Fast Calculation of Soft Shadow Textures Using Convolution

Cyril Soler and François X. Sillion

*i*MAGIS – GRAVIR/IMAG

Abstract

The calculation of detailed shadows remains one of the most difficult challenges in computer graphics, especially in the case of extended (linear or area) light sources. This paper introduces a new tool for the calculation of shadows cast by extended light sources. Exact shadows are computed in some constrained configurations by using a convolution technique, yielding a fast and accurate solution. Approximate shadows can be computed for general configurations by applying the convolution to a representative "ideal" configuration. We analyze the various sources of approximation in the process and derive a hierarchical, error-driven algorithm for fast shadow calculation in arbitrary configurations using a hierarchy of object clusters. The convolution is performed on images rendered in an offscreen buffer and produces a *shadow map* used as a texture to modulate the unoccluded illumination. Light sources can have any 3D shape as well as arbitrary emission characteristics, while shadow maps can be applied to groups of objects at once. The method can be employed in a hierarchical radiosity system, or directly as a shadowing technique. We demonstrate results for various scenes, showing that soft shadows can be generated at interactive rates for dynamic environments.
Keywords: Soft shadows, Convolution, Shadow map, Error-Driven illumination, Texture.

1 Introduction

The computation of *soft shadows*, i.e. shadows cast by extended light sources, is one of the most difficult challenges in rendering for computer graphics. Soft shadows are a result of the continuous variation of illumination across a receiving surface, when the light source becomes partially occluded by other objects in the scene. Their appearance is mainly controlled by the shape and location of *penumbra* regions, which are the regions on a receiver where the light source is partially visible.

Soft shadows play a key role in the overall realism of computer-generated images, because they provide important visual cues about the 3D arrangement of objects [28]. The location of cast shadows with respect to the blocking objects informs the viewer about the main directions of illumination, and the sharpness of the penumbra helps understand the distance relationships between the source, blocker and receiver.

Unfortunately, the calculation of soft shadows is also very difficult. It can be restated as an area *visibility determination* problem, since the goal is to identify the regions of partial source visibility, as well as quantifying the relative area of the source that is visible. There are many methods for computing hard shadows (from point light sources), including some texture-based algorithms that can run in real-time on graphics computers. However, the two main avenues for the treatment of extended light sources each have severely limiting problems. Analytic techniques such as discontinuity meshing suffer from excessive time and memory costs, and numerical robustness problems, while sampling techniques are prone to annoying image artifacts unless they are pushed to a stage where they become too expensive.

In this paper, we present a new method for the calculation of soft shadows, which is able to provide pleasant, artifact-free images in a very efficient way. The method is based on the calculation of *shadow maps*, which are textures created from images of the light sources and occluders using a convolution technique. The convolution is performed with images of the light source and the set of occluders, rendered in offscreen buffers. The shadow textures are then used to modulate direct light source illumination across the receiving objects. Exact images are obtained for some specific cases (parallel polygons), while for general configurations some approximation is necessary. We analyze the error incurred and the various sources of approximation, and show how the overall approximation can be controlled using a spatial hierarchy of object clusters. This is achieved by combining shadow maps of the sub-clusters hierarchically.

The resulting error-driven algorithm automatically computes soft shadows at interactive rates for extended light sources of arbitrary shape and exitance distribution, while avoiding excessive approximation under a feature-based error metric. The method can be used in any rendering technique, with the only requirement of a hierarchy of spatial clusters in order to use the hierarchical combination. The algorithm is naturally adaptive and eliminates the difficulties associated with light source sampling. The error-driven hierarchical combination of shadow maps lets us adapt the effort to user-specified approximation tolerances.

Because it uses a single rendered image of the blocker to generate the soft shadows, our technique effectively trades graphics performance for raw computing performance in the form of FFT calculations and image manipulation, which makes it interesting for computers with low or mid-range graphics capabilities.

The remainder of this paper is organized as follows: Section 2 reviews previous approaches to shadow generation and discusses our goals; Section 3 explains the basic convolution method for computing shadow maps, and Section 4 extends the technique to general source and receiver configurations. A number of implementation choices and details are presented in Section 5. Section 6 then discusses the different sources of error and presents our error-driven hierarchical combination method for shadow maps. We present results obtained in a variety of configurations in Section 7, discuss the merits of the approach in Section 8 and conclude in Section 9.

*i*MAGIS is a joint research project of CNRS, INRIA, INPG and UJF. Postal address: *i*MAGIS/IMAG, B.P. 53, 38041 Grenoble Cedex 9, France. Email: `[Cyril.Soler|Francois.Sillion]@imag.fr`.
Web: `http://www-imagis.imag.fr/~Cyril.Soler`

2 Previous work

Woo *et al.* present an excellent survey of the vast literature on shadow algorithms [31], and we will only briefly review here some of the main approaches, especially the few that allow the computation of soft shadows.

Sampling methods

Ray tracing algorithms compute shadows by casting a ray between a point lying on a surface, and a designated light source [29]. This blends very nicely with the rest of the ray tracing technique, but is quite expensive since each ray must in effect sample the scene for potential occluders. Soft shadows are generated in *distributed ray tracing* by casting several rays towards a set of sample points on an extended light source [5].

Another sampling option is to create a depth image from the point of view of the light source [30]. This *shadow buffer* can be used to check whether a given point, visible in the final image, is visible from the source. The severe aliasing issues experienced in a naive approach can be treated using elaborate depth filtering [21]. This approach uses a single point on the light source and therefore can not render penumbrae due to extended light sources.

Using auxiliary data structures

To avoid the cost of brute force sampling, a specific data structure can be created that represents the visibility relationships in the scene. Such structures vary widely in complexity and cost, and essentially allow a time gain because they let us benefit from the *coherence* of visibility in space.

Shadow volumes are constructed relatively easily from a point light source and polygonal occluders [6], and visible points can be quickly tested for inclusion in object space when rendering an image. Complex volumes can be represented and used efficiently through the use of BSP trees [3]. Approximate soft shadows are obtained by combining several shadow volumes, each corresponding to a sample point on the extended source [2].

A better structure for extended light sources records visibility information on the surfaces of the scene, in the form of a *discontinuity mesh* that includes all lines where the illumination function has discontinuities of various orders. Unless an occluder touches a receiver, the illumination function from an extended source is continuous, and exhibits discontinuities only in its derivatives. Techniques for computing discontinuity meshes operate by considering all possible *visual events* and inserting critical lines in an explicit mesh structure [11, 15], which makes them both quite expensive to use and subject to robustness issues. However, they can provide exact visibility [8] and produce images of the highest quality.

Interactive shadow generation

Shadows can be generated while displaying the scene, using one or more extra rendering passes. This is of course especially interesting when hardware acceleration is available to perform the various passes and combine the results. For instance, shadow maps from point or directional light sources can be created by the rendering pipeline and applied using texture mapping operations [23]. Shadow volumes can be combined using the stencil buffers [7].

The only method we are aware of for pre-calculating soft shadows at interactive rates is Heckbert and Herf's [13], where a number of shadow images are created, registered and averaged on the receiver. Each image corresponds to a sample point on the light source, and they are all combined using the accumulation buffer [9]. This method works very well on high-end graphics machine, but essentially produces a superposition of "hard" shadows. The shadows cast by the individual samples are usually noticeable unless a very large number of samples is used.

Shadows and illumination techniques

The radiosity method is often credited with a unique ability to render subtle effects such as soft shadows and illumination details. Radiosity techniques are based on a surface mesh used to compute, store and reconstruct illumination functions. The exchange of energy between mesh elements is evaluated using *form factors* to represent the effect of orientation, geometric attenuation and visibility. Most modern form factor calculation methods actually decouple the estimation of an "unoccluded" form factor based on the radiosity kernel, from that of a *visibility factor* expressing the fraction of the area of the source element that is visible from the receiver [4, 27, 10].

Zatz [32] pushes this idea further by proposing the separate calculation of sampled *shadow masks* to represent the effects of visibility in a separate step. Several authors observed that in the case of ideal diffuse scenes, the entire illumination can be recorded into *radiosity textures*. Such textures can be precomputed off-line, then allowing high-quality rendering with soft shadows at interactive rates [12, 18]. Keller's "instant radiosity" technique [14] computes radiosity textures in a manner similar to Heckbert's, by averaging shadow images from point samples chosen on the surfaces.

In order to simulate the complex shadows due to sunlight and skylight under tree canopies (as shown in the "Sun and Shade" movie [16]), Max used the convolution of a radiance image of the sky with a transparency mask of the canopy [17].

