering by 3D warping [25] performs a similar task to soft shadow rendering, namely recreating scenes rendered from arbitrary viewpoints based upon a small number of images. While many of the solutions (e.g., multisampling and layered depth images) have been explored by both communities, rendering researchers have generally avoided exploring the use of non-pinhole cameras.

One camera model in particular, the DDOC, captures needed information about barely hidden samples close to depth discontinuities. In the context of image-based rendering, the DDOC enables rerendering a scene within a small locus of viewpoints based upon a single reference image. Our key observation is that a small area light source is the light-space analogy of this locus of viewpoints. Thus, placing a DDOC at the center of the light provides the information necessary to compute soft shadows directly, in a single image.

Section 3.1 reviews the non-pinhole DDOC model, followed by a description of our soft shadow occlusion camera in Section 3.2.

3.1 The Depth Discontinuity Occlusion Camera

Image-based rendering aims to capitalize on previously acquired or computed color data to expedite rendering of a scene from novel viewpoints. A single reference image is not sufficient, as even small viewpoint translations expose new surfaces not sampled by the reference image. Avoiding these disocclusion errors by processing additional reference images has high and unpredictable cost, which defeats the purpose. The DDOC avoids disocclusion errors by constructing single-layer depth images that sample not only visible surfaces but also surfaces “nearly visible” from the reference viewpoint.

The DDOC model renders images identically to a standard pinhole camera, except near discontinuities. In those regions, rays “bend” around the discontinuity to view geometry hidden in standard pinhole renderings (see Figure 2). The resulting images appear similar to standard renderings, except in neighborhoods around discontinuities which condense the visible and nearly visible samples.

Popescu and Aliaga [28] set a user parameter to define the maximum image-space distortion allowable near discontinuities. This specified magnitude controls the locus size around the reference view where novel views exhibit minimal disocclusion errors.

Building a DDOC requires creation of an image-space

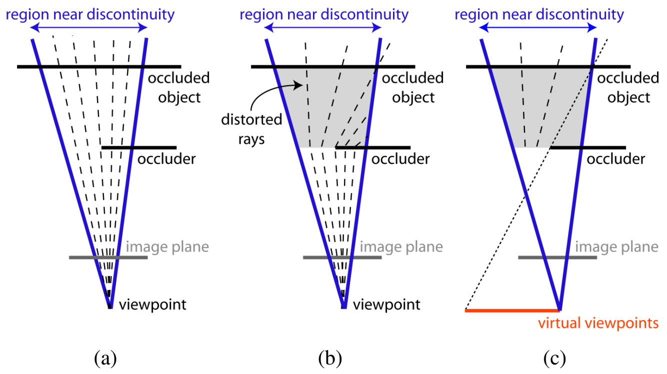


Figure 2. Ray behavior near edge discontinuities: (a) rays from a pinhole camera consist of a single line segment, whereas (b) rays from a DDOC model consist of two line segments. (c) The DDOC distortion increases linearly from the occluder to the occluded surface, and allows image-based rendering from various nearby virtual viewpoints.

distortion map to describe which regions of a standard pinhole rendering require geometrical distortion. Map creation involves first identifying discontinuities, finding the discontinuity edge normals, and finally splatting this information throughout potential distortion regions. The resulting DDOC is used to create a reference image by distorting each vertex based on data found by projecting it into the distortion map. This process effectively pulls out nearly visible samples for imaging by the camera (see Figure 3).

3.2 The Soft Shadow Occlusion Camera

A camera that pulls out nearly visible samples fits naturally with soft shadow algorithms, particularly with techniques such as penumbra maps [36] and smoothies [8] that only act on geometry visible in the shadow map. These techniques approximate only the outer penumbra simply because they lack information about the inner regions. By extending the shadow map to show normally hidden geometry, the extra information allows the trivial extension of these techniques to approximate both inner and outer penumbra. While building a DDOC model is interactive, rendering a reference image with the camera is not, so trivial shadow extensions that rely on these images are currently infeasible.

Creating a reference image involves the classic feed-forward steps of vertex projection followed by rasterization. Although DDOC vertex projection is inexpensive, rasterization in the distorted domain is not. As distortion occurs on a per-fragment basis, triangles may no longer have straight edges. For offline DDOC reference image construction, ras-

