Texture-Based Visibility for Efficient Lighting Simulation

Cyril Soler
and
François X.Sillion
iMAGIS – GRAVIR/IMAG - INRIA

Lighting simulations obtained using hierarchical radiosity with clustering can be very slow when the computation of fine and artifact-free shadows is needed. To avoid the high cost of mesh refinement associated with fast variations of visibility across receivers, we propose a new hierarchical algorithm in which partial visibility maps can be computed on the fly, using a convolution technique, for emitter-receiver configurations where complex shadows are produced. Other configurations still rely on mesh subdivision to reach the desired accuracy in modeling the energy transfer. In our system, radiosity is therefore represented as a combination of textures and piecewise constant or linear contributions over mesh elements at multiple hierarchical levels. We give a detailed description of the *gather*, *push/pull* and *display* stages of the hierarchical radiosity algorithm, adapted to seamlessly integrate both representations. A new refinement algorithm is proposed, that chooses the most appropriate technique to compute the energy transfer and resulting radiosity distribution for each receiver/emitter configuration. Comprehensive error control is achieved by subdividing either the source or receiver in a traditional manner, or by using a blocker subdivision scheme that improves the quality of shadow masks without increasing the complexity of the mesh. Results show that high-quality images are obtained in a matter of seconds for scenes with tens of thousands of polygons.

Categories and Subject Descriptors: I.3.7 [**Computer graphics**]: Three-Dimensional Graphics and Realism—*Radiosity, Color, shading, shadowing, and texture*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Global illumination, Texture-based visibility, Convolution, Hierarchical radiosity

1. INTRODUCTION

Radiosity methods [Cohen and Wallace 1993; Sillion and Puech 1994] can produce very high quality images, because they compute the global balance of light energy and simulate subtle effects such as indirect illumination. However these methods are typically very expensive, due to their extensive usage of visibility calculations, especially in the presence of shadows: fine shadow details are obtained by mesh subdivision, driving the calcula-

An earlier version of this paper appeared in *Rendering Techniques'98* Springer, NY, 1998. Pages 199 – 210.
iMAGIS is a joint project of CNRS, INRIA, INPG and UJF within the GRAVIR/IMAG laboratory.

tion cost up. This paper addresses the issue with a new algorithm that uses an efficient error-driven shadow calculation method based on convolution, together with a more traditional mesh-based radiosity calculation. This allows an effective de-coupling of the mesh subdivision used for energy balancing and the visual quality of shadows, with a graceful combination of refinement strategies operating on either the shadow representation or the calculation mesh.

Simulating the balance of light energy in a scene requires to account for all possible interactions between pairs of points in the environment. The original radiosity method [Goral et al. 1984; Cohen et al. 1988] approximates the solution to this problem by a piecewise polynomial function, obtained by discretizing the integral equation that describes the equilibrium of light energy [Kajiya 1986]. Like other Galerkin methods, the radiosity algorithm requires to solve a linear system to find the parameters on which the approximation of the solution depends. Paradoxically, the cost of solving this system is much smaller than the cost of computing its coefficients, which measure the contribution of each mesh element to the radiosity of other mesh elements. As a consequence the cost of a naive radiosity method is nearly quadratic in the number of mesh elements.

Hierarchical radiosity techniques [Hanrahan et al. 1991; Schröder et al. 1993] can limit this computation cost by determining the optimal set of interactions needed to achieve a given image quality. Energy transfers (or *links*) are established between hierarchical basis functions, and by deciding at which hierarchical level the transfers are computed a continuous compromise between quality and speed can be offered. Although this method has offered a great acceleration over the original radiosity algorithm, the remaining $\mathcal{O}(N^2)$ cost of initial linking all pairs of surfaces has remained the main bottleneck until clustering methods [Smits et al. 1994; Sillion 1995] were introduced. By prolongating the hierarchical representation of the scene up to groups of surfaces and objects between which higher level links could be established, clustering techniques reduce the computation cost to $\mathcal{O}(N \log N)$.

Despite the great adaptability of hierarchical radiosity and clustering methods to configurations in a wide range of complexity levels, the main problem in applying these algorithms to image synthesis using a straightforward mesh subdivision scheme remains the computation of shadows: as users of radiosity systems know too well, attempting to obtain very high quality shadows often pushes the system to its limits in terms of memory and computation time. This is easily explained by the following observations:

First, simple (constant or linear) mesh elements are not adapted to represent shadows since they naturally produce discontinuities in the illumination or its derivatives. Therefore, obtaining realistic shadows under this representation requires a large number of small elements (ideally with a sub-pixel size). However, it is also known that in order to be deemed visually satisfactory, shadows do not necessarily have to be exact but rather free of artifacts [Wanger et al. 1992].

Secondly, mesh refinement is generally based on simple mathematical error estimators (*e.g* L_∞ or L_1 norms) which is efficient for driving the overall convergence of the light propagation iterations according to the same error measures. However, these error estimators do not account for the visual quality of the solution, which only becomes satisfactory for fully refined solutions. Only approaches based on a global human perception of the solution [Gibson and Hubbold 1997] (See also [Prikryl and Purgathofer 1998] for a list of such techniques) are able to limit mesh refinement in an intelligent way, during the computation of the solution or as a post process. Such methods however still require the mesh to be sufficiently fine to follow the visible shadows.

As a consequence, the mesh refinement required for a radiosity solution aimed at synthetic image generation is usually much deeper than the refinement required to simulate the global energy balance with reasonable accuracy. The presence of fine shadow details, indeed, pushes the depth of the refinement far beyond the point where it visually influences the accuracy of the light energy balance everywhere else in the scene.

Because it produces artifact-free shadow maps in short computation times, the convolution technique for generating soft shadows [Max 1991; Soler and Sillion 1998b] appears very promising for shadow generation in hierarchical radiosity/clustering methods. Besides, the *convolution method* described in [Soler and Sillion 1998b] includes an error control algorithm driven by a perceptual error measure, based on the subdivision of the hierarchy of blockers casting shadows, which can be integrated into a classical hierarchical radiosity refinement algorithm. With respect to speed, this method produces very complex shadows within much faster computation times than those needed to obtain an equivalent visual quality with classical hierarchical radiosity mesh elements. It can therefore be used to accelerate the computation of radiosity contributions in regions occupied by fine shadow details. Finally, the convolution method [Soler and Sillion 1998b] is very robust, in the sense that it adapts to many extended configurations, such as casting shadows from complex light sources onto groups of objects.

We therefore propose to use the convolution method as an alternative to deep mesh refinement in regions where fine shadow details occur, still keeping the mesh-based radiosity representation for energy transfer configurations where the convolution method would not be beneficial. Since this method computes textures, our new algorithm manages radiosity under two different forms (mesh-based and texture-based) inside a single hierarchical algorithm using clustering.

In summary this paper addresses the following issues:

—Designing a hierarchical radiosity algorithm with clustering, where radiosity can be represented as both textures and classical mesh elements. This includes the description of new push-pull, gather and display algorithms;

—the construction of a refinement algorithm that integrates error control methods from both the convolution method and the hierarchical radiosity algorithm;

—the de-correlation of visibility calculations from the computation of the global light energy balance, using an adaptive algorithm for computing soft shadow maps, based on convolutions;

—the automatic categorization of configurations where the convolution method proves to be more interesting than the classical radiosity mesh elements;

—the automatic evaluation of various parameters used in the on-the-fly computation of shadow masks.

In Section 2 we review some pieces of previous work related to our new algorithm. Section 3 recalls the principles of the convolution method, and motivates its use to compute shadow masks into a hierarchical radiosity algorithm with clustering. Sections 4 and 5 detail the different steps of light energy propagation in the scene using the proposed method. In Sections 6 and 7 we present respectively the new refinement algorithm and a discussion of issues related to displaying the solution on screen. Finally, Section 8 is devoted to the presentation of results and performance tests for a variety of scene configurations.

2. PREVIOUS WORK

Our technique explicitly stores some contributions to the radiosity function into textures (computed on the fly instead of refining the mesh) in shadowed areas where classical mesh subdivision would require heavy calculations. Besides, the calculation of the shadows benefits from an error control algorithm integrated into the hierarchical radiosity error control scheme. Our method is therefore related to four classes of techniques:

(1) radiosity methods that resort to a specific treatment for shadow computation to improve solution images. This includes *final gathering*, *discontinuity meshing* and *shadow masking*.
(2) radiosity methods that use a multi-resolution shadow computation algorithm.
(3) radiosity methods that use textures as a way to store radiosity during the light propagation phase ;
(4) methods that use textures for rendering a radiosity solution more rapidly.

We explore more deeply the first two classes of methods in Section 2.1 and have grouped the other two classes in Section 2.2. Finally, Section 2.3 positions our approach with respect to these different algorithms.

2.1 Radiosity with specific shadow treatment

After a radiosity solution has been computed, the technique of *final gathering* [Reichert 1992] consists in a ray-tracing pass during which each visible pixel in the scene is considered. All links arriving on a parent hierarchical element of the pixel are collected, and the light energy contributions for the corresponding source patches are re-computed specifically for each point considered. This method is therefore a post-process. Although it produces very clean images with perfect shadows, it is highly time consuming, since many visibility calculations are required for each visible point. Finally, this method is also view dependent.

Discontinuity meshing [Heckbert 1992] consists in analytically determining visual events that cause discontinuities of various orders in the gradient of the illumination, on surfaces receiving light. Using this technique, the radiosity mesh can be chosen so that it respects regions in which the percentage of occlusion varies linearly. This produces very accurate and visually pleasant images including soft shadows [Lischinski et al. 1992; Durand et al. 1999]. The main drawback of discontinuity meshing is that it suffers from high numerical instability and has a large computation cost. Moreover, the mesh produced is usually quite complex hence a larger convergence cost for the radiosity iterations.

Storing shadows in a separate data structure such as shadow maps [Williams 1978], has turned out to be a very interesting way of eliminating the prohibitive cost of a deeply refined radiosity mesh. Zatz proposed to incorporate shadow maps into a high order radiosity algorithm [Zatz 1993]. He showed that computing 40×40 shadow maps using sampled and interpolated visibility values was more interesting than a discontinuity meshed-based shadow computation. However, in his implementation, the placement of shadow maps was left to the user, and the accuracy in the shadow maps themselves could not be adjusted.

The essential drawback of all the above methods is that they do not provide the ability to smoothly trade computation time for accuracy. This makes them difficult to use in a hierarchical setting [Drettakis and Sillion 1996].

The idea of multi-resolution shadows was proposed by Sillion *et al.* [Sillion and Drettakis 1995] to modulate the accuracy during the computation of the shadows by considering clusters as semi-transparent blockers of various sizes. This method therefore proposes a relatively smooth time/accuracy tradeoff. An other advantage over Zatz's method is the relatively short computation time of approximate shadows. On the other hand, because the shadows are still represented as radiosity mesh elements, their accuracy cannot be pushed towards visually accurate images without a high computation time.