Multiresolution shadows

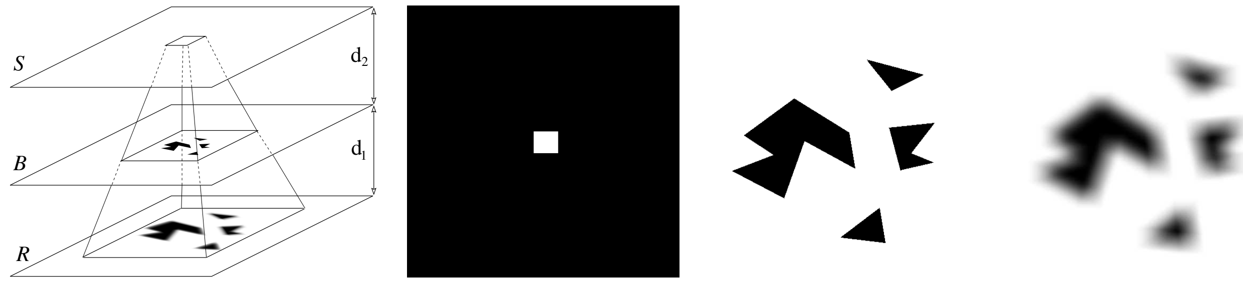
Experimental evidence suggests that while shadows are important in a 3D rendering, they need not necessarily be exact [28]: this is well known by drawing artists who often sketch an approximate shadow with "appropriate" characteristics to increase realism. This idea was applied to the calculation of multi-resolution visibility factors in the context of hierarchical radiosity and clustering [24]. For a given source/receiver pair, an "appropriate" level in the hierarchical representation of the occluders is selected, and used to create an approximate shadow based on an analogy with semi-transparent volumes. This work effectively produces shadows of variable resolution and cost, but does not provide bounds on the error incurred. Such bounds can be computed in a simplified 2d case [26] but appear very difficult to compute in 3D, mainly because the identification of a cluster to a semi-transparent object is too crude.

Discussion

Our goal is to provide a shadowing algorithm running at interactive rates, in a manner similar to Heckbert's. However, we want to avoid the sampling artifacts produced when averaging hard shadows, without having to resort to very large numbers of samples (more than 100 can be necessary for large sources [20]). On the other hand, we do not want to build expensive data structures to represent visibility, but rather to compute necessary information on the fly. The convolution algorithm explained below can be seen as an extension of Max's method [17], and meets these goals by always providing a smooth image in soft shadow regions.

3 Obtaining soft shadows with convolution

In this section we present the basis of our technique in the form of an algorithm for producing a shadow map across a given receiver, subject to the illumination of an extended light source and to shadows cast by a set of objects. We first explain how the shadow can be expressed as a convolution operation, in the special case of parallel objects, and then propose an extension to the general case.



(a) Parallel configuration (b) Source image (c) Blocker image (d) Convolution

Figure 1: A simple case of parallel light source (S), occluder (B) and receiver (R). The source image is convolved with the blocker image to obtain the shadow map.

3.1 Convolution formula for a set of parallel objects

Let us first consider the special case where the light source, the receiver and the occluder are all planar, and lie in parallel planes (Figure 1). The irradiance [25] at a point y on the receiver is:

$$H(y) = \frac{E}{\pi} \int_S \frac{\cos(\theta(x,y))\cos(\theta'(x,y))}{d(x,y)^2} v(x,y)dx$$

where E is the exitance [25] of the source, $d(x,y)$ the distance between x and y, θ and θ' the incident angles of the ray $x \longrightarrow y$ on the source and the receiver, and $v(x,y)$ a binary *visibility function* indicating whether x and y are mutually visible.

A common approximation, e.g. in radiosity algorithms, consists of separating the visibility factor in a distinct integral to obtain:

$$H(y) \approx E \underbrace{\int_S \frac{\cos\theta\cos\theta'}{\pi d^2} dx}_{F_S(y)} \underbrace{\int_S v(x,y)dx}_{V(y)} \qquad (1)$$

The first term $F_S(y)$ is the unoccluded *point-to-polygon* form factor from y to the source, and the second term $V(y)$ is the visible area of the source as seen from y. This approximation implicitly assumes a low correlation between the variations of visibility and the radiosity kernel, an assumption that is reasonable in most cases. In this paper we are focusing on the calculation of $V(y)$. The unoccluded form factor can be computed using integration formulae [22] or approximated using the hardware shading model [13].

Computing $V(y)$ is equivalent to projecting the blocker onto the source from y and measuring the remaining unoccluded area of the source. In the present case, because all three components are parallel, the projection of the blocker simply translates on the source as y moves on the receiver. This is precisely why the unoccluded area of the source can be expressed as a convolution between the source and blocker images.

More formally, let us now introduce the following *characteristic* functions of the source and blocker in their respective planes:

$$S(x) = \begin{cases} 1 & \text{if } x \text{ is on the source} \\ 0 & \text{elsewhere} \end{cases}$$
$$P(x) = \begin{cases} 0 & \text{if } x \text{ is on the occluder} \\ 1 & \text{elsewhere} \end{cases}$$

We can use P to express the binary visibility value between two points x and y, by introducing the point of intersection of the xy line and the plane of the blocker. The visibility factor can then be written as

$$V(y) = \int_S P\left(\frac{d_1 x + d_2 y}{d_1 + d_2}\right) dx$$

To show that this expression is a convolution, let us transform $V(y)$ by extending the integration over the entire plane:

$$\begin{aligned} V(y) &= \int_{\mathbb{R}^2} S(x) P\left(\frac{d_1 x + d_2 y}{d_1 + d_2}\right) dx \\ &= \left(\frac{d_2}{d_1}\right)^2 \int_{\mathbb{R}^2} s_\alpha(-t) p_{1+\alpha}(t+y) dt \\ &= \frac{1}{\alpha^2} (s_\alpha \star p_{1+\alpha})(y) \end{aligned}$$

where

$$\alpha = \frac{d_1}{d_2} \qquad s_\alpha(x) = S(-\frac{1}{\alpha}x) \qquad p_{1+\alpha}(x) = P(\frac{1}{1+\alpha}x) \qquad (2)$$

and $\star$ denotes a convolution operation. Therefore, in this particular geometric configuration, the visibility factor reduces to the convolution of the scaled characteristic functions of the source and blocker.

Note that this particular form of the visibility term implies that it is continuous, therefore implicitly creating soft shadow variations on the receiver. An example of convolution between source and blocker images is presented in Figure 1. For a diffuse surface, we can express the radiosity function B on the receiver by introducing the diffuse reflectance $\rho(y)$ and using Eq. (1):

$$B(y) \approx \frac{\rho(y)E}{\alpha^2} F_S(y) (s_\alpha \star p_{1+\alpha})(y) \qquad (3)$$

Therefore, a possible algorithm for displaying soft shadows is to compute a shadow map using the convolution formula, and use it as a texture to modulate the illumination function $\rho(y)EF_S(y)$ across the receiver.

3.2 Computation of soft shadows in general configurations

We will see in Section 5 that Equation (3) can be used in an efficient algorithm to create illumination textures. But its value is of course severely limited by the assumption that all objects are planar and parallel. In real applications, not only can light sources and receivers be placed at arbitrary orientations, but occluders can also in general occupy a complex volume in 3D.

In a general source/blocker/receiver configuration, it is not possible to derive a convolution formula similar to Eq.(3). Nevertheless, we propose to approximate the resulting shadow effect by using the convolution method for a *virtual geometry* that obeys the preceding requirements, and transform the associated result to fit the actual geometry of the scene. This involves the following operations:

a) choosing a direction $\mathcal{D}$ and a set of three planes containing respectively a virtual source S_v, a virtual blocker B_v and a virtual receiver R_v, all planar and orthogonal to $\mathcal{D}$;

b) computing the illumination function on the virtual receiver using the convolution formula (3);

c) projecting the result back on the actual receiver.

Clearly, depending on the actual geometry of the scene, such an approximation may produce some artifacts. The different sources of error and the way to control them are addressed in Section 6. We now discuss each of these steps in more detail.

3.2.1 Choice of the virtual geometry

The choice of the direction of projection $\mathcal{D}$ is the first issue to be addressed. Obviously, the nature and importance of the approximation will largely depend on this choice. We will discuss this question in more detail in Section 5.1. For now, we observe that it seems natural to have $\mathcal{D}$ be some average of the directions actually involved in the transfer of energy. Therefore we suggest as a possibility to choose $\mathcal{D}$ to be the mean direction of all possible rays between the source and the receiver (Figure 2.(a)).