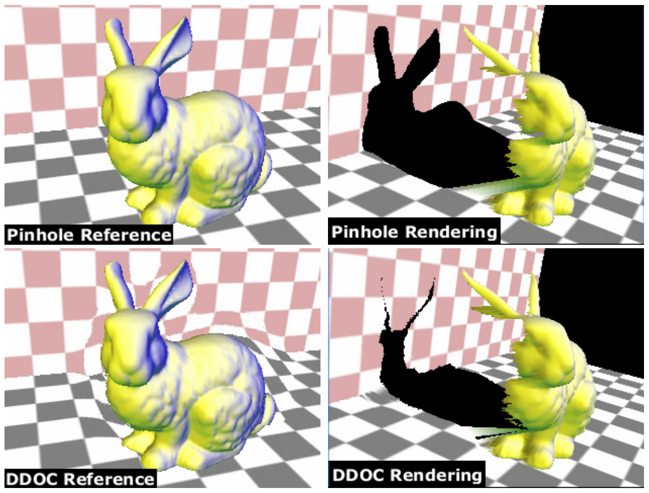


Figure 3. Using a pinhole camera to capture a reference image for image-based rendering via warping leads to novel views with significant missing information. A DDOC camera distorts some of this missing geometry so it is visible in the reference. Note how the information still missing looks remarkably similar to the umbra of a shadow under illumination from a small area light.

terization in the distorted domain relies on subdividing the scene until edge deformation is insignificant, allowing triangles to be rasterized conventionally. For most interactive applications, such extensive subdivision is infeasible. Note that for the simpler single pole occlusion camera [26] the distortion function is invertible, enabling efficient rasterization in the distorted domain. However, that camera model is too simple to model penumbral effects.

We observe that an actual occlusion camera reference image is unnecessary for rendering shadows. In an image-based rendering context, the reference image provides a compact and high-quality scene approximation that remains valid over a continuum of viewpoints around the reference viewpoint. In the context of shadows, the scene is rendered in the usual undistorted domain defined by the eye's pinhole camera. All that is needed is a fast and high-quality approximation of a fragment's light exposure.

We achieve this with a soft shadow occlusion camera placed at the center of the light. A distortion map specifies the SSOC and acts like a regular shadow map, except in the vicinity of depth discontinuities. By associating the sizes of distortion regions with light and occluder size and placement, we can construct the SSOC so that a fragment that projects to an undistorted region can be ruled as fully lit or shadowed as per standard shadow mapping. Illumination for other fragments directly corresponds to the distortion magnitude specified in the SSOC distortion map.

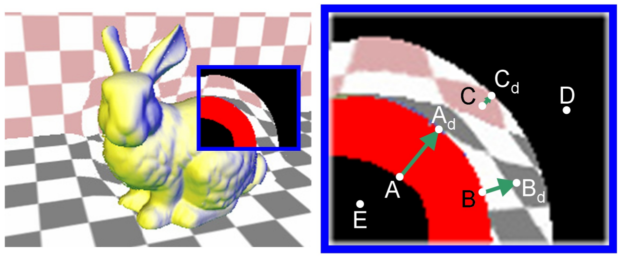


Figure 4. Consider the views from Figure 3. A pinhole camera only images half the geometry from penumbral regions. With a SSOC camera, all the geometry in the penumbra is warped so it is visible near silhouettes. Points A, B, C, D, and E are distorted varying amounts by the SSOC or processed with standard shadow mapping, depending on where they fall in the distortion map.

Figure 4 illustrates usage of the distortion map for a portion of the occlusion camera reference image from Figure 3. Pixels outside the edge region are set to black, and those inside the inner penumbra appear in red. We provide the occlusion camera reference image for illustration purposes; it is not needed for shadow computations. An occluded point **A** that projects between the umbral and penumbral regions is distorted the most, to $\mathbf{A}_d$. Points **B** and **C** are progressively more illuminated, and are distorted less, to points $\mathbf{B}_d$ and $\mathbf{C}_d$. Points **D** and **E** fall outside the distortion region and are therefore processed by conventional shadow mapping.

3.2.1 SSOC Distortion Map Construction

The DDOC distortion map [28] stores a five-tuple $(dir_u, dir_v, z_n, z_f, d_f)$ that specifies a maximal distortion in the direction (dir_u, dir_v) with magnitude varying linearly in $\frac{1}{z}$. The magnitude starts from zero at the near point z_n and reaches a maximal distortion d_f at the far point z_f. The maximal distortion $d_{max}(\mathbf{p}_z)$ of a point $\mathbf{p}$ with an eye-space z-value of $\mathbf{p}_z$ can be computed:

$$d_{max}(\mathbf{p}_z) = \begin{cases} 0 & \text{when } \mathbf{p}_z < z_n \\ \left(\dfrac{1/z_n - 1/\mathbf{p}_z}{1/z_n - 1/z_f}\right) d_f & \text{when } z_n \le \mathbf{p}_z \le z_f \\ d_f & \text{when } \mathbf{p}_z > z_f . \end{cases}$$

This maximal distortion only occurs for points **p** at the innermost edge of the distortion region (e.g., the dotted line in Figure 2(c) and point **A** in Figure 4). For rays further from the discontinuity, the distortion linearly shrinks to zero.