2.2 Using textures to represent radiosity

Because textures can represent very complex bidimensional functions in quite a compact data structure, and are easily handled using graphics hardware, storing radiosity into textures has been used in the context of many different optimizations in radiosity methods:

Gershbein proposed to represent complex emission maps as textures [Gershbein et al. 1994], in addition to their more classical application to reflectance maps, while integrating this representation in a hierarchical (wavelet) system. His results show that the visual complexity of a scene could be pushed to a very high level without the usual cost of a purely geometric representation.

In [Heckbert 1990], Heckbert proposes to use adaptive textures to store radiosity, thus making an economic representation of the illumination produced by particle tracing algorithm. His implementation is quite similar to a progressive radiosity algorithm in the sense that light is stored in adaptive data structures on all surfaces of the scene. However it does not benefit from texture mapping to display the solution or compute light interaction. Textures are only an economical way of storing radiosity, which is extracted from the textures at rendering time using a classical ray-tracer.

Since radiosity textures can be obtained very efficiently using rendering engines to project shadows and illuminate objects in the scene, some methods use graphics hardware to directly compute the radiosity contribution of an object to an other [Heckbert and Herf 1997; Keller 1997]. In [Heckbert and Herf 1997], for instance, illumination textures including soft shadows are obtained by averaging images of the receiver from several points on the light source. Although it converges to an exact soft shadow, this method requires averaging a large number of images to get an artifact free solution.

Radiosity has also been represented as textures in some applications [Myszkowski and Kunii 1994; Moller 1996; Bastos et al. 1997] in order to increase the performance of rendering the solution. Approaches like [Myszkowski and Kunii 1994] have shown that replacing large meshed polygons by a single texture opens the way to real time walkthroughs in scenes with arbitrarily complex illumination solutions. The textures containing radiosity were formed from the final radiosity solution but did not contribute to its computation. Conversely, Granier *et al.* [Granier and Drettakis 1999] replace portions of the mesh subdivision by textures during the calculation of direct lighting, in order to decrease memory requirements.

Examining the state of the art in these classes of methods, it rarely appears that textures have been used in hierarchical radiosity as an auxiliary way of storing partial illumination contributions for which they would provide an advantage over mesh elements. This might be due to the fact that hierarchical radiosity incorporates a smooth adaptive refinement scheme that does not appear to be compatible with the fixed cost of pointwisely computing a radiosity texture.

One existing method however has several common points with ours: Martin *et al.*[Martin et al. 1998] propose a two-pass method in which a hierarchical radiosity solution is first computed with limited link refinement. During the second pass, polygons that receive links with unsufficient accuracy due to rapid variation of the illumination (shadows mainly) are equiped with a texture that contains these contributions, computed using graphics hardware. Like in our method, textures are used to store radiosity maps, in cases where mesh elements do poorly, based on a classification of the links. However, several differences exist between this method and our algorithm which are pointed out in the next section.

2.3 Position of our approach

In our algorithm, radiosity contributions along links are represented as either textures or uniform mesh elements, depending on the particular characteristics of each transfer configuration. Our method is related to shadow mask-based algorithms in the sense that each "texture" representation of radiosity is composed of a shadow mask plus sparse direct illumination values, but here the complete textures (*e.g* shadows masks modulated by the direct illumination values) are used to compute energy transfers and to form the solution images.

In contrast to Martin *et al.*'s approach [Martin et al. 1998], our method belongs to the category of hierarchical radiosity methods with clustering. Radiosity textures are integrated into the hierarchical radiosity computation, rather than used as a way of increasing the accuracy during a second pass. Textures indeed are attached to links, and therefore do not merge into a single texture per polygon, thus allowing the refinement of any contribution that arrives on an element, whatever its nature, during the simulation. Another consequence is that texture-based and mesh-based representations can contribute to the illumination at multiple instances and various levels of the hierarchy for the same location of the scene. Radiosity textures can virtually serve as all kinds of links, not only those coming from light sources. As a consequence the emitter of a link can itself have part of its radiosity represented as textures.

Another difference to Martin *et al.*'s approach is that our results are view-independant. Finally, in this paper we also fully treat the various aspects related to the coexistence of radiosity textures and piecewise uniform mesh elements when displaying the solution, such as achieving a correct smooth-shaded display of the solution.

Our shadow masks are obtained with a fast computation method [Soler and Sillion 1998b], allowing them to replace subdivision into hierarchical mesh elements at a comparable cost. Therefore we have designed an algorithm for deciding for each configuration whether it is worth computing a radiosity texture. Furthermore, the convolution method for creating shadow masks incorporates an error control scheme that allows to compute shadow masks at different accuracy levels and computation costs. We have integrated this refinement algorithm and the hierarchical refinement already provided by hierarchical radiosity into a coherent global refiner.

The efficiency of the convolution method for computing approximate shadow masks drastically speeds up the simulation, which allows us to present results on complex scenes with up to 70 000 polygons in short computation time.

2.3.0.1 . The present paper is an extension of a paper [Soler and Sillion 1998a] presented at the Eurographics Workshop on Rendering in 1998. Whereas the first publication contained the basic principles of the method described here, the present paper adds several contributions: more detail is provided about the refinement process, link classification, error control integration across different types of transport representations, automatic parameter selection and display issues. More detailed results are presented with an extensive study of the influence of refinement parameters. We also include a presentation of the convolution method, based on geometric considerations, that is more comprehensive than in the original publication [Soler and Sillion 1998b].

3. COMPUTING SOFT SHADOW MASKS USING A CONVOLUTION OPERATION

In this section we briefly review the convolution technique for computing soft shadow masks introduced in [Soler and Sillion 1998b]. We only outline the important ideas to understand this technique and refer the reader to this reference for more complete technical information.

3.1 Short description of the convolution method

Consider a uniform source S of intensity E lighting a receiver through an obstacle, as shown in Figure 1. The irradiance at each point y of the receiver can be expressed as:

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

Where θ and θ' are incident angles of the ray $x \rightarrow y$ on the surfaces of the source and the receiver, r is the distance from x to y, and $v(x,y)$ is one if x sees y through the obstacle and zero otherwise. This expression of the irradiance is usually approximated by the following expression, which assumes a low correlation between the visibility and the radiosity kernel [Soler 1998]:

$$I(y) = E \underbrace{\int_S \frac{\cos\theta\cos\theta'}{\pi r^2} dx}_{F_S(y)} \underbrace{\int_S v(x,y)dx}_{V(y)}$$

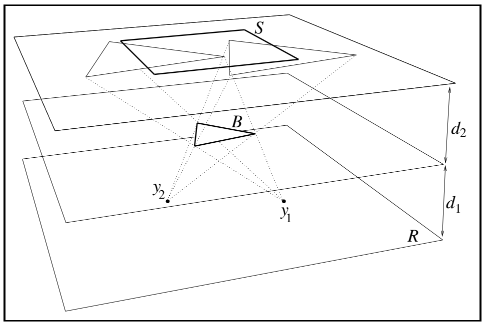


Fig. 1. *Particular configuration of planar and parallel source blocker and receiver. In such a configuration, the visible part of the source through the blocker from any point* y *on the receiver is expressed as a convolution. The justification of that property is that in the planar-parallel case the projection of the blocker onto the source plane simply translates as* y *moves into the receiver plane.*

The visibility term $V(y)$ measures the proportion of the source that is visible from point y on the receiver. It can be measured as the area of the source that is not masked by the projection of the blocker from y onto the source.

Suppose, as shown in Figure 1, that all three elements are planar and lie in three parallel planes. In that particular case, simple geometric considerations show that this projection only translates in the source plane as y moves into the receiver plane, hence the part of the source that is not masked by the projection of the blocker can be expressed as a convolution between an image of the source and a scaled image of the blocker.

The convolution method directly uses this property, and basically consists in the following three steps:

(1) Sample an image of the blocker and an image of the source by rendering offscreen these objects using frustums of adequate size;
(2) compute the convolution of the two images using fast Fourier transforms (FFT) to obtain the shadow mask;
(3) use the convolution image as a texture on the receiver, modulated by analytically computed direct illumination color values.

In real life, sources, blocker and receivers are rarely planar and parallel. The illumination on the receiver then is no longer a convolution, because the projection of the blocker onto the source not only translates when y moves into the receiver, but also changes its shape and topology. In that case however, the convolution method can be used to compute an approximation of the shadow mask on the receiver:

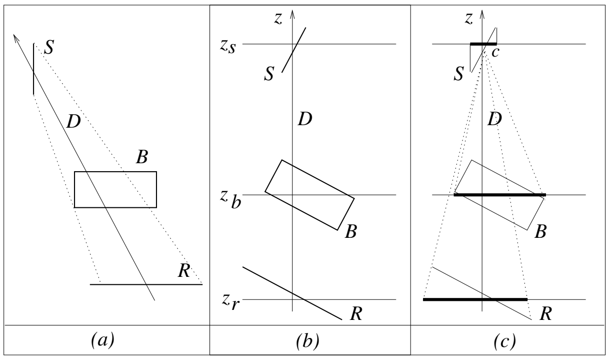


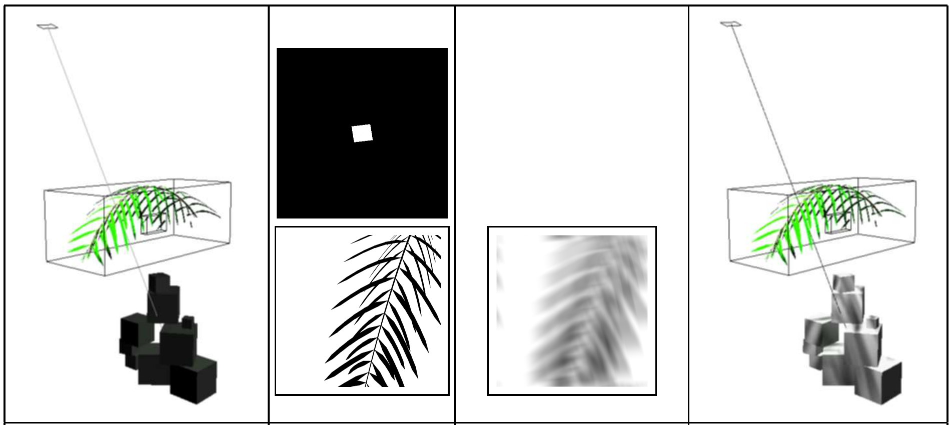
Fig. 2. *In the general case, the percentage of occlusion can no longer be expressed as a convolution. We build virtual planar and parallel elements for which the convolution applies using projections from the center of the source. The resulting shadow can be shown to be a reasonable approximation of the original one.*

The source, blocker and receiver are replaced by planar and parallel virtual elements for which the convolution still produces an exact visibility map. As shown in Figure 2, these elements are obtained by projecting the actual ones from the center of the source, onto planes at arbitrary altitudes z_s,z_b and z_r. Although a complex analysis of the error for each particular configuration could lead to find the values of these three parameters that minimize the error, we prefer to choose a more simple heuristic: the extension of the source, blocker and receiver along the z axis is measured and the middle of each interval is taken. These virtual elements are not explicitly computed, but their corresponding images functions are directly sampled by rendering offscreen the real elements using the same projections. The shadow mask is finally projected onto the real receiver where it is modulated at display time with direct illumination color values. This computation of the shadow is indeed an approximation because it implicitly replaces the blocker by a planar blocker, thus limiting the amplitude of shadow sharpness in the shadow mask.