Once $\mathcal{D}$ has been chosen, it defines the orientation of the three virtual planes. Let us denote by Z an axis parallel to $\mathcal{D}$. Then, we choose altitude values z_s,z_b and z_r for the three virtual planes (Figure 2.(b)). The choice of these values is discussed in Section 5.2.

Each component is now projected onto its virtual plane: The virtual source is obtained from the source by orthographic projection along $\mathcal{D}$. Thus, viewed from the blocker, this virtual source has nearly the same aspect as the original one (See Figure 2.(c)).

The virtual blocker is the projection of the original blocker on the virtual blocker plane. The projection used is a perspective projection, with eye set to the center of the source (See Figure 2.c). Using the same projection onto the virtual receiver plane, we obtain the virtual receiver from the original one. Viewed from the center of the source, the original and virtual blockers (resp. receivers) are thus identical.

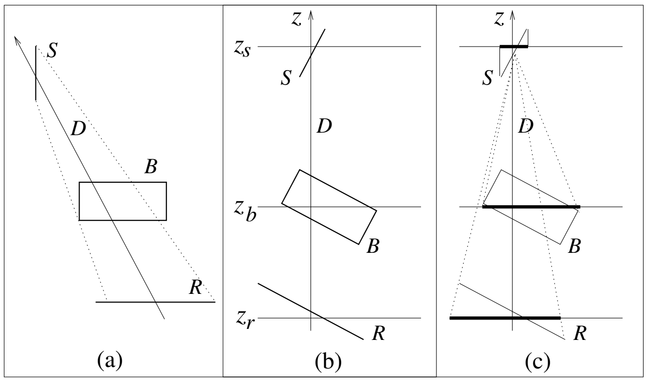


Figure 2: Construction of a virtual source, blocker and receiver for a general shadow configuration. (a) Choosing a preferred direction. (b) Choosing altitudes for the virtual planes. (c) Projecting the original elements to obtain their virtual counterparts.

3.2.2 Back to the actual geometry

Once computed for the virtual receiver, the visibility term $V(y)$ is projected back to the actual receiver where it is multiplied by the direct illumination factor from the source $\rho E F_S(y)$. In practice, the convolution image is set as a shadow texture and modulated by direct illumination values for uniformly sampled points on the receiver.

4 Extensions to the basic principle

In this section we show that our convolution method can be adapted with little modification to even more general lighting conditions. Non-uniform light sources, sources with complex 3D shapes, and complex receiver shapes can all be simulated. Practical examples for each of the three cases are presented in Section 7. In particular, we show that groups of objects can be used to model sources or receivers in a single shadow map calculation.

4.1 Dealing with non-uniform radiosity over light sources

When the source is not uniform, but still planar, we can modify the derivation of Section 3, replacing the uniform exitance term E by a non-uniform exitance $E(x)$:

$$\begin{aligned} B(y) &= \frac{\rho}{\pi}\int_S \frac{\cos\theta\cos\theta'}{d^2} E(x)v(x,y)dx \\ &= \frac{\rho}{\pi}\int_S \frac{\cos\theta\cos\theta'}{d^2} dx \int_S E(x)v(x,y)dx \end{aligned}$$

Replacing S by $S\times E$ in the derivation of Equation 3, the "visibility" term $V(y)$ turns into:

$$\begin{aligned} V(y) &= \int_{\mathbb{R}^2} S(x)E(x)v(x,y)dx \\ &= \frac{1}{\alpha^2}(s'_\alpha \star p_{1+\alpha}) \end{aligned}$$

Where

$$s'_\alpha(x) = S(-\frac{1}{\alpha}x)E(-\frac{1}{\alpha}x)$$

This is another convolution, which can easily be calculated using our method by equipping the source with a texture containing its relative exitance function before rendering it to the offscreen buffer. We essentially include the variations of the source's emission in the visibility integral, with the double advantage that (a) the calculation of the unoccluded illumination is not modified, and (b) the potential correlation between visibility and source emission is properly accounted for.

Note that the same approach could lead to adapt our method for translucent occluders by replacing the binary term $P(x)$ by a more general one varying in the range $[0,1]$. However, translucent object generally operate in a non linear manner on light propagation (because of refraction and diffusion), which prevents us from deriving a proper convolution based formula.

4.2 Complex light source shapes

Three-dimensional light sources do not require much more computation than a planar light source: All we need is the projection of the source onto its virtual plane. Apart from computing its projection in the offscreen buffer, using a volumetric source requires attention to be paid to the choice of direction $\mathcal{D}$, which will not follow the same criteria as those described for a planar light source.

4.3 Simultaneously shadowing a group of objects

Just as for a polygonal receiver, a shadow map can be assigned to an entire cluster. The shadow map is then shared among objects, while each surface receives its own texture coordinates.

Note that this does not address self shadowing in the cluster, which may be achieved by applying our method using one part of the receiving cluster as a potential occluder for the remaining part.

5 Practical computation of shadow textures

We now describe our implementation, in which we use the convolution operation to create soft shadow textures. We have integrated this algorithm in our research testbed for hierarchical radiosity, but it should be noted that it can be used in other environments as well. We make use of two features of the radiosity system: first, we use the form factor calculation routines to evaluate the unoccluded illumination term. Second, we use the hierarchy of object clusters to select potential occluders between a given pair of source and receiver. Other techniques could be used to compute the illumination, such as Heckbert's combination of hardware point sources [13]. As for the cluster hierarchy, common structures such as hierarchies of bounding volumes are easily constructed and provide the necessary hierarchy.

Let us assume for now that we have selected a light source, a receiver object and a cluster of occluders. Such configurations can be automatically selected by ranking their potential for the creation of soft shadows, as a function of their absolute and relative sizes and distances. For each of the issues discussed below, we suggest a suitable strategy or solution.

5.1 Choice of the direction of projection

As suggested earlier, the direction of projection $\mathcal{D}$ must adequately represent the set of all possible rays between the source and the receiver. If the source and receiver are planar surfaces, we first determine a *useful receiver* by clipping the receiver by the source plane and a *useful source* by clipping the source by the receiver plane. This operation prevents $\mathcal{D}$ from being parallel to the source, which would produce a empty source image, or parallel to the receiver, which would make the computed texture projection fail. We then restrict the set of rays between the useful source and receiver to rays that actually encounter the blocker (that is, the extent of the cluster's bounding box). The direction of projection is chosen as a median value into this set.

The choice of the direction $\mathcal{D}$ does not affect the placement of umbra and penumbra regions, but for some special cases, such as the subdivision of the receiver, we shall see that it can be important not to choose $\mathcal{D}$ for each receiver independently.

5.2 Choice of the virtual planes

The choice of the altitudes of the virtual planes directly affects the size of the resulting penumbra regions in the computed texture. Altitudes for the virtual planes of the receiver, source, and blocker could simply be chosen as the centers of the altitude ranges of the three elements (See Figure 2.(b)), but more accuracy can be achieved on the resulting shadow texture by choosing the altitudes so as to obtain penumbra regions of median sizes in the range of those actually produced.

We will explain in Section 6.2 how to compute the size of the actual and computed penumbra. Using this calculation we can compute an "optimal" virtual blocker altitude which creates the desired median size: denoting by z_s and z_r the virtual source and receiver altitudes, and by z_b^{min} and z_b^{max} the extremal altitudes of blocking objects, the optimal altitude for the virtual blocker is

$$z_b^{opt} = \frac{D_1 z_b^{min} + D_2 z_b^{max}}{D_1 + D_2} \quad \text{where} \quad D_1 = z_s - z_b^{max} \quad \text{and} \quad D_2 = z_s - z_b^{min}$$

5.3 Sampling the virtual source and blocker characteristic functions

In order to perform the convolution, we need two images of the scaled characteristic functions following Equation (3). These images of the virtual source and blocker are obtained by rendering the objects that constitute the real source and blocker, using the projections previously described, in an offscreen buffer of desired size.

Source and blocker frustum are scaled to achieve the required α and $1+\alpha$ scaling factor with respect to the intrinsic dimensions of the objects, as dictated by Equation (2).

Polygons are rendered in white over a black background, with no z-buffering. Note that a non-uniform source is rendered with a texture modulating its color to follow its relative exitance function. The blocker image is inverted while reading pixel values from the offscreen buffer. In addition, the negative sign in the convolution Equation (2) means that the source image must be reflected across its horizontal and vertical axes. This is achieved by scaling the geometric model of the source with a negative factor when rendering.