Popescu and Aliaga use a six-step process to create this map. However, we observe this process effectively performs three basic operations: 1) detecting silhouette edges, 2) extruding these edges and splatting them into the distortion map, and 3) cleaning up the distortion map (e.g., splat resizing). To make this process more amenable to hardware acceleration and improve robustness for such implementations, we suggest an implementation different than theirs:

1. Compute silhouette edges on CPU in light-space,
2. Create a standard shadow map z-buffer,
3. Extrude silhouettes along edge normals, creating quads perpendicular to the light's viewing direction,
4. Render the quads into the distortion map, storing an 6-tuple specifying the required warp, and
5. Resolve conflicts between overlapping quads using a depth test based upon the distance to the silhouette.

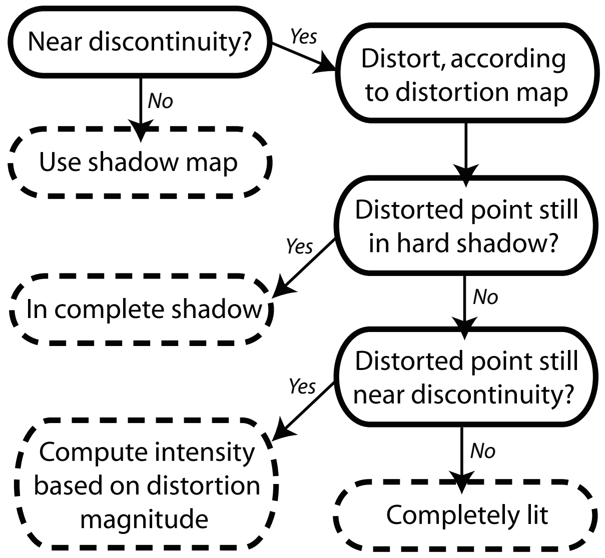
In particular, by performing silhouette detection on the CPU our method avoids robustness issues (e.g., thresholds) with the image-based edge detection used by the original DDOC model. Along with silhouette detection we explicitly determine silhouette normals from the geometry, which eliminates discretization artifacts introduced by an image-based approach. Finally, we use silhouette quads instead of per-pixel silhouette splats to render data into the distortion map. This reduces redundant pixel operations and, we found, further improves robustness.

The resulting shadow distortion map contains six floating point values: a two-component image-space coordinate for the nearest point on the silhouette edge, a two-component silhouette edge surface normal, and two values specifying the magnitude of the discontinuity (z_{near} and z_{far}). The value d_f stored in the DDOC distortion map is unnecessary, as the maximal distortion in the SSOC varies based upon light radius, z_{near}, and z_{far}.

3.2.2 Determining Intensity from the Distortion Map

After creating a distortion map, a per-fragment shader indexes into the map to determine shadow intensity. A naive approach for this shader is outlined in Figure 5. For points far from a shadow boundary, we use standard shadow mapping to quickly identify points inside and outside the shadow. This restricts the distortion process described below to points inside (or very near) the penumbral regions.

The actual distortion aims to solve the problem with techniques such as smoothies and penumbra mapping, namely that they cannot render inner penumbrae due to a lack of information about which silhouette edges partially occlude the light. Correctly computing the per-fragment distortion specified by the SSOC implicitly computes the information needed for soft shadows—the distance to the edge occluding the light. Given the distortion $d(\mathbf{p}_z)$ at point $\mathbf{p}$ found by indexing into the distortion map, the shadow intensity is simply computed as follows:



Near discontinuity?
Yes
Distort, according to distortion map
No
Use shadow map
Distorted point still in hard shadow?
Yes
In complete shadow
No
Distorted point still near discontinuity?
Yes
Compute intensity based on distortion magnitude
No
Completely lit

Figure 5. A naive approach to computing soft shadow intensity from a distortion map.

$$S_{intensity} = 1 - \frac{d(\mathbf{p}_z)}{d_{max}(\mathbf{p}_z)}. \tag{1}$$

To intuitively understand the equation, consider the extremal cases shown in Figure 4. Pixels near the inner edge of the penumbra, such as **A**, distort the full d_{max}. These pixels should be fully shadowed, and hence have an intensity of zero. Pixels at the outer edge of the distortion region correspond to those at the outer edge of the penumbra, distort not at all, and should be fully lit. As in many previous soft shadow approximations [7,8,14,27,36], we modify this linear gradient by using the Bernstein interpolant $s = 3t^2 - 2t^3$ to approximate the sinusoidal falloff of a spherical light source.