The direct illumination color values necessary to modulate the shadow textures are computed using analytical form factor formulas [Baum et al. 1989] and stored in a *display mesh*, which is basically a coarse collection of uniformly sampled values on the receiver.

3.1.0.2 . In addition to standard situations encountered in lighting simulation, such as the source-receiver-blocker configuration used above, the convolution method applies to different extended cases with hardly any modification: for non-uniform sources, such as illuminated or textured patches as well as entire clusters, the binary source image is changed into a grey-level image still computed by rendering the source offscreen, and the convolution approximation still holds. When the receiver is not a single polygon but a cluster of surfaces, a single texture is shared between all surfaces, each one being equipped with its own *display mesh*. Finally, the method directly applies to directional sources defined by a set of illumination directions [Soler 1998].

Figure 3 shows a simple example of applying the convolution method to compute the illumination in the general case.



(a) Configuration	(b) Object images	(c) Convolution	(d) Texture

Fig. 3. *Three basic steps of the convolution method: To compute the shadow cast by an extended light source on a receiver (here a cluster of cubes) through a blocker (a) we first compute offscreen images of the source and blocker (b), then compute the convolution of these images (c). The result is used as a texture to modulate illumination onto the receiver (d)*

3.2 Error control

Replacing volumes by planar elements produces approximate shadows. Looking more carefully, one can see that shadows cast in the planar and parallel configuration have a degree of sharpness that is uniform across the receiver, which is not true in the general configuration: objects closer to the receiver should project sharper shadows and those closer to the source project smoother shadows. In the general case, a shadow mask of non adequate sharpness is thus projected on the receiver. The error can thus be estimated by predicting the variation of the expected shadow sharpness on the receiver [Soler and Sillion 1998b].

This characterization of the error is not exhaustive, because it does not account for changes in the topology of the blocker viewed from various points on the source, but it corresponds to the far most visible part of the error.

To control this error, we have proposed a simple algorithm based on the computation of several convolutions [Soler and Sillion 1998b]: the blocker is divided into multiple parts to diminish its extent along the main direction of light propagation. A shadow mask is computed for each part, and all shadow masks are finally combined to form a shadow mask having as many levels of sharpness as there are convolutions computed, and corresponding to the entire blocker. The division of the blocker into parts is facilitated by its expression as a hierarchy of clusters.

3.3 Motivation for incorporating the convolution method into a hierarchical radiosity algorithm

The convolution method presented above has several interesting features:

First of all it produces shadow masks for very complex blockers in a short computation time: typical time for a shadow texture of resolution 256×256 is less than $400ms$ on both *SGI* O_2 and $Onyx^2$ computers.

Secondly, it benefits from the presence of specific hardware: computation of the images corresponding to complex blockers are greatly accelerated on hardware-accelerated graphics cards. Besides, fast Fourier transforms can be performed efficiently on commonly found DSP chips.

The convolution method also incorporates an error-control algorithm, that is based on a hierarchical representation of the obstacles. We will see that this particular feature will allow the convolution method to integrate smoothly into the traditional refinement process of a hierarchical radiosity algorithm with clustering.

Finally, the method produces visually pleasant shadow masks, especially compared to the methods based on discrete sampling, or shadows simulated by uniform mesh elements (like in most hierarchical radiosity systems), that are associated to various discontinuities. Of course shadow masks computed with the convolution method are approximations of the exact shadows, but the associated error is hardly perceptible to the human eye, making this technique particularly suitable for image synthesis. The error in the shadow masks concerns two aspects: the topology of the discontinuity mesh in the shadow mask and the variations of the shadow in penumbra regions shared by different sub-blockers in case of blocker subdivision. Both of them are not visually noticeable, except for built-in special test cases.

In the next section, we discuss the main avenues of integrating the convolution method into a hierarchical radiosity algorithm with clustering.

4. PRINCIPLE OF USING RADIOSITY TEXTURES IN A HIERARCHICAL RADIOSITY ALGORITHM

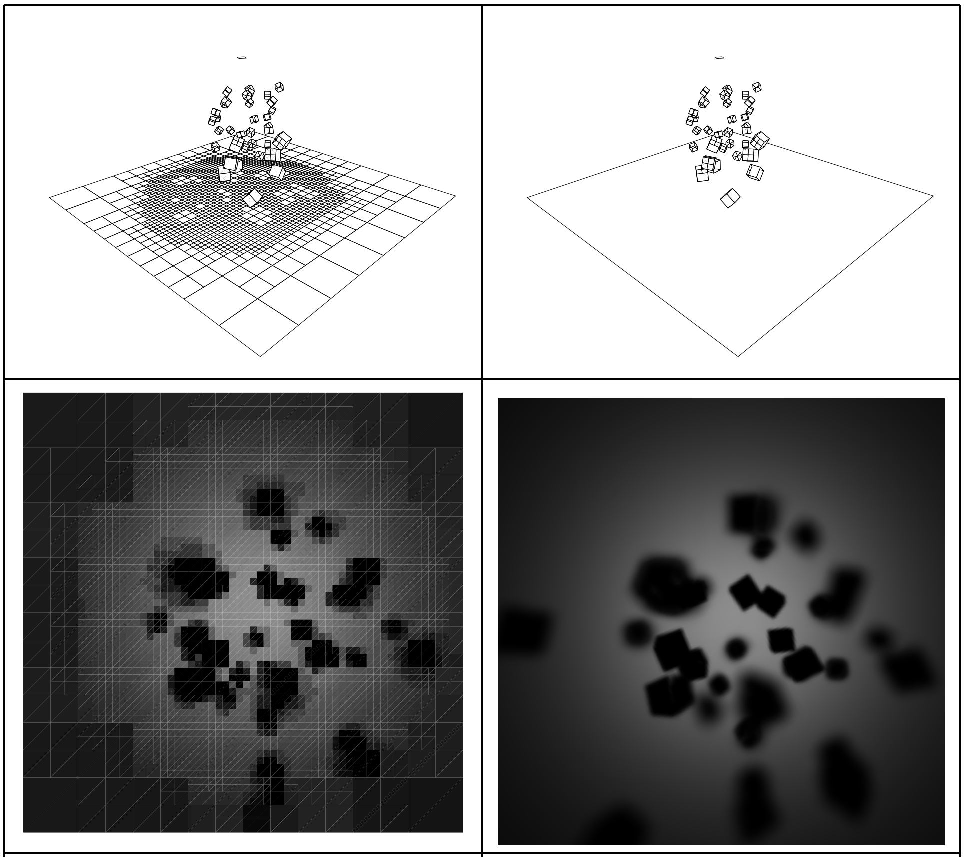
Considering the potential benefits of using the convolution method for the calculation of shadow maps, we would like to emphasize the power of the technique in the context of hierarchical radiosity.

As said in the introduction, most of the refinement in a hierarchical radiosity calculation is performed to represent shadow variations and this refinement goes far deeper than the level simply required by the computation of energy balance. In case of partial visibility, the number of mesh elements as well as the number of links increases rapidly and slows down the computation.

The idea behind our algorithm is thus to compute shadows using the convolution method previously described, whenever appropriate, still using standard energy transfer calculations in other situations. In other words we choose between two different representations and methods of computation of radiosity for each energy transfer.

In order to do this, we add a new kind of link to the standard hierarchical radiosity links, for which the energy transfer is represented by a radiosity texture, *e.g* the combination of a shadow mask computed using a convolution and a display mesh containing unoccluded illumination information. We call such links *convolution links*. An example of such a substitution is presented in Figure 4.

In our system, radiosity will consequently be represented by two different forms: textures and mesh based elements. This means that the different phases of energy propagation (*gather* and *push/pull*) as well as the algorithm used to display the solution must account for this variety. New gather and push/pull algorithms will be detailed in Section 5 and display will be treated in Section 7.



Standard link hierarchy of $2,000$ links (3.2 sec)	Single convolution link (< 0.5 sec)

Fig. 4. *Example of successfully using a convolution link to compute the illumination on the receiver. A hierarchy of about* $2,000$ *standard hierarchical radiosity links has been replaced by a unique convolution link. Computation time is shorter and the result visually more pleasant because it is free of discontinuity or interpolation artifacts.*

Error control is a very important feature of hierarchical radiosity algorithms because it allows to continuously trade accuracy for speed during the computation. In classical hierarchical radiosity and clustering algorithm, this is achieved by deciding at which level of the hierarchy the links are established. In our system, more refinement alternatives are offered to the refiner, since it can also refine the blockers when computing a convolution texture. Moreover, the refiner also has to choose which kind of link to use for each transfer configuration and must estimate the adequate link parameters, according to the memory and CPU cost associated to each kind of computation. This is explained in Section 6.

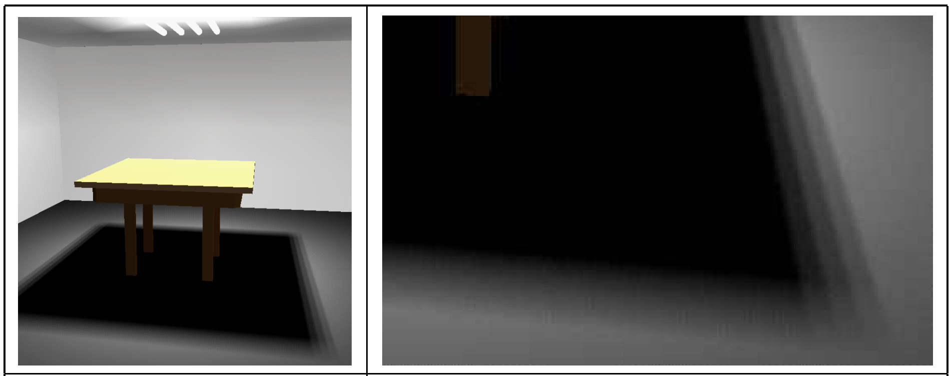
5. PROPAGATION OF LIGHT ENERGY

5.1 Gather

In the *gather* phase of a hierarchical radiosity algorithm energy is usually propagated along links that have been established during the *refine* phase of the algorithm.