Selection of the blocker frustum

The blocker frustum actually used is computed as the intersection of the receiver and blocker frustums viewed from the source (Figure 3). This avoids the computation of large unshadowed areas when the blocker is too small, and the computation of too large a texture when the receiver is too small or when the source is very close to the blocker. The *near* and *far* clipping planes are set so as to capture the blocking polygons in this regions and to avoid projecting irrelevant geometry.

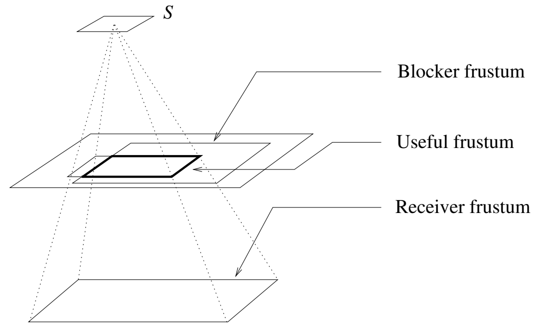


Figure 3: Construction of the blocker projection frustum

Computing the convolution

Once computed, the source and blocker images can be convolved using the following well known property:

$$(f \star g)(y) = \mathcal{F}^{-1}(\mathcal{F}(f) \times \mathcal{F}(g))$$

where $\mathcal{F}(f)$ denotes the Fourier transform of function f. Since we are dealing with $2D$ images, we perform a two dimensional FFT on each image, multiply them and finally transform the result by a normalized inverse FFT. The result is a sampled version of the visibility function $V(y)$. We use the standard FFT library supplied by SGI on our systems.

5.4 Security zone for proper convolution

When performing the convolution of two images with a Fourier transform, we implicitly assume the images to be periodic. This is obviously true for the source image by construction (because the source is strictly contained in the image), but it is not always the case for the blocker image, because of the clipping operation by the receiver frustum. As a result, the sum of the image space sizes s_1 and s_2 of the windows actually occupied by the source and blocker sampled functions must not exceed the total sampling window size

s. To ensure this property, we further scale both frustums by a factor of $\frac{s}{s_1+s_2}$, which is the secure scale factor that allows the greatest relative resolution for the effective texture (See Figure 4).

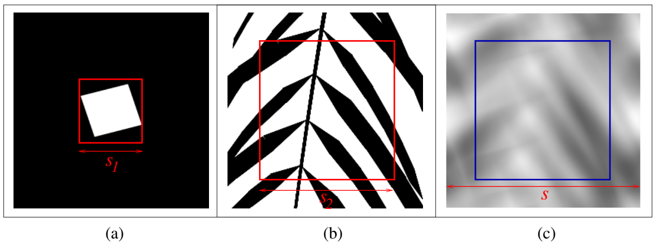


Figure 4: An example convolution between a source image (a) and a blocker image (b), for which the receiver clips the blocker frustum. The red square on the blocker image indicates the interesting area for the given receiver. The source and blocker frustums have been equally scaled until $s_1 + s_2 < s$ (note that enlarging the frustum reduces the effective size s_1 and s_2). In the resulting image (c), pixels outside the blue region are spoiled by FFT wrap-around effects and are not used in the shadow map.

5.5 Resolution issues

The resolution of the shadow map should be chosen carefully. On the one hand, it determines the resolution of all auxiliary images for the convolution operation, and the cost of the convolution itself (See the results in Section 7). On the other hand, it should be appropriate for the size of the receiver in object space and the variations of penumbra across its surface: Whereas a nearly hard shadow due to a small spotlight demands a large texture to be rendered accurately, a very smooth shadow mainly based on penumbra does not require very dense sampling. Such situations can be easily characterized since they simply depend on α. Section 7 shows practical examples of scenes with the texture sizes used.

Due to the different scaling factors, and especially for small values of α, the source sample can actually have a different area (ratio between the numbers of white and black pixels) than that of an ideally sampled image. This area plays an important role in the texture as it determines the maximum value of the convolution. Thus, a wrong area value on the source produces inappropriately normalized shadow maps, with annoying discontinuity artifacts. This problem can be addressed using antialiased rendering for the source, so that the area of the source sample has a more accurate value. Unfortunately, depending on the OpenGL implementation, the value of a pixel in an antialiased polygon is not always exactly the area of the pixel fractions covered by the polygon [19].

Blocker aliasing does not affect the resulting textures in the same way, but antialiasing is also required for the blocker characteristic function to avoid inconsistencies or discontinuities in the penumbra regions, caused by very long and fine blocking objects. The images of the mobile in Figure 12 would be particularly affected without antialiasing.

5.6 Using the shadow texture

As previously stated, the computed convolution image is used as a shadow texture on the receiver, modulated by direct illumination values.

Since we need to represent the variations of the unoccluded illumination across the receiver, we create –for display purposes only– a regular mesh of vertices P_i. Each vertex of this mesh is equipped with a color value computed as $\rho(P_i)EF_S(P_i)$.

When rendering, the receiver is displayed as a textured triangle strip set. For each vertex, texture coordinates are computed by projecting the corresponding receiver point onto the virtual receiver plane. These coordinates can either be pre-computed and stored with the mesh, or directly computed by *OpenGL* by adequately setting the texture projection matrix[19]. However, since texture coordinates are provided only for those vertices, the mesh size must account for both the illumination gradient on the receiver, and the strength of the deformation due to the projection from the virtual receiver. Practically, typical mesh sizes range from 2×2 for small polygons (For example the cubes in Figure 14) to 20×20 for walls.

The unoccluded point to area form factors are computed using the exact point-to-polygon formula[1]. When the receiver is a cluster of objects, each surface receives its own display mesh of adequate size.

6 Error-driven shadow computation

Although, by construction, our method places umbra and penumbra regions in the right place, the different approximations do not lead to exact illumination values. In this section, we first examine the different sources of error and the way to quantify this error. We then review possible refinement techniques to produce more accurate results, and finally present a hierarchical algorithm to compute the shadow texture with a given precision.

6.1 Qualitative discussion of error sources

Virtual blocker

To characterize the error due to the use of the virtual blocker, let us consider the case where the receiver is parallel to the source. Since the light source is not a single point, the umbra of the (planar) virtual blocker will differ from that of the actual blocker in the following two respects:

- When projecting the actual blocker to the virtual one, all the triple-edge discontinuity curves [15] of the discontinuity mesh collapses into Edge-Vertex events. This modification of the discontinuity mesh's internal topology affects the illumination gradient into penumbra regions. This effect is all the more noticeable that the source is large.
- Since all parts of the virtual blocker share the same altitude, all computed penumbra regions will have the same sharpness. This is the most obvious visual effect of using a virtual blocker.

Projection on the receiver

Let us consider the blocker to be planar and parallel to the source, and study the difference between the umbra directly computed on the actual receiver and that projected back from the virtual one.

Although the projection of the shadow texture back onto the actual receiver tends to produce a general shadow region of the right size, it also conserves the size ratio of penumbra and umbra regions. This ratio on an actual receiver strongly depends on the distance between the receiving point and the source. Thus the computed shadow on a large receiver may show umbra where there actually is penumbra, or the reverse.

6.2 Measuring the error

A simple way to estimate the error would be to derive a bound on the difference between the exact and computed shadow functions, depending on the virtual geometry parameters. Although such a method based on standard L_1,L_2 or L_∞ distances would produce conservative bounds on the global error, it would not allow a reliable characterization of the shadow artifacts, because of its inherent non-locality and its lack of coherence towards human perception criteria [24]. We instead consider a form of perceptual error, and study its variation in terms of the virtual geometry parameters.

The most noticeable artifact due to the use of the virtual geometry is the production of penumbra regions of inadequate size. We propose to estimate the ratio between the computed and exact penumbra sizes. The range of variation of this ratio for all points in the configuration will serve as an error measure for the use of the virtual geometry.

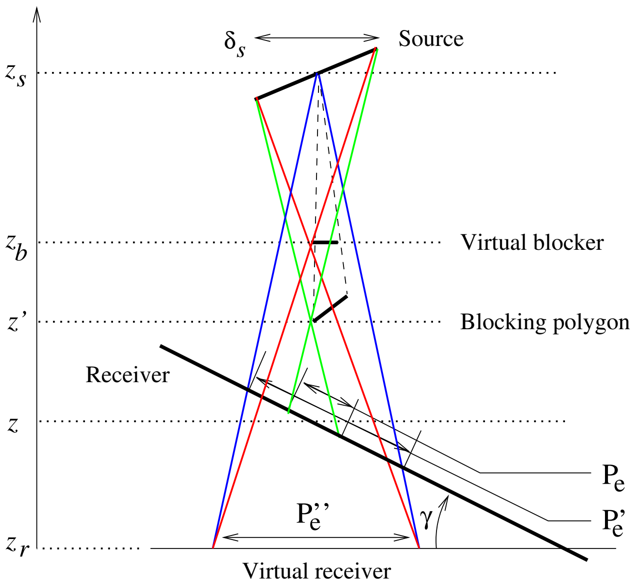


Figure 5: Sizes of computed and exact penumbra regions for a given source/blocker/receiver configuration and virtual planes altitudes z_s,z_b and z_r.