The two remaining cases in Figure 5 describe boundary cases for incorrectly classified pixels. The SSOC distortion map contains a conservative approximation of the penumbral region, thus a few pixels distorted by the map may actually be fully illuminated or fully shadowed. Distorted points that are visible but lie too far from the discontinuity were mistakenly classified as penumbral, and are actually fully illuminated (for us, this arises due to numerical errors). Pixels that never distort enough to be visible from the light will be fully shadowed. This mainly arises during self shadowing (see Figure 6), and it prevents light leaks when points occluded by multiple surfaces are mistakenly

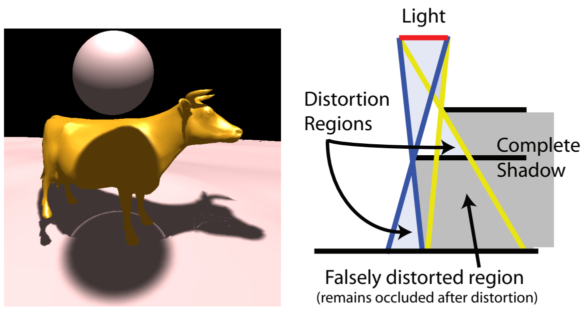


Figure 6. Light leaks, such as those shown under the cow, are eliminated by checking if distorted points are visible in the shadow map. On the right, a 2D example where points in the yellow distortion region remain occluded after distortion, signifying they are completely shadowed.

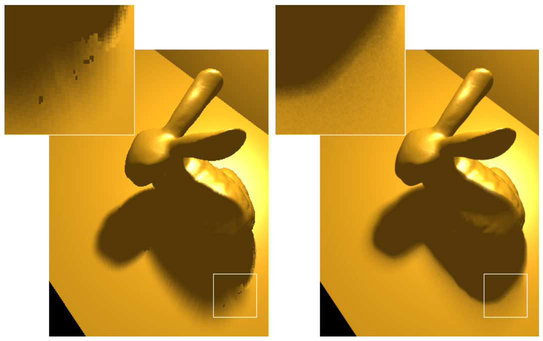


Figure 7. Using the naive approach from Section 3.2.2 leads to artifacts when multiple edge discontinuities affect the same pixels in the distortion map. Depending on which edge's data is stored inaccurate distortion can occur, resulting in inappropriately dark or light pixels. On the right is a ray traced image for comparison.

distorted by the one closest to the light. In these cases, the distorted point still lies in the hard shadow and thus is not visible from the light.

3.2.3 Improving Intensity Determination

Generally the naive approach works well, except where edge quads overlap. The problems visible in Figure 7 exemplify the issues encountered in such cases. These artifacts typically appear as dark spots in the middle of penumbra or as overly bright regions where penumbrae overlap. This arises from indexing into an incorrect edge quad, leading to inaccurate distortion and intensity. If Figure 7, the dark errors occur where distortion from the bunny's back (instead of its ear) is used to compute intensity.

Commonly, other researchers eliminate problems due to adjacent discontinuities by storing additional information to identify the correct penumbral region. This often involves storing multiple layers [1, 7, 32] or explicitly storing geometric representations of the penumbrae (e.g., penumbra wedges [3,4]). We observe, however, that the distortion map itself generally stores enough information to identify the correct penumbra. Instead of storing overlapping penumbral data in layers or with explicit geometry, a distortion map stores this data in different texels. The key is identifying which texels.

When penumbrae overlap the distortion map only stores one or the other, leaving the distortion regions truncated (see Figure 8). After distortion a pixel may thus project into a different distortion quad, which represents another of the overlapping penumbrae. We propose recomputing the original fragment's distortion according to the new silhouette edge. Each time we recompute the distortion, we may find another overlapping penumbra. Using these successive, independent distortion map lookups, we may find:

1. Only one penumbra contains the current fragment,
2. Multiple penumbrae overlap, but only one is relevant (e.g., one penumbra is completely inside another), or
3. Multiple penumbrae affect the fragment's intensity.

In either of the first two cases, Equation 1 describes the shadow intensity, as only one of the overlapping penumbrae affects the fragment. In the third case, multiple surfaces occlude different portions of the light and must be accounted for independently. While there are multiple ways to approximate this combination, we used a multiplicative combination of the penumbral intensities in our prototype. In other words, we independently evaluate Equation 1 for each penumbra and multiply the results. We found two or three such steps are generally sufficient, and we implement them together in a single pass.

4 Implementation Details

We implemented our prototype in OpenGL with vertex and fragment shaders written in Cg. This required a number of significant changes from the original DDOC implementation, which ran as a batch CPU process. Section 3.2.1 outlined the major changes, including the removal of the image-space edge detector and normal computations. However, a number of other implementation details affected the design of our prototype.