For standard links, the gathering operation consists of multiplying a single radiosity value of the source element by the form factor associated to the link, which yields the irradiance contribution of the emitter. For such links, the radiosity value of the emitter is the sum of its uniform radiosity values and the average of its radiosity textures. The average values of radiosity textures is computed in the push/pull phase of the algorithm, as explained in the next section.

For a convolution link, the energy transfer is computed using a convolution, as described in Section 3. In order to obtain the source image, the source element (which can be a polygon or a cluster) is rendered offscreen with all its radiosity textures, the same way it is rendered on screen for displaying the solution. Proceeding this way accounts for the correlation between the source radiosity function, which may not be uniform, and the visibility through the blocker. This allows to automatically simulate interesting phenomena, like the stepping effects due to parallel elongated lamps also parallel to the edge of a table, in the shadow cast on the floor, as illustrated on Figure 5.



(a) configuration | (b) close-up view of bottom-right corner

Fig. 5. Example of a side effect of the high correlation between the source and blocker shapes: in the configuration shown in (a), the neon tubes are parallel to one of the edges of the table. A single convolution is sufficient to capture the stepping effect in the shadow on the floor, in the direction perpendicular to the axis of the neon tubes (See (b)).

For convolution links, a complete gathering operation would ideally consist of computing the soft shadow texture and modulating it by the unoccluded illumination values. For practical reasons (especially concerning the display) we have chosen to store soft shadow textures –equipped with illumination values– only at the leaves of the hierarchy. The reason for this is that the direct illumination values strongly depend on the geometry of the leaf itself, rather than on the parent to which the convolution link normally points, and which in most cases is a cluster, or a large polygon.

Therefore, the soft shadow textures computed during the gather step are just stored in an *soft shadow texture list* at the current hierarchical level, waiting to be pushed down to the leaves and equipped with adequate unoccluded illumination values during the push/pull operation.

Note that, just as in a standard clustering algorithm, the intra-cluster visibility of the receiver is not accounted for. In Section 7.5 we discuss how to simulate this effect when rendering, but for the radiative exchange, we consider this as a regular approximation due to the high level the link has been established at, just like in standard algorithms based on clustering.

5.2 Push/Pull

As in classical hierarchical radiosity algorithms, the push/pull phase is a depth first traversal of the hierarchy, that both computes radiosity values from irradiance information at leaves of the hierarchy, and ensures that the multi-level representation of radiosity is consistent. This means that the value stored at each level is the average value of the radiosities values of lower levels.

During push/pull, uniform irradiance values from standard links are added together down to the leaves where they are multiplied by the local reflectance value. At the same time, soft shadow masks stored during the gathering phase are collected on a *texture stack* and pushed down to the leaves of the hierarchy.

Once at a leaf, a proper set of modulation values is computed from reflectance and unoccluded form factor values for each of the collected soft shadow masks. Practically, these values are computed at the vertices of a raw *display mesh*, the size of which depends on the geometry of the leaf. At each of these points we also store adequate texture coordinates in the plane of the soft shadow texture, to be used when rendering. Thus the top-down phase of the push/pull operation essentially consists of transforming the shadow masks into complete soft-shadow textures at the leaf nodes of the hierarchy.

Using the same direct illumination samples computed for display, we also compute an approximate mean value of the radiosity contribution of the texture on the leaf. This value is added to the uniform radiosity value due to standard links and pulled up in the hierarchy in a classical way.

Since we only store the complete soft shadow textures on leaves of the hierarchy, we must also push down the complete soft shadow textures possibly existing as a result of a preceding push/pull pass on elements that are no longer leaves of the hierarchy. In that case, the shadow texture is separated from its display mesh, and the texture itself is added to the stack of pushed down textures.

Using this algorithm, the array containing the convolution image is shared between all leaves of an element that receives a convolution link. Only display meshes are specific to each leaf, and they only consist in a sparse array of samples. The entire push/pull process is summarized in Figure 6.

6. REFINEMENT

The role of a refinement algorithm is to decide at which level of the hierarchy energy transfer links must be established, thereby controlling the accuracy of the simulation. Since our system uses two different kinds of links the refiner must also choose the right kind of link for each transfer configuration, considering both the cost and the relevance of the two different methods for any particular situation. In addition, when it has decided to use a convolution link, the refiner must compute the number of convolutions associated to the link in order to achieve a given precision in the transfer computed. This adds a third possibility to the usual refinement of the emitter and receiver: refinement of the obstacles.

In summary, the refiner must answer the following questions in order to decide a link is to be established between a receiver A_i and an emitter A_j:

Is it possible to estimate the energy transfer from A_j to A_i with sufficient accuracy
—with a standard link ?
—with a convolution link ?

If so, what kind of link is suited best to the current situation ?

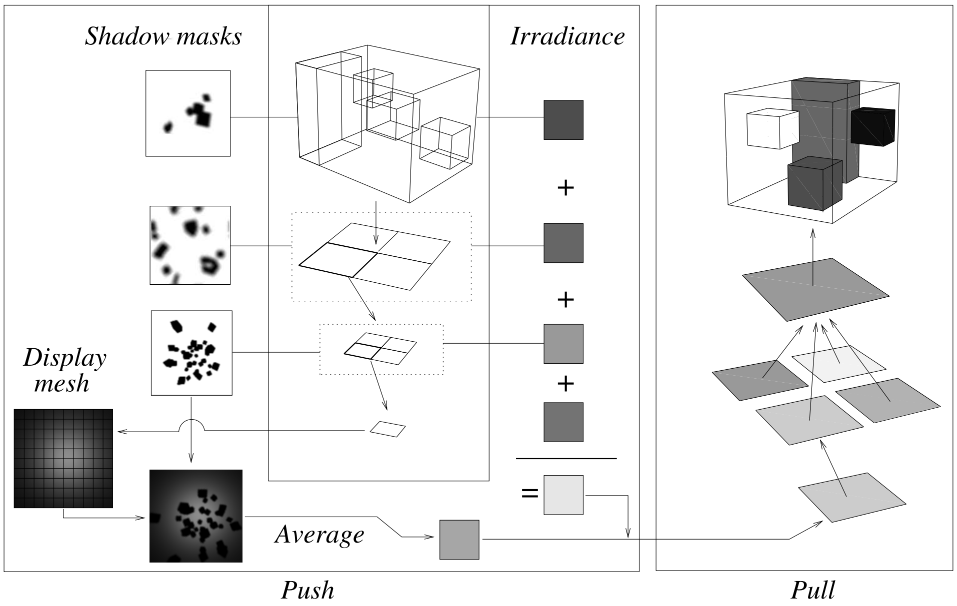


Fig. 6. *Illustration of the push/pull algorithm. Irradiance contributions are collected down to the leaves of the hierarchy. At this point, display meshes are computed for each shadow mask, and the average value of all radiosity textures is added to the radiosity pulled up. Considering multiple shadow masks at subsequent levels of the hierarchy is common for scenes with multiple light sources.*

If not, is it possible to increase the accuracy
—by subdividing the receiver ?
—by subdividing the emitter ?
—by subdividing the obstacle, using a convolution link ?

Considering the speed of the convolution method, our strategy is to use convolution links whenever possible. To establish a link, we thus start by checking if a convolution link can be used, then try a standard link, and finally refine the current transfer. We explain the criteria used to identify convolution sites in Section 6.1. In Section 6.2 we show how to automatically set the various parameters used in the computation of radiosity textures. In Section 6.3 we describe the refiner we use for energy based refinement, and we explain the order of operations in Section 6.4.

6.1 Automatic identification of convolution sites

Computing the energy transfer along a convolution link requires fast Fourier transforms, and consequently has a certain cost. We therefore wish to limit this operation to cases where it is really useful: we avoid convolution links when the receiver is very small, when no blockers are present and when the shadow produced will be too smooth, in which case smooth shaded standard piecewise uniform elements would do fine.

6.1.1 *Absolute minimum receiver size.* Considering the cost of storing and displaying a texture, we want to limit the number of convolution links. As we are computing a global view-independent solution, we can not afford to assign textures to polygons depending on their appearent size, in a particular view of the scene. We therefore choose an absolute minimum size for a receiver in proportion to the size of the scene. This proportion must of course be adapted to the kind of scene we are treating (5 percent for the scene in Figure 14, but much less if we want images of a room in a very large building).

A more complete approach could be to compute textures in order of decreasing importance (*e.g* absolute size) within a limited amount of texture memory, but this hardly comforms to the spirit of the refinement algorithm, that performs a depth first (instead of breadth first) traversal of the link hierarchy.

6.1.2 *Presence of blockers.* The convolution method is not interesting in cases where no blockers cast shadows on the receiver, although it would work with no modification in such a configuration. To avoid such situations, we determine the set of blockers between the source and the receiver as a list of clusters using a *shaft-culling* method [Haines and Wallace 1991]. This allows to discard clusters that don't intersect the set of rays between the source and the receiver in a matter of a few arithmetic operations.

The algorithm for blocker selection starts by considering the root cluster of the hierarchy as a potential blocker since it is sure to intersect the shaft between the emitter and the receiver. It then proceeds by recursively considering the children of the current cluster until they contain neither the emitter nor the receiver. The recursion does not stop until more than one child of the considered cluster intersects the shaft (see pseudo-code in Figure 7). The algorithm produces a list of potential occluders that are finally grouped into a single temporary cluster, in order to simplify the manipulation of all obstacles.

Note that this algorithm can select blockers that do not effectively produce shadows on the receiver because the shaft-culling test is a conservative one, but this does not cause any problem to the convolution algorithm.

```
ListOfClusters SelectBlockers (receiver p, emitter q, cluster B,shaft s)
    if p = B or q=B
        return NULL
    if B→Contains(q) or B→Contains(p)
        ListOfClusters L
        For All Children b of B
            L→AddToList(SelectBlockers(p,q,b,s) )
        return L
    integer n = 0
    cluster selected_child
    For All Child b Of B
        if s→Intersects(b)
            selected_child = b
            n = n+1
    if n = 0
        return NULL
    if n = 1
        return SelectBlockers(p,q,selected_child,s)
    return B
```

Fig. 7. *Algorithm for the selection of all potential blockers between a source and a receiver, using shaft-culling. The function is called on the root of the hierarchy as* **SelectBlockers**(S,R,Root,shaft(S,R)).

6.1.3 *Estimation of shadow sharpness.* Once blockers have been located we estimate the sharpness of the shadows they cast on the receiver and compare it to the size of polygons in the receiver itself. The idea is that if the variation of the illumination on the receiver is smooth enough to be represented by piecewise uniform mesh elements, we do not need a convolution link.