Our error estimate is derived below with a simple reasoning in two dimensions. As shown in Figure 5, the penumbra regions on the virtual receiver (follow red lines), due to a polygon side of the virtual blocker, at a given altitude z_b, have size

$$P''_e = \alpha\delta_s \quad \text{where} \quad \alpha = \frac{z_b - z_r}{z_s - z_b}$$

After projection (blue lines) onto the actual receiver, at altitude z, the computed penumbra will have approximate size:

$$P'_e(z) = \alpha\delta_s \frac{1}{\cos\gamma} \frac{z_s - z}{z_s - z_r}$$

The penumbra due to the actual blocking polygon at altitude z' can also be approximated by:

$$P_e(z', z) = \delta_s \frac{1}{\cos\gamma} \frac{z' - z}{z_s - z'}$$

Thus, the relative error between the true and computed penumbra will be

$$\Delta_e(z', z) = \frac{P'_e(z)}{P_e(z', z)} = \frac{(z_s - z')(z_s - z)}{z' - z} \times \frac{z_b - z_r}{(z_s - z_r)(z_s - z_b)}$$

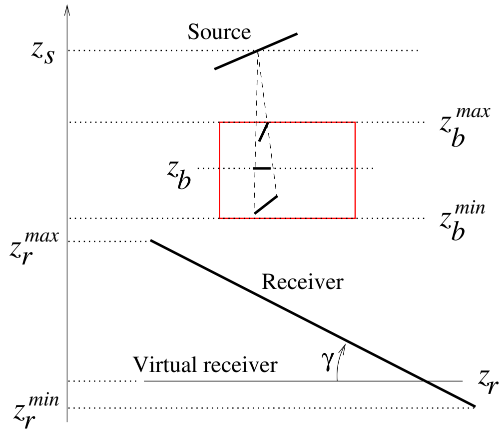


Figure 6: Altitude ranges for the blocker and receiver

Let us assume that the set of blocking objects lies between altitudes z_b^{min} and z_b^{max}, and that the receiver is bounded by z_r^{min} and z_r^{max} (Figure 6). We assume without loss of generality that

$$z_r^{max} < z_b^{min}$$

In this case, the approximation error $\Delta_e(z', z)$ reaches its maximum value for $z = z_r^{max}$ and $z' = z_b^{min}$, and its minimum value for $z = z_r^{min}$ and $z' = z_b^{max}$. The difference between these two extremal values is the maximal error amplitude for the current configuration:

$$E_{max}(Receiver, Blocker) = \Delta_e(z_b^{min}, z_r^{max}) - \Delta_e(z_b^{max}, z_r^{min}) \qquad (4)$$

As expected, this error estimate decreases to zero when the blocker and receiver become planar and parallel.

6.3 Reducing the error

Now that we can estimate the amount of approximation incurred, we consider the options available to reduce it. We first list all potential parameters of the problem, and focus on the combination of several shadow maps corresponding to sub-clusters of a given occluder.

6.3.1 Parameters influencing shadow quality

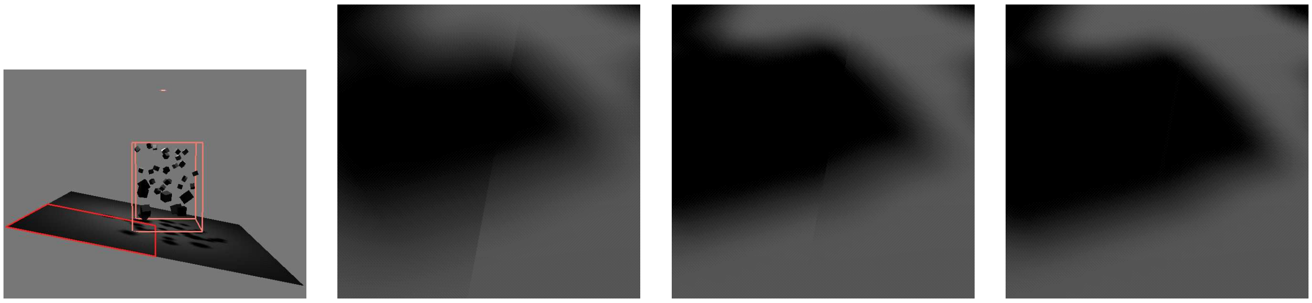
Source subdivision Subdividing the source would help reducing the discontinuity mesh topological error described in Section 6.1, and also improve on the kernel-visibility low correlation assumption. Since these two kinds of error do not significantly affect the visual aspect of the shadow, except for very large light sources, we currently ignore this option.

Image resolution Improving on the shadow texture resolution or on the choice of the direction of projection helps reduce their specific error, but it does not lead to arbitrarily accurate shadow textures, in terms of penumbra accuracy.

Therefore, it is generally more efficient and practical to subdivide either the receiver or the blocker as sketched in Figure 7.

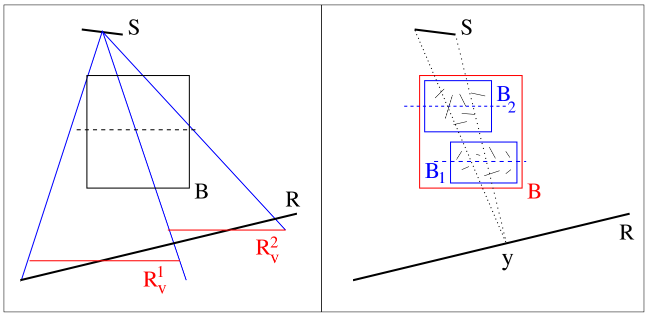
Receiver subdivision Subdividing the receiver into two or more sub-receivers accounts for the different ratios of characteristic sizes of umbra and penumbra for different receiving regions (Figure 7.a).

In this case, a shadow texture is computed separately for each sub-receiver, using the convolution method. This increases the



Automatic choice of direction | Source direction | Receiver direction

Figure 8: Artifacts produced along a subdivision boundary on the receiver.



(a) (b)

Figure 7: (a) A receiver subdivided into two receivers, with their associated virtual receiver. (b) A blocking cluster subdivided into two blocking sub-clusters, with their associated virtual blockers.

computation time, due to the larger number of convolutions to compute, but takes into account different α configurations for the different parts of the subdivided receiver.

Particular attention should be paid to the choice of direction $\mathcal{D}$ for each sub-receiver, as illustrated in Figure 8. As each texture is projected back to its own receiver, boundary artifacts may appear: in this Figure, a receiver patch is subdivided into four receivers (left). The next three images (from left to right) show a shadow detail in the boundary region of two sub-receivers, with three different choices for $\mathcal{D}$:

automatic $\mathcal{D}$ is chosen independently for the four receivers as described in Section 5.1. Note the discontinuity along the boundary, due to non-matching shadow textures.

source $\mathcal{D}$ is the source's normal direction. The four receivers thus have the same direction $\mathcal{D}$, but penumbrae still have different sharpness. The discontinuity is barely noticeable in the penumbra.

receiver $\mathcal{D}$ is the receiver's normal. The four receivers have the same virtual plane, and shadows fit together perfectly.

Thus we see that a proper choice of $\mathcal{D}$ (the same direction for all receivers, in this case) can reduce or eliminate most of the artifacts.

Blocker subdivision When a large set of blocking objects (grouped in a cluster) projects shadows on the receiver, it produces penumbra regions of different sizes. In such a configuration, subdividing the receiver would not suffice. We can expect a more accurate result by considering separately subsets of objects in the cluster (Figure 7.b), computing their associated shadow texture using our convolution method with suitable virtual blockers.

Such a subdivision requires a procedure for combining shadow maps created from the subclusters into a shadow map corresponding to the entire blocking cluster. This issue is addressed in the next section.

6.3.2 Combination of shadow maps from different subclusters

When two blockers are treated separately, all information on their spatial correlation is lost. Thus, exact recombination of the two shadow maps requires the knowledge of the correlation function of the two blockers. The simple experimental case described in Figure 9 illustrates the impossibility of retrieving an exact shadow map value knowing only that of the subclusters.