One important decision we made was to use two separate distortion maps, one for inside and one for outside the hard shadow boundaries. Initially, this allowed us to better

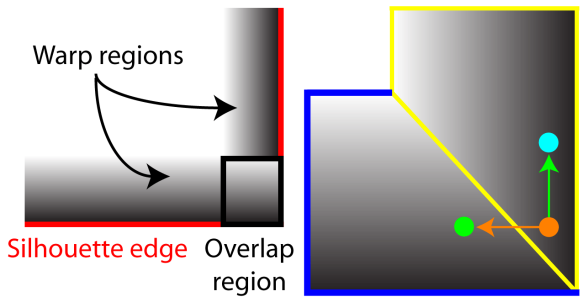


Figure 8. When distortion regions from adjacent silhouette edges overlap (left), only one may be stored (right). Thus, distortion may move a fragment into a new region. In such cases, we know multiple silhouette edges affect the fragment, so we perform additional distortions based upon newly found silhouette edges. Combining the results gives our final intensity.

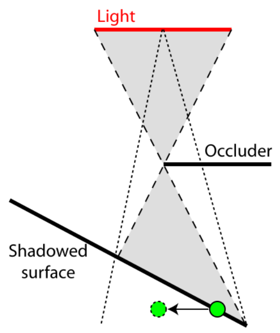


Figure 9. False shadows may occur when occluded surfaces vary quickly in z. Here, the green point is warped under the occluded surface. Following Figure 5, this point mistakenly falls in the umbra. We fix this with a bias, though constructing a SSOC reference image would avoid the problem, as the entire occluded surface would be warped to the left.

debug the algorithm and identify incorrectly warped fragments. However, this choice also reduces overlaps between distortion regions, particularly for objects that exhibit self shadowing, and may thus be important for any implementation.

All shadow mapping algorithm exhibit numerical precision issues; adding a small bias during comparisons typically solves this problem. Our algorithm exacerbates these issues, simply due to the increased number of depth comparisons. The most severe problem arises because the algorithm does not explicitly consider the geometry of occluded surfaces (see Figure 9). We fix this by adding a rather large bias, though other solutions are possible. Note that algorithms using DDOC reference images would avoid the problem, as the camera would distort both surfaces in the reference.

Section 3.2.3 discusses how we combine contributions from overlapping penumbrae. After warping a fragment, our implementation determines which of the three cases occurred by comparing the silhouette positions and normals, as well as the depth range $[z_{near} \ldots z_{far}]$. If pre- and post-warped fragments belong to the same distortion region, a simple dot product between the two stored normals returns a value near unity. If multiple penumbrae collide in the distortion map but have non-intersecting $[z_{near} \ldots z_{far}]$ ranges, only one penumbra is likely to affect the current fragment. In the third case, where multiple penumbrae interact, the normals, silhouette positions, and depth ranges will be markedly different.

5 Results

Our OpenGL prototype was benchmarked at a resolution of 512^2 on a 3.2 GHz Pentium 4 Xeon with 2 GB of memory and a GeForce 8800 GTX. The timings shown in Table 1 show the costs involved with various stages of our prototype. Note that our implementation lacks optimizations, providing significant potential for speedups. In particular, due to issues with texture interpolation in GL_RGBA_FLOAT32_ATI buffers used as vertex textures, we independently compute shadow intensity four times per fragment and linearly interpolate, to eliminate aliasing from nearest-neighbor sampling.

Table 1 shows that costs for silhouette extraction and SSOC creation vary roughly linearly with the complexity of the scene. As we explicitly check each edge every frame to determine if it appears as a silhouette when viewed from the light, this process becomes the bottleneck for more complex models. Using more efficient approaches [21] or identifying these edges using the GPU's geometry processor should improve performance.

On the other hand, the cost to render from the eye varies mainly based upon the number of pixels covered by penumbrae, not by scene complexity. This is a significant advantage of image-based techniques such as our SSOC model, which is essentially a compact image-based approximation of the scene. In regions far from a penumbra, we can rely on a cheap shadow map to control shadows, and only for regions near penumbrae must we rely on the distortion map.

Figure 10 shows the effect of varying the size of the spherical light source from a radius 0.0 to 0.1 and 0.3. Note, in particular, the reduced size of the umbra as the light

Scene (and triangle count)	512^2					1024^2
	Silhouette Extraction	Create Shadow Map	Create SSOC	Render from Eye	Framerate	Framerate
Bunny (70k)	8.0 ms	0.8 ms	1.7 ms	8.2 ms	85.4 fps	41.6 fps
Cow and Sphere (26k)	2.6 ms	0.5 ms	0.6 ms	5.0 ms	135.1 fps	68.9 fps
Dragon and Sphere (270k)	27.7 ms	1.8 ms	5.3 ms	16.5 ms	29.6 fps	24.5 fps
F-16 (4.5k)	0.4 ms	0.4 ms	0.4 ms	4.7 ms	180.2 fps	58.6 fps
Two Teapots (13k)	1.2 ms	0.4 ms	0.5 ms	4.9 ms	186.6 fps	76.5 fps
Venus and Sphere (64k)	6.2 ms	0.6 ms	1.4 ms	7.7 ms	84.7 fps	39.6 fps

Table 1. Computation costs for the scenes shown in Figures 1, 10, 11, and 12. We have timed individual steps of the algorithm at 512^2 resolution, and final framerates are given for both 512^2 and 1024^2. Due to overhead for timing individual steps, summing step costs does not exactly equal the stated framerate.