We estimate the gradient of the illumination on the receiver by dividing an estimate B_{max} of the maximum direct illumination value on the receiver by an estimate P_{min} of the minimum size of penumbra regions cast by the blockers. As illustrated in Figure 8a, the size of penumbra regions cast by a planar blocker parallel to the source plane can be estimated using bounds on the altitudes of the projection of the source, blocker and receiver along a direction D:

$$P_{min} = \frac{1}{\cos\gamma} \frac{z'_{min} - z_{max}}{z_s - z'_{min}} \tag{1}$$

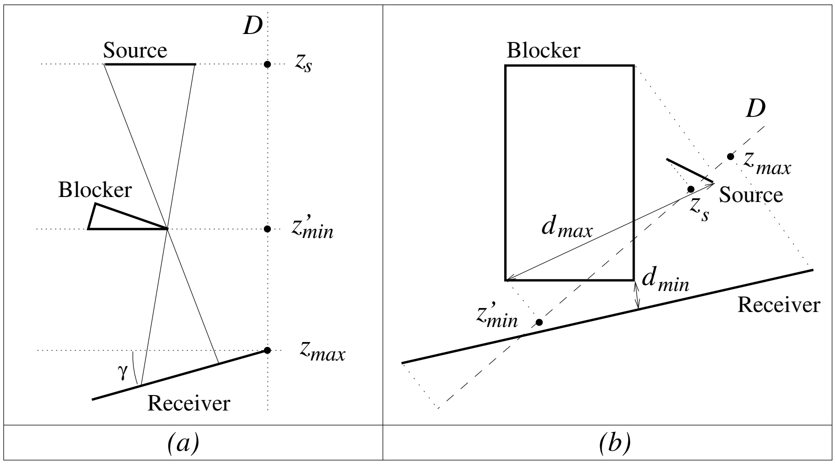


Fig. 8. *Estimation of the sharpness of the shadow cast by an occluder on a receiver. (a) in this ideal case, the penumbra size can be estimated using altitudes of the projections of the different components. (b) in this case, a lower bound based on the maximum distance d_{max} between points in the source and blocker and the distance d_{min} between the receiver and the blocker is used.*

However, as shown in Figure 8b, some configurations do not allow a relevant estimation of shadow sharpness using this simple scheme because the source, blocker and receiver projections overlap. In such cases we use a more time consuming lower bound computed from the relative maximum distances between points in the three elements:

$$P_{min} = \frac{d_{min}}{d_{max}} \tag{2}$$

Since the bounding box of the blocker can overlap the receiver or the source, this estimate is not always significant. Our algorithm for shadow sharpness estimation thus consists of a depth-first traversal of the blocker hierarchy, collecting values given by Equations (1) and (2), only for blockers that cut the shaft between the source and the receiver, until the values found for the sharpness of the shadow are relevant.

Once computed, the lower bound P_{min} on the size of penumbra regions in the shadows gives a bound on the gradient of the illumination on the receiver, using the maximum value B_{max} of the unoccluded illumination on the receiver:

$$\Delta_{max} = \frac{B_{max}}{P_{min}} \tag{3}$$

For a given receiver, that may be a single polygon as well as a cluster, we pre-compute the maximum diameter T_{max} of its polygons. A radiosity texture will be of interest if the illumination varies more rapidly than that of a smooth shaded polygon for which illumination values range from 0 to 1, *e.g* if

$$T_{max}\Delta_{max} \geq 1$$

Some objects in a scene however, may be constituted from a collection of small polygons connected to each other (a faceted surface for instance). For such objects, the diameter T_{max} used should rather be the diameter of the entire object. We account for such possibilities by forcing the scene cluster hierarchy to respect the connectivity between polygons: each cluster must know if it contains polygons and clusters forming a single object.

6.2 Automatic selection of shadow texture parameters

6.2.1 *Texture resolution.* The resolution $n \times n$ of the texture must be sufficiently large to accurately represent the variations of the illumination including shadows on the entire receiver, and at the same time be as small as possible to save computation time during fast Fourier transforms. For practical reasons we also require texture sizes to be powers of 2. Using Expression (3), the adequate size n is the smallest power of two that verifies:

$$\frac{T_{max}}{n}\Delta_{max} \leq 1.0$$

This basically illustrates the fact that the illumination does not vary more than 1.0 per texture pixel on the receiver.

Choosing N this way may produce arbitrarily large texture sizes, especially when the blocker touches the receiver. We consequently clamp the value obtained depending on the size of the receiver and at most to the maximum resolution of 512×512 which is highly sufficient even for the largest polygons of a scene.

6.2.2 *Size of the display mesh.* As stated in Section 3.1, the display mesh contains direct illumination values from the source that are used to modulate the texture when rendering. Each of its vertices has also been supplied with precomputed texture coordinates for the projected shadow texture. These coordinates do not vary linearly, although **OpenGL** linearly interpolates between the provided samples. The resolution of the display mesh must therefore be adapted to both the gradient of the radiosity kernel and the variations of texture coordinates across the receiver, but still be as coarse as possible. The correct way to achive this would be to build an adaptive mesh based on these variations. In our current implementation, we have used a simpler alternative consisting of a uniform mesh, with a sufficiently fine resolution that is simply determined in proportion to the size of the receiver. As an example, for the largest polygons such as the walls in the fourth scene called 'Bar' in Section 8, we use meshes of 20×20 samples.

6.2.3 *Computation of blocker refinement depth.* As explained in Section 3.2, the error in the shadow mask computed using the convolution method can be controlled according to a perceptually based error criterion that measures the expected variation of the shadow sharpness across the receiver. More accurate shadow masks are obtained by computing separate shadow masks for separate parts of the blocker and finally combining them into a suitable shadow mask.

The goal of the present task is to determine a (preferably small) partition of the set of blockers selected for the shadow computation, that leads to a final error in the shadow map that is less than a user-defined error threshold. We will see in Section 6.4 that the number of sub-blockers used directly determines the number of convolutions to perform and thus the computation time for the shadow mask.

The blockers are selected as follows: the hierarchy obtained by the blocker selection process is traversed depth-first until the encountered clusters satisfy the error criteria based on the sharpness of the shadow, while still intersecting the shaft between the source and the receiver. At the same time, the algorithm keeps trace of the number of blockers selected as well as the maximum error reached. Depending on the hierarchy, it is not always possible to reach an arbitrary accuracy value, because we only split clusters and not surfaces. Upon return, the algorithm thus gives the best error value reached and the number of blockers selected. The traversal stops if these numbers exceed their maximum permitted values.

6.3 Energy based refinement

Our criterion for error estimation along standard links is based on a method similar to that proposed by Lischinski [Lischinski et al. 1994], using non conservative bounds on the radiosity transfered along the link. Upper and lower bounds on the radiosity are estimated from values sampled in the radiosity textures for elements receiving convolution links, and from sampling values also used to compute the form factor, for standard links.

For a more comprehensive presentation, we have separated standard refinement from convolution links refinement. In practice the two methods exchange important information. In particular, the refinement of standard links benefits from the computations performed before discarding the possibility of a convolution link.

In our algorithm, visibility configurations are classified into two categories: full visibility and partial visibility. When refining a link, child links will require new occlusion tests to be performed only in the case of partial visibility. Finally, in the case of partial visibility, we save the occluders for a given link in form of a cluster. This cluster will be used as a starting point to find occluders for child links, in case of further refinement.

After trying to select blockers for a convolution link, we know in a conservative manner if the visibility is full. In that case, although we give up the computation of convolution link information, we know that the standard link won't have to perform visibility tests, and refined child links will keep this property.

Finally, when trying to establish a standard link, the estimate of the illumination gradient computed during the tests for convolution links gives us the minimum size of useful uniform elements. In fact, in most cases of partial visibility, a classical hierarchical radiosity refiner would refine until the element sizes reach the minimum area allowed by the user, because the refiner has no information about the variation of the illumination in the shadow regions. This refinement is useless in cases where a large source casts a very smooth shadow.

This important feature, missing in most existing refiners, greatly increases the efficiency of the algorithm.

6.4 Refinement algorithm

In the previous sections, we have detailed the different operations required to decide which kind of link is needed and at which hierarchical level to use it. All these operations take place in the general process of refinement that we explain here. We consider an emitter and a receiver element, that may be patches or clusters (An overview of the refinement algorithm can be seen in Figure 9).

We first check whether the amount of energy traveling from the emitter to the receiver is greater than a user-defined minimum threshold ε. If not, a standard link is established. If it is, the refiner checks whether the conditions required for a convolution link are fulfilled, at the same time collecting the information described in Section 6.3.

If it is not possible to establish a convolution link, the refiner checks if a standard link would be accurate enough, with respect to the required accuracy ε_{max}. If not, it may recursively call the whole refinement algorithm on the children of the emitter and receiver.

If the configuration allows the establishment of a convolution link, the refiner estimates the number N of convolutions required to compute a shadow texture respecting the user-defined accuracy E_{max}. Denoting by $n \times n$ the resolution of the texture, the time required by a single convolution will be that of three fast Fourier transform, *e.g*

$$t = Cn^2 \log(n)^2$$

When performing several convolutions for multiple sub-blockers, we compute the fast Fourier transform of the image of the source only once and adapt the images of the different blockers accordingly [Soler and Sillion 1998b]. The number of FFTs necessary for N convolutions is consequently:

$$p = 2 + 2N$$

Hence the approximate computation time for the convolution link, neglecting the computation of the source and blocker images:

$$t = C(2 + 2N)n^2 \log(n)^2$$

In this expression, C is a constant dependent on the machine, and determined once by the algorithm. If the required duration t is greater than the maximum allowed duration t_{max} for a convolution link, we give up its computation, and try standard refinement instead.

In summary, the refinement algorithm is controlled by six independent parameters:

Parameter	**unit**	**role**
E_{max}	m	Maximum error on convolution links
ε_{max}	$W.m^{-2}$	Maximum error on standard links
A_{min}	m^2	Absolute minimum size of a potential receiver for a convolution link
α_{min}	m^2	Minimum size of a receiver for a standard link
ε	$W.m^{-2}$	Minimum value for energy transfered along a link.
t_{max}	sec	Maximum time allowed to the computation of a convolution link.