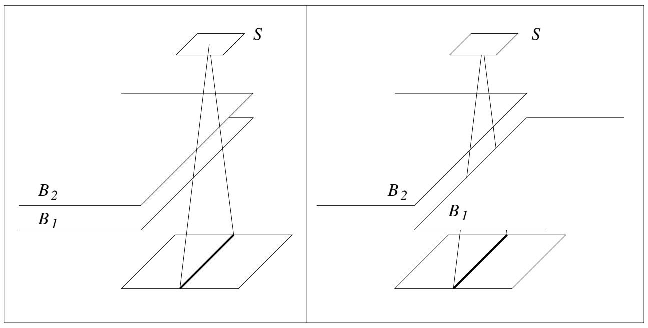


Figure 9: Extremal situations for blocker-to-blocker correlation. For any receiver point y on the bold line, the visibility values $V_1(y)$ and $V_2(y)$ for each separate blocker are exactly $\frac{1}{2}S$. But the actual visibility $V(y)$ is $\frac{1}{2}S$ in the left-hand case, and 0 in the right-hand one. These extremal values are in fact those given by Equation (5).

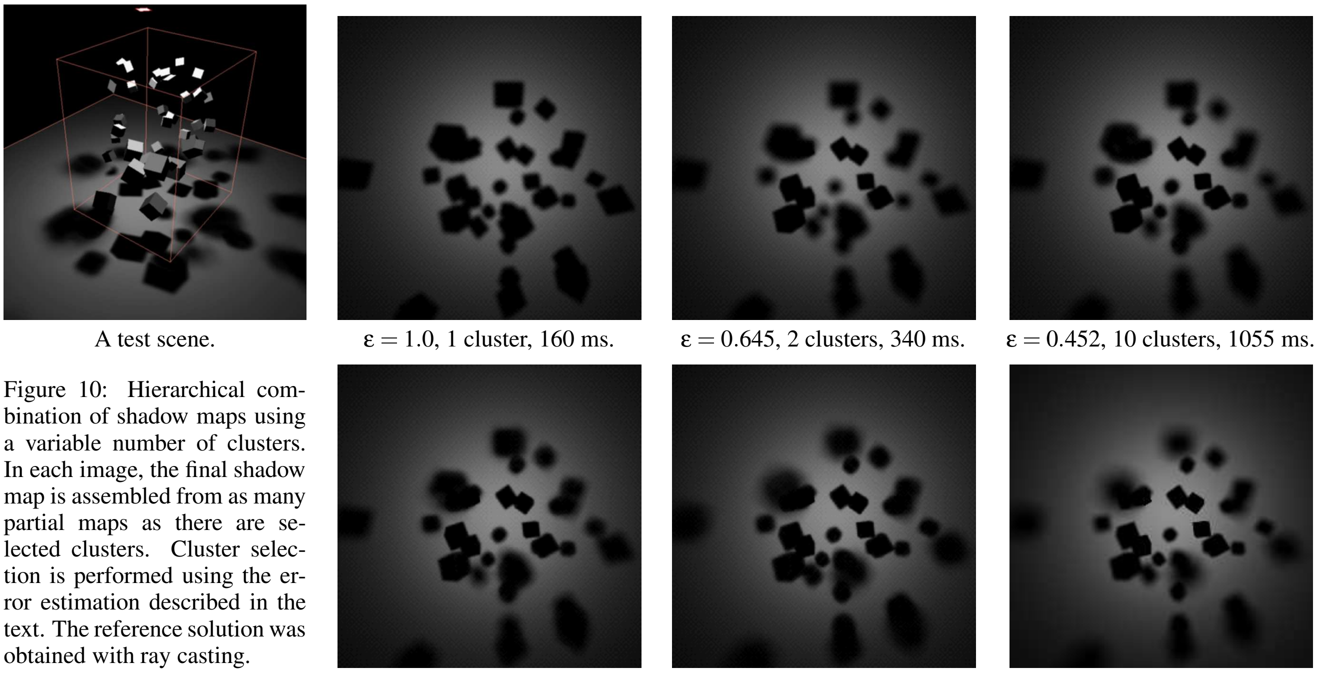
We propose an approximation method to achieve such a combination for the case of a subdivision into two subclusters. This combination method generalizes readily for more sub-clusters. Let us call $V_1(y)$ and $V_2(y)$ the shadow maps computed for each sub cluster separately, and $V(y)$ the shadow texture associated to the parent cluster. Recalling that the value of the shadow texture is the area of the portion of the source that is visible from a receiver point, we can consider the worst and best correlation cases between blockers 1 and 2 and write:

$$V_2(y) - (S - V_1(y)) \leq V(y) \leq \min(V_1(y), V_2(y))$$

and thus:

$$\max(0, V_1(y) + V_2(y) - S) \leq V(y) \leq \min(V_1(y), V_2(y)) \qquad (5)$$

(S is the area of the source). Thus, we can use the following median



A test scene. — $\varepsilon = 1.0$, 1 cluster, 160 ms. — $\varepsilon = 0.645$, 2 clusters, 340 ms. — $\varepsilon = 0.452$, 10 clusters, 1055 ms.

$\varepsilon = 0.387$, 34 clusters, 3.38 s. — $\varepsilon = 0.0$, 210 clusters, 21 s. — Reference solution.

Figure 10: Hierarchical combination of shadow maps using a variable number of clusters. In each image, the final shadow map is assembled from as many partial maps as there are selected clusters. Cluster selection is performed using the error estimation described in the text. The reference solution was obtained with ray casting.

value for the combined texture, as an approximation of $V(y)$:

$$V_{1,2}(y) = \frac{1}{2}\left(\min(V_1(y), V_2(y)) + \max(0, V_1(y) + V_2(y) - S)\right) \quad (6)$$

The maximum error incurred by this approximation arises in the two configurations depicted in Figure 9 where it reaches the value $\frac{1}{4}S$. It should be noted that such configurations rarely occur, only when two polygons sharing an edge (as viewed from the source) are treated as separate blockers. This is the case in the last image of Figure 10 where the 210 polygons forming the cubes have been used as separate blockers. The visible effect is a slightly faster variation of the penumbra around the shadow of the cubes in comparison to the reference solution.

6.4 Shadow approximation algorithm

We can now organize the preceding elements into a complete hierarchical refinement algorithm, controlled by explicit error estimation.

Refinement criterion

For a given configuration, Equation (4) gives an estimation of the error due to the use of the convolution method. By comparing the error estimates for the two separate cases of blocker and receiver subdivision, we can decide which choice leads to the smallest error on the final texture:

$$\begin{aligned} E(R_1, \ldots, R_n, B) &= \max\left(E_{max}(R_1, B), \ldots, E_{max}(R_n, B)\right) \\ E(R, B_1, \ldots, B_p) &= C_{Err} + \max\left(E_{max}(R, B_1), \ldots, E_{max}(R, B_p)\right) \end{aligned}$$

C_{Err} is a *combination error* term, that is a bound on the error due to the correlation of sub-cluster shadow maps. It turns out to be negligible in practice. An important property is that for any subcluster b of B, and any sub-receiver r of R, we have

$$E(r, b) \leq E(R, B)$$

Equality occurs only when $b = B$ and $r = R$. The refinement algorithm is summarized in Figure 11, where the procedure *ConvolutionTexture(S,B,R)* computes the shadow texture using the given source, blocker and receiver. *CombineTextures* performs the combination of Eq.6 and *PasteTextures* makes a single texture from the texture of the four sub-receivers. When subdividing the cluster of occluders, we save and re-use the Fourier transform of the source image, and adapt the scale of the occluder images to account for the fixed size of the source, thereby saving the cost of one FFT per occluder used. Figure 10 shows the results of the application of this algorithm for different error thresholds. It clearly shows that for a single cluster, all penumbra regions have the same extent, while the range of possible penumbra sizes increases with the number of clusters. The reference solution was computed using ray casting on the true geometry, with 1024 rays per pixel.

```
Texture ComputeTexture(S,B,R)
  if E_max(S,R,B) < ε_max
    return ConvolutionTexture(S,B,R)
  else
    if E(R_1,...,R_n,B) < E(R,B_1,...,B_p)
      T_1 = ComputeTexture(S,B,R_1)
                ...
      T_n = ComputeTexture(S,B,R_n)
      return PasteTexture(T_1,...,T_n)
    else
      T_1 = ComputeTexture(S,B_1,R)
      ...
      T_p = ComputeTexture(S,B_p,R)
      return CombineTextures(T_1,...,T_p)
```

Figure 11: Algorithm for shadow texture computation

Discussion of convergence

Each leaf of the cluster hierarchy contains one or more surfaces, arbitrarily oriented. Thus we cannot refine the blocking clusters into arbitrarily small subclusters. The maximum extent δ_z of atomic objects in the blocking cluster produces the minimum possible error of the computed texture, using criterion (4). The associated texture artifact will be localized in the shadow region produced by the object, and can be imputed to the object model quality.