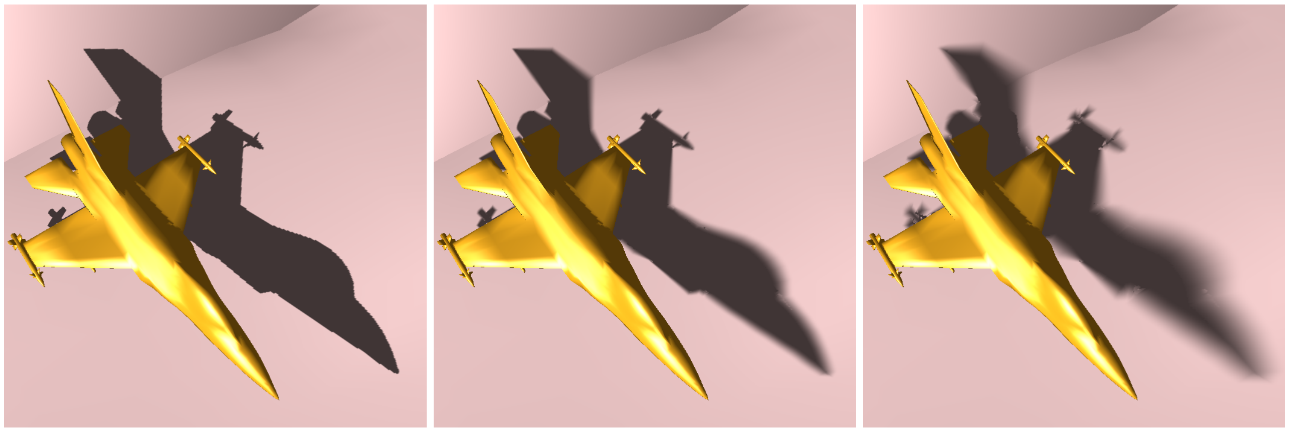


Figure 10. An F-16 model rendered using a point light and two sources of increasing area.

increases in size. Varying the penumbra size does affect runtime costs, as larger penumbrae require more fragments to index into the distortion map. For the penumbra sizes shown in Figure 10, the framerate varies from 175 to 185 frames per second at 512^2. Figures 1 and 11 focus on more complex examples involving self shadowing and multiple occluders. Note that the SSOC camera model correctly handles the sharpening of shadows near contact points as well as overlapping penumbrae of various sizes.

6 Discussion

While the soft shadows generated using the SSOC model are quite plausible, there are a couple of limitations that should be mentioned. These issues arise in situations where the SSOC distortion map becomes heavily populated. In such cases we may miss overlapping penumbra, because we only store one distortion value per texel and search for alternate distortions using a small number of point queries (as per Section 3.2.3). More exhaustive object-space [11] and image-space [13] searches avoid these problems.

In particular, the DDOC and SSOC models provide an

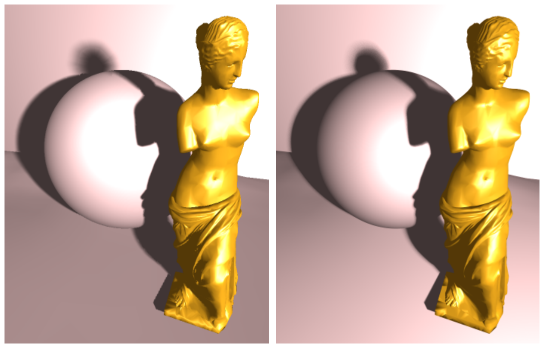


Figure 12. A comparison of our approach (left) with a ray traced reference (right).

approximation to the scene geometry in a small locus of views around a reference viewpoint. As the size of this locus increases, typically the errors do as well. This means a single SSOC image is only effective for relatively small lights, such as those shown in Figure 10. Objects with many small concavities (e.g., a fork) also lead to heavily popu-