Since the error for convolution links is measured as the maximum deviation of the computed size of penumbra regions in comparison to the expected penumbra size in the shadow masks, the threshold E_{max} is measured in meters. Because of this, comparing the error for convolution and standard links is difficult. A discussion about the consequence of having

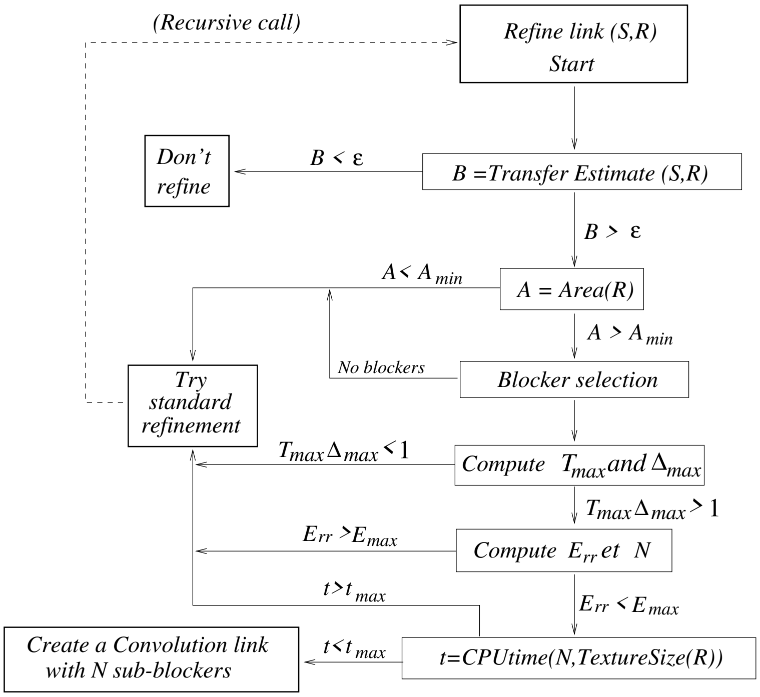


Fig. 9. *Overview of the refinement algorithm: convolution links are established after the source* S *and receiver* R *have past several tests (see the description in the text) concerning the relevance and the cost of using a convolution to simulate the energy transfer. If one of these tests fails, either a standard link is established between* S *and* R*, or a recursive call of the whole procedure on their children occurs.*

two independent error thresholds for controlling image quality is presented in Section 8.4.

7. DISPLAY

Because the radiosity textures and the uniform radiosity values are stored with the leaves of the hierarchy, the rendering algorithm simply consists of an in-depth traversal of the hierarchy, during which each leaf polygon is displayed by first rendering its uniform radiosity values and then successively blending the radiosity textures on it, using a simple addition operation.

A number of interesting questions occur during this phase of the algorithm. We explain below what problems arise in modulating textures because of large direct illumination values, how to cope with reflectance textures, and how to achieve smooth shading across mesh elements. We also discuss the possible techniques for optimizing rendering speed and how to account for intra-receiver visibility for convolution links.

7.1 Achieving a correct texture modulation

As described in Section 3, the radiosity textures are rendered using the computed shadow masks as textures, modulated by direct illumination color values sampled on the receiver.

Achieving a correct texture modulation using the **OpenGL** library is not straightforward in the case where the direct illumination component is greater than unity. Let us call $V(y)$ the value in the shadow mask and $B_0(y)$ the direct illumination component, and assume that we have:

$$V(y) < 1 \quad B_0(y) > 1 \quad \text{and} \quad V(y)B_0(y) < 1$$

In such a configuration, some **OpenGL** implementations clamp the color value $B_0(y)$ to 1 before modulation [Neider et al. 1993] and the texture fragment is rendered with color $V(y)$ instead of $V(y)B_0(y)$.

Obtaining an accurate color modulation is not only important for rendering but also for computing convolutions along links for which the source is a textured polygon.

Two slightly different methods let us correct the modulation. The first one consists of rendering the texture n times with color values divided by n, and accumulate the result on screen. To achieve a correct modulation in the resulting image, n must verify:

$$n \geq 1 + \lfloor I_{max} \rfloor$$

The second method consists of rendering the texture $n+1$ times with n defined as above, using for pass number $0 \leq p \leq n$ the actual color values reduced by p. Since the **OpenGL** implementation we use clamps input color values to $[0, 1]$ before modulation, the sum of all rendered slices matches the requested result, each slice being modulated exactly as expected. For example, if $n = 5$ and the color value at a point of the mesh is 3.14 and the texture value is c, it will be rendered as

$$v' = 3.14 \times c + 2.14 \times c + 1.14 \times c + 0.14 \times c - 0.86 \times c - 1.86 \times c$$

which produces the expected value on screen, because of the clamping artifacts:

$$\begin{aligned} v &= 1 \times c + 1 \times c + 1 \times c + 0.14 \times c + 0 \times c + 0 \times c \\ &= 3.14c \end{aligned}$$

Although these two methods have the same cost, we prefer the second one because the first method introduces color quantization through the division, and thus produces stepping effects for large modulation values, whereas the second method does not. On the other hand, the first method does not depend on the clamping strategy of the particular **OpenGL** implementation and should be prefered for multi-platform implementations.

7.2 Environments with reflectance textures

Applying our algorithm to scenes that include textures in the usual meaning (*e.g* reflectance textures) requires a few modifications: as a general rule, the values stored in patches equipped with reflectance textures are not radiosity values but irradiance values. This means that instead of uniform radiosity values we store irradiance values. The same is applied to radiosity textures for which direct illumination values in the display mesh do not account for the reflectance of the receiver. At display time, such patches with reflectance textures are first rendered with their radiosity textures (that contain irradiance in this case) and piecewise uniform irradiance values. Then we use the **OpenGL** blending capabilities to pointwisely multiply the resulting image by the reflectance texture:

```
glBlendFunc(GL_DST_COLOR, GL_ZERO)
```

The same operation is accomplished during the gather phase for convolution links, to obtain an image of an emitter having reflectance textures. Examples using our algorithm on textured scenes are presented in the results section.

7.3 Smooth shading

Rendering a piecewise uniform radiosity solution with smooth shading consists of computing for each vertex of the mesh a weighted average of the uniform radiosity values of the adjacent elements, and then rendering each mesh polygon using the values obtained for each of its vertices [Sillion and Puech 1994]. When radiosity is represented by both textures and uniform values on patches, rendering with smooth shading is a little more complex. In particular, we cannot simply render the radiosity textures over smooth shaded mesh elements, because the classical smooth shading average values at mesh vertices have a global influence on the entire adjacent patches, whereas illumination values from radiosity textures on the vertices only depend on the illumination at the corresponding points (See *Figure 10*).

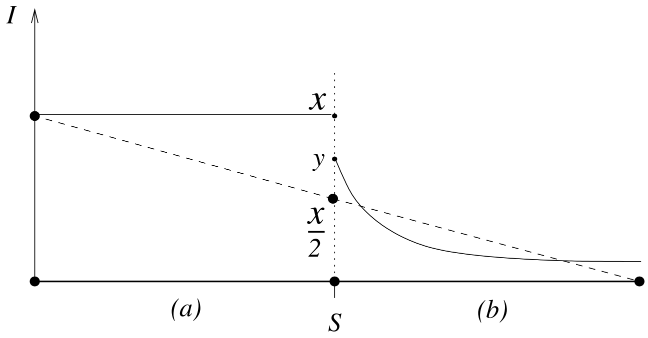


Fig. 10. *Patches (a) and (b) share a vertex S. Patch (a) only has uniform value x and no textures whereas (b) has a texture with value y at S and no uniform radiosity. Adding radiosity textures on smooth shaded piecewise uniform elements we would get the two different values $\frac{x}{2}$ and y at point S. Using our method, we get at vertex S, $v_s = \frac{x}{2} + \frac{y}{2}$ and $v_{Tex}^{max} = y$, thus $v_{ref} = \frac{x}{2} + \frac{y}{2}$. Patch (a) is rendered with $v_S = \frac{x}{2} + \frac{y}{2}$ and patch (b) is first rendered with $v_S = \frac{x}{2} - \frac{y}{2}$, which is non negative, before the texture is added to it.*

To enforce the concordance of the smooth shading illumination values at the mesh vertices and those coming from radiosity textures at these points, we define a reference value v_{ref} for each vertex of the mesh, and modify the smooth shading values of the adjacent patches so that, when blending radiosity textures, they all finally render with this reference value at the vertex. More precisely, we first compute and store the following information, at each vertex of the mesh:

—the *smooth shading* value v_s, which is the average value of uniform mesh values plus the average value of radiosity textures at this point.

—the maximum value v_{Tex}^{max} of radiosity textures for adjacent patches at this vertex.

The reference value v_{ref} for a given vertex of the mesh is then defined as the maximum of these two values:

$$v_{ref} = \max(v_s, v_{Tex}^{max})$$

We then render each mesh element M using the color v_S for vertex S:

$$v_S = v_{ref} - Textures_M(S)$$

Doing this, all adjacent patches of a given vertex finally render with the same value at this vertex. Note that the color value v_S is always non negative, because of the expression of v_{ref}. Since this technique also applies to $T - vertices$ in a constrained quadtree, we also use it to render anchored polygons, thus eliminating discontinuities that remain at such points. An example of application of this method can be seen on Figure 17 in the results section.

Unfortunately, this technique only eliminates discontinuities at mesh vertices, and not on edges shared by both radiosity textures and piecewise uniform elements. In such cases, discontinuities can be ruled out by further subdividing the piecewise uniform patches down to a point where the resulting mesh matches the display mesh of the radiosity textures, on the given edge.

7.4 Optimizing display

At equivalent resolutions, displaying a single texture is much more efficient than sending a complex mesh of uniform elements to the graphic board [Myszkowski and Kunii 1994]. For this reason, we favor the use of textures to represent radiosity. However, on most machines texture memory is limited and must be saved.

To save texture memory, we collapse into one texture several textures issued from different convolution links, but arriving at a same patch. Doing this simply requires rendering the patch radiosity textures and values to an offscreen buffer and setting the resulting image as a radiosity texture, with no shadow mask. The drawback of such an operation in that it merges the contributions of different links and thus prevents any further refinement of the patch, unless we re-compute the contributions of each link separately. However, it should also be noted that convolution links are rarely refined, because they mainly come from highly illuminated patches, such as light sources, whose radiosity does not vary from one iteration to the other. This operation is thus not only interesting for accelerating display but also for reducing the memory requirements of the light propagation algorithm [Granier and Drettakis 1999]. Converting illumination into textures also makes it easy to export the solution into a common graphics format such as VRML.

7.5 Intra-receiver visibility

We saw in Section 5.1 that our algorithm does not account for intra-receiver visibility more than other classical clustering algorithm. In a cluster receiving a link, two polygons having the same orientation will receive the same amount of light even if one is occluding the other. This approximation, that can be considered acceptable with respect to the high level at which the link has been established, slightly increases the amount of light received by some polygons in the cluster. As in other clustering algorithms, subdividing the link automatically allows some parts of the receiving cluster to act as potential occluders for the remaining parts, and thus reduces the approximation.