Conversely, the possibility of refining the receiver is only limited by the allowed computation time for the shadow map.

Our subdivision criterion does not take into account the size and orientation of the source, which means that it does not capture the totality of the error, and that the images do not fully converge to the true images.

7 Results

We present in this section a number of images illustrating the results of our algorithm. As a general rule, shadowed polygons (walls, objects) are entirely illuminated using our convolution method, whereas blockers themselves are illuminated using standard radiosity computation, without any extra visibility treatment. Computation times are given for the shadow texture treatment only, since any illumination method could be used for other objects, including hardware lighting.

7.1 Breakdown of computation time

The scenes used for the following images contain between 212 and 45,000 polygons, mainly concentrated into the blockers. Offscreen rendering times range from less than 1 ms to about 160 ms on SGI Onyx2/iR and $O2$ computers. The cost of one FFT calculation is proportional to $n \log n$, where n is the number of pixels in the images. Therefore this cost is very sensitive to the resolution of the shadow maps. On the Onyx2 we observe the following computation times (in milliseconds, for the calculation of a single representative map such as that the floor). The corresponding images can be seen in Figure 10 (Cubes, 212 polygons), Figure 14 (Pyramid, 4340 polygons) and Figure 12 (Mobile, 45000 polygons).

Text. size	FFT	Cubes	Pyramid	Mobile
128	6	50	60	220
256	25	170	170	340
512	110	610	620	770
1024	500	2510	2540	2730

For comparison, the same operations on the $O2$ take similar amounts of time:

Texture size	FFT	Cubes	Pyramid
512	170	1100	1190
1024	770	5440	4980

In each case we indicate the total time to compute a shadow map, and in the FFT column the time for a single FFT operation. Three FFTs are needed to obtain a shadow map, and other image-based operations take another 140% of the time of a single FFT. We see that the cost of FFTs dominates for textures of 512^2 pixels and higher. Note that these are fairly large texture sizes, used only for large polygons (walls...) in the images. Smaller textures suffice for most objects.

7.2 Hierarchical combination of shadow maps

Figure 12 demonstrates the use of the hierarchical shadow map combination of Section 6.3.2. The scene contains 45,000 polygons, mostly in the complex objects attached to the mobile. The resolution of the three shadow maps is 256×256 for the images shown, although it should be noted that half this size produces almost indistinguishable results. Therefore we indicate computation times for both 128 and 256 resolution.

7.3 Casting shadows on several surfaces at once

Figure 14 shows a cluster of cubes for which a single 128×128 shadow map has been calculated. Each polygon of the cluster is equipped with a coarse display mesh (2×2 to 3×3) containing unoccluded form factor to the source. 512×512 shadow maps are used for the floor and walls. The same occluding cluster containing the plant has been used for all maps. The plant itself is made out of $4,340$ polygons.

7.4 Complex light sources

Figure 13.(a) illustrates the lighting effects that can be simulated when a light source with a complicated shape casts shadows. The "98" shape is three-dimensional text, made of 360 triangles. A 256×256 texture has been computed for each of the three walls using the "hole" cluster as an occluder.

Figure 13.(b) shows how a single convolution can create the effect of several small sources. Note that the stepping effect is normal here, because there really are four distinct small sources in the scene. A single image of the cluster of light sources is used (shown in Figure 13.(c)).

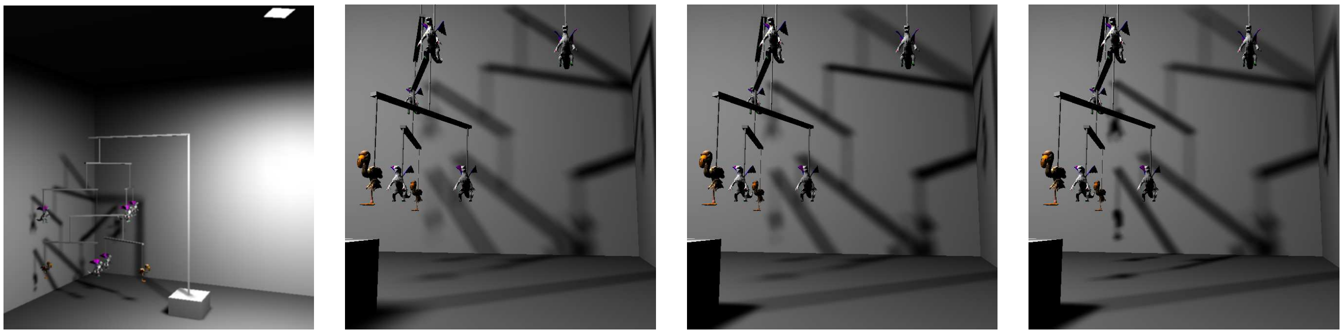
Figure 13.(d) demonstrates shadows cast by elongated light sources: with neon tubes, penumbra regions are smooth in the direction parallel to the tubes, but exhibit a stepping effect in the direction perpendicular, to the axes of the tubes.

8 Discussion

Our results demonstrate the advantage of a convolution method over an explicit sampling method, in that penumbra regions are always continuous. Note that we are still performing a discrete sampling of the source via the offscreen rendering step, but we are able to treat many samples in a single operation (and the samples are antialiased). Our method can be considered to encompass Heckbert's [13] since we can simulate an extended light source with non-uniform exitance distribution, where only a fixed number of sample points have non-zero energy. In our images, we have chosen to use light sources of small or moderate size, so that shadows contained identifiable penumbra/umbra regions. Naturally our method works for light sources of any size, whereas sampling methods would have to use very large numbers of samples.

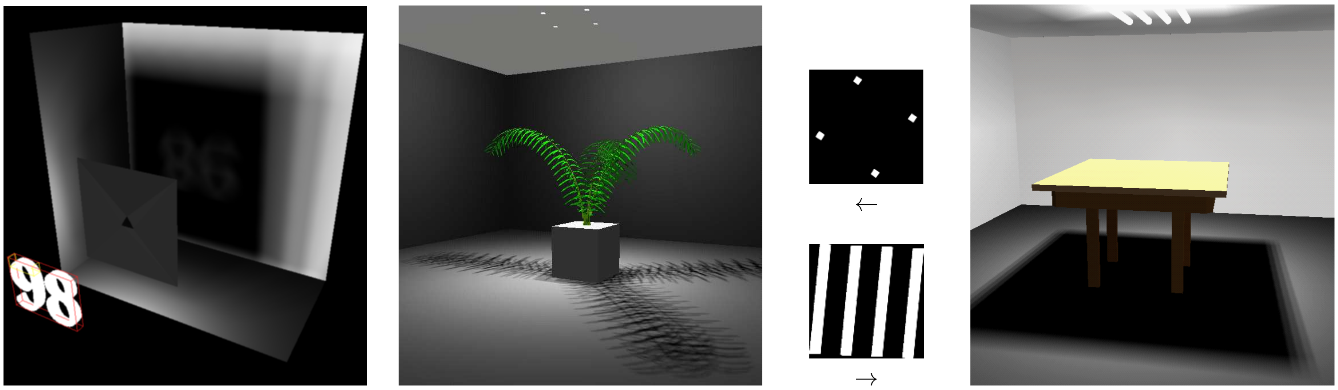
Even when no subdivision is performed (i.e. with a single occluding cluster grouping all potential occluders), the method produces visually pleasing images without stepping effects. Note that all occluders are always taken into account exactly once with their complete shape, no matter how many clusters are used in the hierarchical subdivision. In this respect, our method provides a better solution to multi-resolution shadow calculation than the simple volume approximation of [24]. As more hierarchical levels are used to recombine shadow maps, better shadows are obtained, and the computation time becomes dominated by the cost of the FFT.

Interestingly, we observe that our method in essence trades graphics performance for raw compute power, since it renders a single image of the source and occluder but requires a number of FFT calculations. This paradigm shift appears consistent with the



1 cluster, 660 ms / 980 ms. 7 clusters, 1.1s / 2.75s. 21 clusters, 2.2s / 7.4s.

Figure 12: Hierarchical combination of shadow maps: results obtained with different error thresholds, requiring more and more shadow textures to be computed. Two timings are given for each image (3 textures in each), for texture resolutions of 128^2 and 256^2 pixels.