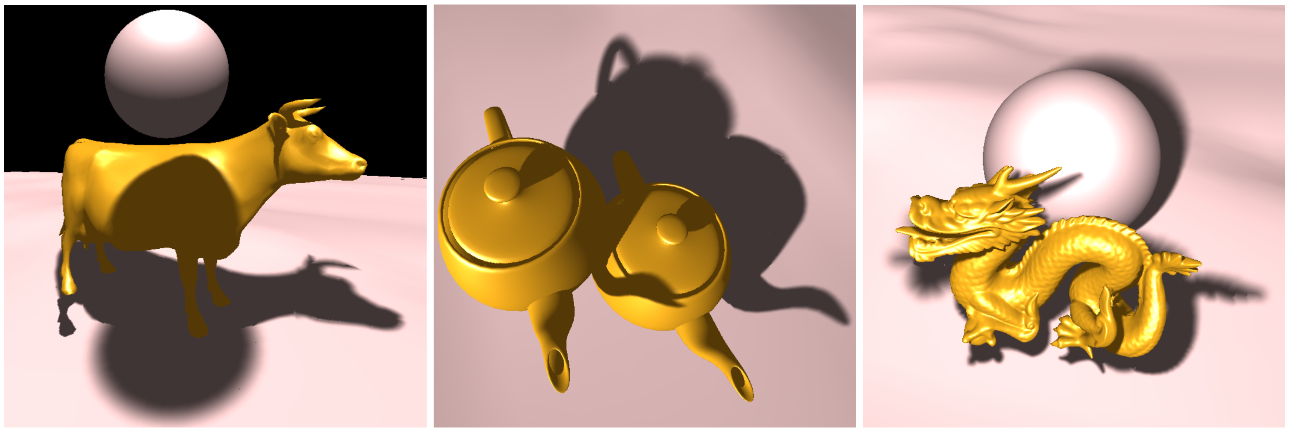


Figure 11. Soft shadows on objects of varying complexity. In particular, note the self shadowing and multi-layered shadows.

lated distortion maps, as multiple penumbrae interact in a small region. Finally, scenes with high depth complexity, as seen from the light, increase the complexity of the distortion map. Unlike many methods, our technique is not limited by depth complexity per se. Rather, the SSOC has difficulty when multiple edges collide in the distortion map. Effectively, only two or three silhouette edges can overlap in the distortion map before artifacts start to appear.

For cases of large lights, small object concavities, and high depth complexity, such a highly populated distortion map leads to incorrectly shaded regions and discontinuities between correctly and incorrectly shaded regions, as shown in Figure 13. A number of solutions may alleviate the problem, including performing a more expensive search for overlapping penumbral regions in the distortion map. A hierarchical distortion map may help reduce the overhead of a more extensive search. Another possibility would utilize a multiple layered distortion map, though this defeats the purpose of the depth discontinuity occlusion camera model.

Another issue we ignored for our prototype was the shape of the light source—we assume it is spherical. While this is not an inherent limitation of the SSOC, as the SSOC is valid for any shaped locus of points around the center of the light, it allowed our prototype to use a simple intensity determination. We hope to address this limitation in future work, so that the SSOC can be used with varied shaped lights and those with non-constant emission.

7 Conclusions and Future Work

We introduced the soft shadow occlusion camera, a non-pinhole camera model inspired by the depth discontinuity occlusion camera that samples geometry normally hidden by occluders in standard shadow maps. While reconstructing a warped shadow map interactively is currently infeasible, we have shown that the camera model may be directly used to approximate soft shadow at interactive rates. This is accomplished by correlating the camera's distortion with a sample's location in the penumbra.

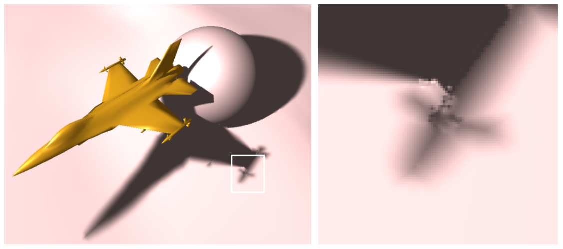


Figure 13. Small concavities can cause a highly populated distortion map, which makes identifying all relevant penumbral regions more difficult. This leads to incorrectly shaded pixels and discontinuities between correctly and incorrectly shadowed regions.

In addition to the contribution to shadow rendering, we hope that our work shows the applicability of non-pinhole cameras to interactive rendering problems, stimulates the development of interactive techniques for rendering with non-traditional camera models, and encourages further research using such cameras for realistic rendering. In particular, future work could examine alternate ways for improving and utilizing the SSOC model, such as developing an interactive implementation to reconstruct a reference image (instead of relying only on the distortion map). We also believe the SSOC model may prove useful for other soft shadow techniques, including backprojection schemes. Finally, these camera models seem like a natural fit for other realistic rendering problems such as depth-of-field, motion blur, and glossy reflections, which all need information about fragments nearly visible in a static pinhole camera image.

References

[1] M. Agrawala, R. Ramamoorthi, A. Heirich, and L. Moll. Efficient image-based methods for rendering soft shadows. In *Proceedings of SIGGRAPH*, pages 375–384, 2000.

[2] T. Aila and T. Akenine-Moller. A hierarchical shadow volume algorithm. In *Proceedings of the Graphics Hardware*, pages 15–23, 2004.

[3] T. Akenine-Möller and U. Assarsson. Approximate soft shadows on arbitrary surfaces using penumbra wedges. In *Proceedings of the Eurographics Rendering Workshop*, pages 309–318, 2002.