Nevertheless, during rendering, the lack of self shadows in clusters receiving a large amount of light, especially with shadows, can become noticeable. A proper shadowing effect can be achieved using the following three-steps method: the cluster is first rendered from the center of the emitter into the depth buffer. In a second step, the cluster is rendered from the normal view point, while supplying **OpenGL** with the previous depth image as a texture and using the **SGIX_shadow** extension of **OpenGL** to only render pixels that don't lie in the shadow region. Because this is done using the depth map as a texture, we can't render radiosity textures at the same time (as well as reflectance textures, if needed). The image in the second step is thus rendered to the *stencil buffer*, that we finally use to render

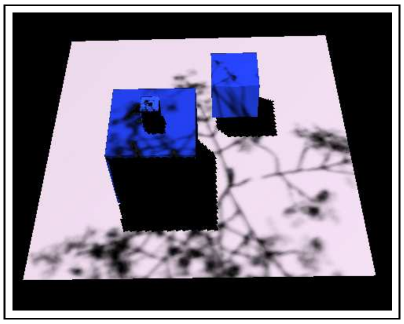
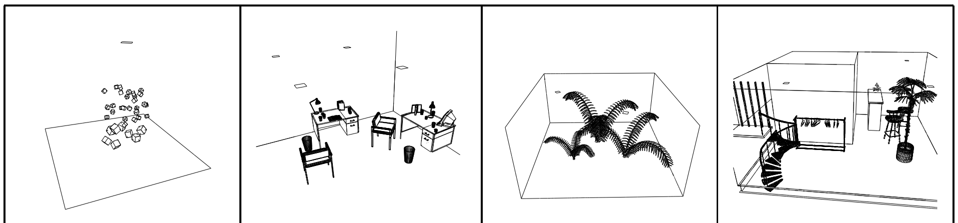


Fig. 11. *Simulating receiver self-shadowing using a 3-passes method. The receiver is the set formed by the cubes and the plane. The tree that casts the soft shadow is not represented for clarity. The scene is first rendered from the center of the source. Using the* **OpenGL** *DepthTexture extension and the previous depth view, the receiver self-shadowed image is computed, and set as a mask in the stencil buffer. The scene is finally rendered through the stencil buffer with its original textures (The shadow of a tree in this example).*

the cluster with its radiosity texture during the third and final step. The **SGIX_shadow** extension of **OpenGL** is hardware supported on *InfiniteReality* systems and rapidly handles the first two steps. Figure 11 gives an example of using such a technique. Note that the resulting self shadow is a hard shadow, and is prone to aliasing effects without adapted treatments [Reeves et al. 1987], but usually adds to image realism.

8. RESULTS

In this section we present various examples and experiments to demonstrate the efficiency of our algorithm. The four different scenes used for these tests are displayed in Figure 12. In Section 8.1 we discuss the computation time and memory cost for these scenes. In Section 8.2 we compare our results to purely energy-based refinement, and study the influence of various parameters on the algorithm performance in Section 8.3. A discussion about controling the quality of the resulting images is given in Section 8.4. Finally, we conclude in Section 8.5 with more examples showing some important features of the algorithm.



Name:	**Cubes**	**Office**	**Trees**	**Bar**
Polygons:	212	5380	22024	71517
Clusters:	46	2018	4863	21378
Sources:	1	4	2	3

Fig. 12. *Test scenes in order of increasing complexity.*

8.1 Computation times and memory cost

The table below presents the computation times and approximate memory costs for the solutions presented in Figure 13. These experiments have been done on both *Silicon Graphics* $Onyx^2$ - *Infinite Reality* and O_2 workstations. Refinement parameters have been set so as to obtain reasonable quality pictures while keeping short computation times. This implies an average number of blockers for convolution links of about 3. As a general rule, faster results can be obtained without any blocker refinement in the convolution links.

Comparing computation times for O_2 and $Onyx^2$ clearly illustrates the idea that our algorithm relies on graphics hardware to compute the shadows: the results are obtained much faster on the $Onyx$ with an average gain factor of 10.

	Cubes	Office	Trees	Bar
Computation time (sec)				
$Onyx^2$- Infinite Reality	**1.4**	**11.7**	**10.2**	**21.3**
O_2	13.0	93.9	103.5	303.2
Memory cost (MB)	13	52	131	170
Display time (ms)				
$Onyx^2$- Infinite Reality	4	100	340	1040
O_2	24	270	1360	2500
Texture size limit	256	256	256	256
Convolution links	2	16	13	20

Computation times are mainly proportional to the scene complexity, except between scenes *Office* and *Trees*, where the increase in complexity mainly concerns that of blockers involved in convolution links. Indeed, the complexity of blockers is only accounted for when sampling the blocker image in the convolution method, which is done at a negligible cost by the graphics hardware. On the other hand, the computation time for a complete radiosity solution mainly depends on the number of convolution links used for the scene. It also depends on on the maximum size allowed for the radiosity textures, as will be confirmed in Section 8.3.

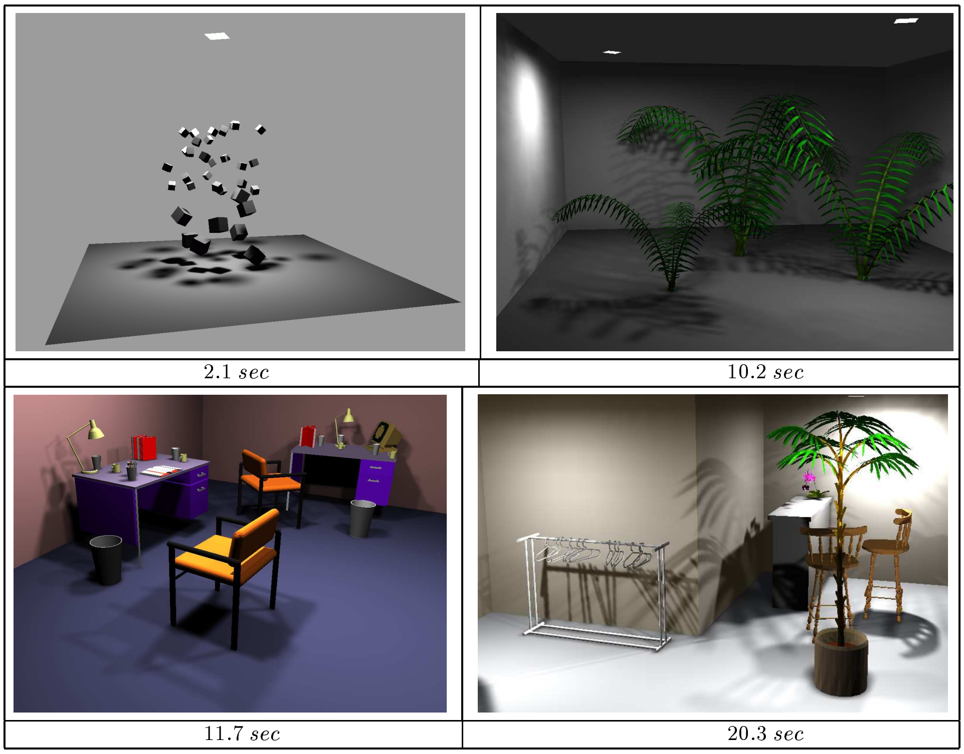
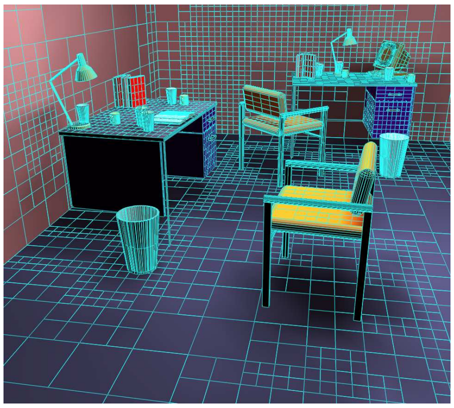


Fig. 13. *Images and computing time for the scenes presented above.*

Time for displaying the solution increases proportionally to the number of polygons for the input scene, and not to the number of convolution links. This is not surprising because the number of textured polygons is in general very small as compared to the non textured ones and their rendering cost is nearly the same on the two graphics workstations we used.

As predicted, approximations in the shadows can be seen as incorrect variation of the illumination in penumbra regions for objects having a large extent toward the source that produces the shadow, such as for the feet of the chairs in the image of the *Bar* scene of Figure 13.

(a) Hierarchical radiosity, 22 *sec*



(b) Mesh for image (a)

(c) Convolution radiosity, 17.5 *sec*

(d) Hierarchical radiosity 890 *sec*

Fig. 14. *Comparison of our algorithm and a standard hierarchical radiosity with clustering.* (a) *and* (b)*: result obtained by pure hierarchical radiosity in* 20 *seconds. The method did not have enough time to simulate the fine shadows (behind the lamp, the bin, on the desk) obtained in a shorter time by our method (displayed in* (c)*). In* (b) *we see that the mesh refinement was not pushed far enough to capture the shadows. To obtain comparable shadows, the hierarchical radiosity algorithm had to work for about* 890 *seconds. Artifacts in* (a) *like the missing illumination on the side of the front desk come from the approximate computation of form factors associated to links arriving on clusters.*

We also have observed that convolution links are mainly established during the first iteration because this corresponds to the propagation of direct illumination, which is the contribution of the illumination that contains the largest local variations.

8.2 Comparison with hierarchical radiosity and clustering

Figure 14 presents a comparison of our algorithm and a classical hierarchical radiosity method, obtained by directly transferring control to the standard refinement algorithm in our own refinement scheme. This example clearly demonstrates the argument given in the introduction of this article: hierarchical radiosity wastes a lot of time computing shadows using very fine mesh elements. If the required computation time is too small, such a method can not simulate fine shadows. Note that because it is difficult to compare the accuracy of shadows computed using textures and using piecewise uniform elements, the computation time in Figure 14d is not very significant. To obtain this image, we simply pushed the mesh refinement so as to obtain a resolution comparable to the resolution of the textures (256×256) in the shadow regions.

With respect to rendering times and memory costs, our algorithm is also much more interesting than classical hierarchical radiosity for equivalent accuracy: rendering a texture is much faster than rendering mesh elements for equivalent resolution, whatever the content of the texture. In our implementation, a mesh element requires 228 bytes of memory, which could roughly be compared to 1 byte per shadow mask pixel. This means that, at equivalent resolution, a texture is preferable to mesh elements as soon as the region occupied by the shadow is more than $\frac{1}{228}$ times the area of the polygon (This is even less if the mesh in constrained).

8.3 Influence of various refinement parameters

As we saw in Section 6, the algorithm is controlled by five parameters. We focus here on the two main parameters that are usually sufficient to produce simulations at various levels of speed and quality: the maximum error for convolution links E_{max}, and the maximum time t_{max} dedicated to their computation.