(a) (b) (c) (d)

Figure 13: Shadows cast by complex light sources: (a) 3D source (b) set of small sources (c) light source images (d) elongated sources .

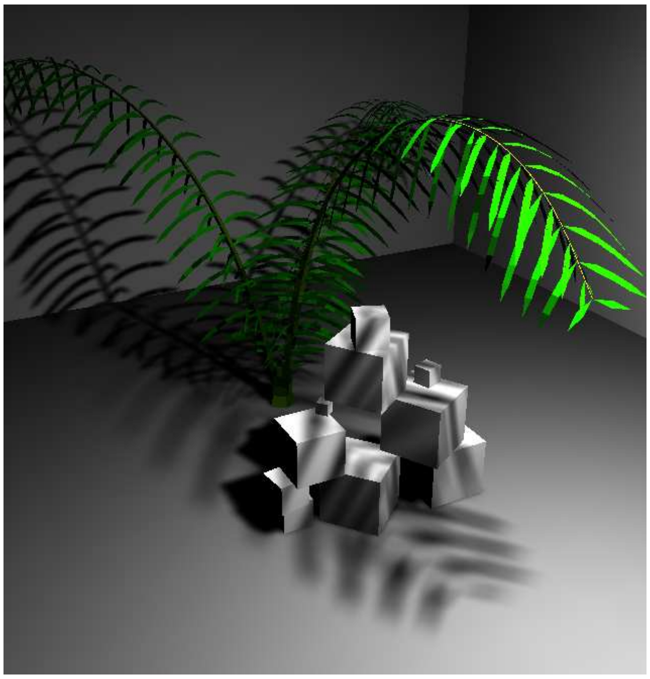


Figure 14: A single 128×128 shadow map was computed for the cluster of cubes, and used to obtain shadows on each individual cube according to its location in space.

evolution of computer technology. We also note that DSP chips are commonly found on multimedia computers, and significantly accelerate FFT calculations. In fact, tests run on DSP-equipped *O*2 computers show that the FFT cost for large images is comparable to that of the Onyx2.

Finally, the algorithm is highly parallelisable. Not only can we compute FFTs in parallel, but also the recombination operations for blocker or receiver subdivision.

As for any approximation method, there exist extremal cases where our algorithm does not work properly. One example of this could be obtained using a large blocking polygon that lies in a plane containing the direction $\mathcal{D}$. In such a case, the blocker image is nearly empty and hardly no shadow is produced. Subdividing the source into two regions that see a particular side of the polygon and adding the associated shadow maps together would correct this problem.

Large objects touching the receiver also produce bad configurations unless they can be subdivided because the ideal α values for such objects range from 0 (for parts of the polygons that touch the receiver) to larger values that produce smoother shadows. For a table lying on the floor, for example, although the shadows are produced in the right place, they appear to be too smooth where the table legs touch the floor. In such a configuration, explicit sampling methods would produce better results [13].

9 Conclusions and Future Work

We have presented a new calculation method for soft shadows from extended light sources. The method is based on the expression of visibility functions using convolution operations. It allows the simulation of soft shadows from complex light sources or clusters, having complex shapes and non-uniform exitance distributions. Receivers can be individual surfaces or object clusters, in which case shadows are correctly cast on all objects of the cluster. Occluders can be arbitrary object clusters.

The approximations introduced by the formulation as a convolution have been discussed, and a hierarchical algorithm has been proposed for the combination of shadow maps from sub-clusters. A subdivision criterion was derived to limit the error incurred in the size of the penumbra regions. The algorithm is automatic and can be readily integrated in existing rendering systems.

Future work includes the extension of the convolution approach to other illumination problems. For instance the illumination by the hemispherical sky dome can also be expressed as a convolution. This was first shown by Max in a restricted case where a horizontal plane is used to model skylight [17]. The expression of the illumination kernel, in the absence of occlusion and for parallel source/receiver pairs, is also a convolution.

The current hierarchical combination algorithm will not be able to compute an exact shadow if the clusters contain an object whose "vertical" extent (along the direction of interest) is too large. More elaborate refinement criteria should include provisions to identify such cases and provide alternate methods to compute associated shadow maps, which can then be combined in the same way with those obtained by convolution.

Another important research direction is the re-use of source and occluder images: saving the cost of the associated FFT would significantly accelerate the process for large textures. Image-based rendering methods could perhaps be adapted to derive such images from a set of precomputed images. Such a derivation would have to take place in the Fourier domain to be really effective.

10 Acknowledgments

This work was supported in part by grants from the Région Rhône-Alpes (EMERGENCE) and the European Union (Esprit LTR 24 944: ARCADE). The anonymous reviewers provided a wealth of useful suggestions and comments, which greatly helped improving the paper. Many thanks to Frédo Durand, George Drettakis, and all the iMAGIS team for fruitful discussions and moral support.

References

[1] Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors. *Computer Graphics*, 23(3):325–334, July 1989. Proceedings SIGGRAPH '89.

[2] Lynne Shapiro Brotman and Norman I. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics and Applications*, 4(10):5–24, Oct. 1984.

[3] Norman Chin and Steven Feiner. Near real-time shadow generation using bsp trees. *Computer Graphics*, 23(3):99–106, July 1989. Proceedings SIGGRAPH '89.

[4] Michael F. Cohen and Donald P. Greenberg. The hemi-cube : A radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. Proceedings SIGGRAPH '85.

[5] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. *Computer Graphics*, 18(3):137–145, July 1984. Proceedings SIGGRAPH '84.

[6] Franklin C. Crow. Summed-area tables for texture mapping. *Computer Graphics*, 18:207–212, July 1984. Proceedings SIGGRAPH '84.

[7] Paul J. Diefenbach and Norman I. Badler. Multi-pass pipeline rendering: Realism for dynamic environments. In *1997 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 1997.

[8] George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In *Proceedings SIGGRAPH '94*, pages 223–230, 1994.

[9] Paul Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. *Computer Graphics*, 24(4):309–318, August 1990. Proceedings SIGGRAPH '90.

[10] Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. Proceedings SIGGRAPH '91.

[11] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.

[12] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics*, 24(4):145–154, August 1990. Proceedings SIGGRAPH '90.

[13] Paul S. Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical Report TR CMU-CS-97-104, Carnegie Mellon University, January 1997. See also the technical sketch *"Fast Soft Shadows" at SIGGRAPH'96*.

[14] Alexander Keller. Instant radiosity. In *Proceedings SIGGRAPH '97*, pages 49–56, 1997.

[15] Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.

[16] Nelson Max. Sun and shade, 1988. SIGGRAPH Video Review, ACM. Issue 36, Segment 8. Available from First Priority, P.O. Box 576, Itasca, IL 60143.

[17] Nelson Max. Unified sun and sky illumination for shadows under trees. *CVGIP: Graphical Models and Image Processing*, 53(3):223–230, May 1991.

[18] Karol Myszkowski and Tosiyasu L. Kunii. Texture mapping as an alternative for meshing during walkthrough animation. In *Photorealistic rendering techniques (Proceedings of the Fifth Eurographics Workshop on Rendering)*, pages 375–388. Springer-Verlag, June 1994.

[19] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide*. Addison-Wesley, Reading MA, 1993.

[20] Pierre Poulin and John Amanatides. Shading and shadowing with linear light sources. In *Eurographics '90*, pages 377–386. North-Holland, September 1990.

[21] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. *Computer Graphics*, 21(4):283–291, July 1987. Proceedings SIGGRAPH '87.

[22] Peter Schröder and Pat Hanrahan. On the Form Factor Between Two Polygons. In *Proceedings SIGGRAPH '93*, pages 163–164, 1993.

[23] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadows and lighting effects using texture mapping. *Computer Graphics*, 18(2):249–252, July 1992.

[24] François Sillion and George Drettakis. Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination. In *Proceedings SIGGRAPH '95*, pages 145–152, 1995.

[25] François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann publishers, San Francisco, 1994.

[26] Cyril Soler and François Sillion. Accurate Error Bounds for Multi-Resolution Visibility. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 133–142. Springer-Verlag/Wien, 1996.

[27] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):315–324, July 1989. Proceedings SIGGRAPH '89 in Boston.

[28] Leonard R. Wanger, James A. Ferwerda, and Donald P. Greenberg. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications*, 12(3):44–58, May 1992.

[29] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23:343–349, 1980.

[30] Lance Williams. Casting curved shadows on curved surfaces. *Computer Graphics*, 12(3):270–274, August 1978. Proceedings SIGGRAPH '78.

[31] Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, Nov. 1990.

[32] Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. In *Proceedings SIGGRAPH '93*, pages 213–220, August 1993.