[4] U. Assarsson and T. Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics*, 22(3):511–520, July 2003.

[5] L. Atty, N. Holzschuch, M. Lapierre, J.-M. Masenfratz, C. Hansen, and F. Sillion. Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum*, 25(4):725–741, 2006.

[6] L. Bavoil, S. Callahan, and C. Silva. Robust soft shadow mapping with depth peeling. Technical Report UUSCI-2006-028, University of Utah, 2006.

[7] X.-H. Cai, Y.-T. Jia, X. Wang, S.-M. Hu, and R. Martin. Rendering soft shadows using multilayered shadow fins. *Computer Graphics Forum*, 25(1):15–28, 2006.

[8] E. Chan and F. Durand. Rendering fake soft shadows with smoothies. In *Proceedings of the Eurographics Symposium on Rendering*, pages 208–218, 2003.

[9] E. Chan and F. Durand. An efficient hybrid shadow rendering algorithm. In *Proceedings of the Eurographics Symposium on Rendering*, pages 185–196, 2004.

[10] F. Crow. Shadow algorithms for computer graphics. In *Proceedings of SIGGRAPH*, pages 242–248, 1977.

[11] G. Drettakis and E. Fiume. A fast shadow algorithm for area light sources using backprojection. In *Proceedings of SIGGRAPH*, pages 223–230, 1994.

[12] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH*, pages 43–54, 1996.

[13] G. Guennebaud, L. Barthe, and M. Paulin. Real-time soft shadow mapping by backprojection. In *Proceedings of the Eurographics Symposium on Rendering*, 2006.

[14] E. Haines. Soft planar shadows using plateaus. *Journal of Graphics Tools*, 6(1):19–27, 2001.

[15] J.-M. HasenFratz, M. Lapierre, N. Holzschuch, and F. Sillion. A survey of real-time soft shadow algorithms. *Computer Graphics Forum*, 22(4):753–774, 2003.

[16] P. Heckbert and M. Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, Carnegie Mellon University, January 1997.

[17] T. Heidmann. Real shadows, real time. *Iris Universe*, (18):23–31, November 1991.

[18] W. Heidrich, S. Brabec, and H.-P. Seidel. Soft shadow maps for linear lights. In *Proceedings of the Eurographics Rendering Workshop*, pages 269–280, 2000.

[19] H. Hu, A. Gooch, W. Thompson, B. Smits, J. Rieser, and P. Shirley. Visual cues for imminent object contact in realistic virtual environments. In *Proceedings of Visualization*, pages 127–136, 2000.

[20] Y.-H. Im, C.-Y. Han, and L.-S. Kim. A method to generate soft shadows using a layered depth image and warping. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):265–272, May/June 2005.

[21] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A Developer's Guide to Silhouette Algorithms for Polygonal Models. *IEEE Computer Graphics and Applications*, 23(4):28–37, 2003.

[22] D. Kersten, D. C. Knill, P. Mamassian, and I. Bulthoff. Illusory motion from shadows. *Nature*, 279(6560):31, 1996.

[23] S. Laine. Splat-plane shadow volumes. In *Proceedings of Graphics Hardware*, pages 23–32, 2005.

[24] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH*, pages 31–41, 1996.

[25] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *Proceedings of SIGGRAPH*, pages 39–46, 1995.

[26] C. Mei, V. Popescu, and E. Sacks. The occlusion camera. *Computer Graphics Forum*, 24(3):335–342, 2005.

[27] S. Parker, P. Shirley, and B. Smits. Single sample soft shadows. Technical Report UUCS-98-019, University of Utah, October 1998.

[28] V. Popescu and D. Aliaga. The depth discontinuity occlusion camera. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 139–143, 2006.

[29] P. Rademacher and G. Bishop. Multiple-center-of-projection images. In *Proceedings of SIGGRAPH*, pages 199–206, 1998.

[30] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. In *Proceedings of SIGGRAPH*, pages 249–252, 1992.

[31] J. Shade, S. Gortler, L. wei He, and R. Szeliski. Layered depth images. In *Proceedings of SIGGRAPH*, pages 231–242, 1998.

[32] J.-F. St-Amour, E. Paquette, and P. Poulin. Soft shadows from extended light sources with penumbra deep shadow maps. In *Proceedings of Graphics Interface*, pages 105–112, 2005.

[33] L. Wanger, J. Ferwerda, and D. Greenberg. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics & Applications*, 12(3):44–58, May 1992.

[34] L. Williams. Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH*, pages 270–274, 1978.

[35] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Computer Graphics & Applications*, 10(6):13–32, November 1990.

[36] C. Wyman and C. Hansen. Penumbra maps: Approximate soft shadows in real-time. In *Proceedings of the Eurographics Symposium on Rendering*, pages 202–207, June 2003.