		Number of	texture size			Blocker subdivision			Number of
E_{max}	Time	convolution links	min	mean	max	min	mean	max	FFT
10.0	22.0s	10	256	460	512	1	3.7	27	57
8.20	36.0s	9	256	455	512	1	7.44	27	85
6.15	75.0s	8	256	416	512	1	49.3	167	411
4.04	91.0s	14	128	274	512	1	39.9	167	587
2.30	99.0s	14	64	201	256	8	84.5	183	1211

Fig. 15. *Evolution of the solution for various values of* E_{max}.

8.3.1 *Influence of* E_{max}. The table in Figure 15 shows the computation time and other interesting data associated to simulations in the *Trees* scene. The first comment on these results is that the computation times are statistically inversely proportional to E_{max}. Looking more closely at the statistics, it appears that the mean number of blockers used for convolution links (due to blocker refinement) and the total number of Fast Fourier Transforms follow the same behavior. At the same time, the mean number of pixels in shadow textures (*e.g* the square of the mean size for shadow textures) is proportional to E_{max}. For small values of E_{max} (*e.g* $E_{max} = 4.04$), some textures can not satisfy the accuracy at a high resolution and are therefore given up for smaller textures at lower hierarchical levels.

Looking at the corresponding images in Figure 16, the visual quality is increased as E_{max} is decreased until it reaches 6.15: shadow sharpness becomes more and more adequate, depending on relative positions of the tree leaves and the walls. In the last picture, however ($E_{max} = 4.04$), two visual artifacts appear: on the floor a discontinuity line separates the two palm tree shadows along a mesh boundary, and the complete shadow of a leaf has disappeared on the right wall. The discontinuity is due to the fact that different directions of projection have been chosen by the algorithm for the left side and right side textures, thus producing shadow maps of different sharpness along this line. Although such an artifact would normally disappear with further blocker refinement, it can also be eliminated by choosing the same direction of projection for all textures arriving on the same patch. The disappearing shadow is due to the fact that a convolution link can't satisfy the accuracy E_{max} in less time than t_{max} on the corresponding patch. The algorithm has therefore decided to use standard links, and in this case the chosen ε_{max} threshold was not low enough to push standard link refinement far enough to account for the shadow.

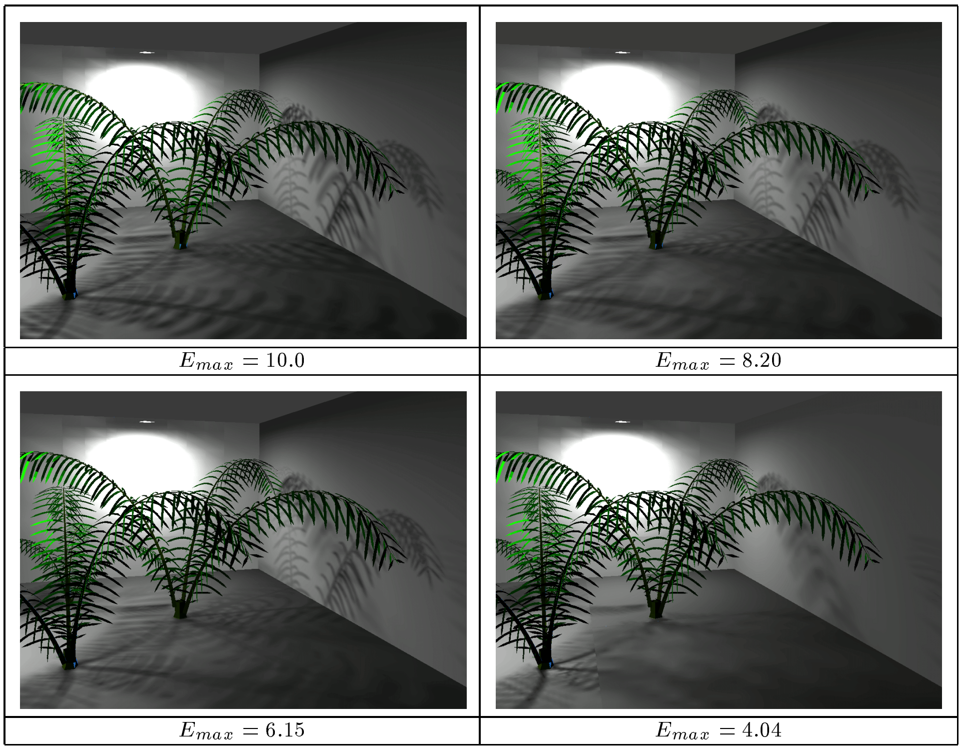


Fig. 16. *Images of the* Trees *scene for various values of the parameter controlling accuracy along convolution links.*

In conclusion, E_{max} is a relevant parameter for smoothly converting computation time into quality in the pictures, in the range where shadow textures are still applicable. To enlarge this range, we should either increase the time limit t_{max} for convolution links, or decrease the hierarchical radiosity error threshold ε_{max} to stimulate refinement in shadow areas not covered by convolution textures.

8.3.2 *Other parameters.* The error limit for energy-based refinement ε_{max} will not be studied deeply here but its impact on the solution is comparable to that in classical hierarchical radiosity algorithms with clustering: the smaller ε_{max}, the finer is the mesh of piecewise uniform radiosity, and the slower is the computation. Since ε_{max} is always used when a convolution link cannot be established, it should also be noted that it does not influence the computation time dedicated to the calculation of visibility textures: it is only considered after a convolution link has been given up.

8.4 Controling the quality of the simulation

Having two independent error threshold parameters E_{max} and ε_{max} for convolution and standard links is not very practical for rapidly choosing a compromise between fast, coarse solutions, and slow yet accurate simulations. Ideally, a user would prefer to have an interface with only one slider, that controls the overall accuracy of the simulation.

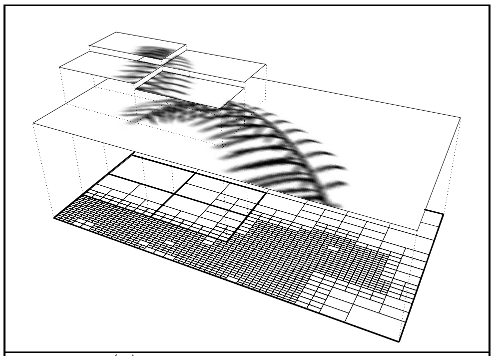
The problem with obtaining a single parameter for controling the error is that parameters E_{max} and ε_{max} do not have a common signification in terms of quality in the resulting images: one characterizes the accuracy in the shadows as seen by the human eye, while the other is a pure $L_\infty-$based convergence error of light propagation in the scene. Besides, visually accurate images can correspond to a large convergence error, and conversely fully converged solutions can contain visual artifacts. Therefore, when changing a convolution link into a set of standard links (or changing standard links into a convolution link) it is not possible to obtain a smooth variation of both the overall quality of the simulation and the computation speed. For instance, when breaking a convolution link into standard links, it is possible to adapt the resolution of the mesh so as to reach the accuracy in the texture using the estimation of the shadow sharpness described in Section 6.1, but the computation cost will explode for textures with hard shadows because they correspond to a sub-pixel mesh. The same problem occurs when two adjacent textures share an edge and when one of the corresponding convolution link is refined into standard links. To avoid discontinuity artifacts between the remaining texture and the mesh elements, the mesh must be refined along the common edge down to the accuracy of the texture.

The alternative to the problem of comparing both error thresholds is to suppress the time limit on blocker subdivision for convolution links. This is obtained by setting t_{max} to infinity. In that case, convolution links are never turned into standard links and solutions smoothly converge to nearly exact images, when decreasing ε_{max} and E_{max}, as far as the blockers can be subdivided. As the convolution method only computes an approximation of the shadows for blockers that can't be subdivided, our hierarchical radiosity algorithm then will not converge to an exact solution as far as it uses convolution links, and will invariably go back to standard links for very high precision. Anyway, the strength of our method is to produce visually accurate pictures even for large error threshold / small computation times.

8.5 Various examples

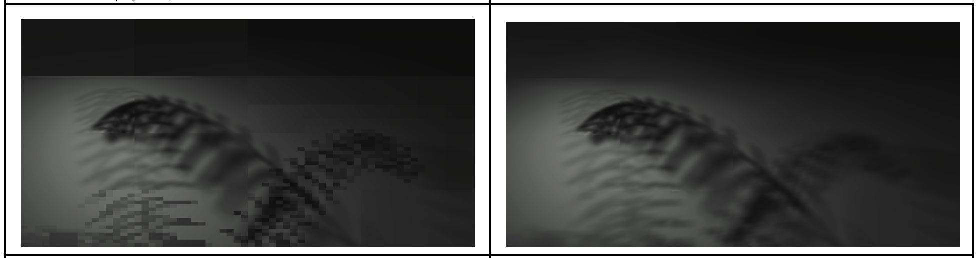
An example of smooth shading radiosity textures and piecewise uniform mesh is shown in Figure 17. Since the wall is lit by two different light sources, convolution links can produce partially overlapping radiosity textures. In this figure, the four small textures are produced by the same light source (say source #1) and the large one by the other source (source #2). At the same time, the patch has been refined in areas where light source #1 produces shadows that are not already handled by textures. The progressive mesh refinement in adjacent regions is due to a constraint in the quadtree, mainly used for anchoring.

Figure 18 shows an example where reflectance is also represented by textures. The computation and rendering is handled as discussed in Section 7.2. In this example, one can also see that a large proportion of fine details in the radiosity textures are not noticeable when rendered over reflectance textures, which suggests that reducing the radiosity texture



(a) Layers of mesh and textures

Fig. 17. *This figure illustrates the application of our smooth shading technique to a patch corresponding to the back wall of the* palm tree *scene as seen in Figure 13. The patch has been partially subdivided (See* (a)*) to account for one part of the shadow, while five radiosity textures handle the other parts at various hierarchical levels. In* (b) *the patch is rendered without smooth shading and we clearly see that shadows have been split between radiosity textures and piecewise uniform elements. In* (c) *the smooth shading operation (with anchoring) on the patch hierarchy eliminates all vertex discontinuities.*



(b) Not smooth-shaded result

(c) smooth-shaded result

resolution would be an interesting way of accelerating computation without decreasing the visual accuracy of the solution [Ferwerda et al. 1996; Myszkowski and Kunii 1994].

As previously explained, our refinement algorithm allows convolution links to arrive on complex clusters as well as single surfaces, producing a shadow on the cluster with a single shadow map. An example of such configuration is shown in Figure 19a: the plant pot receives a shadow cast through the chair by the light source in blue. The set of blue polygons on the tree trunk is also selected as potential occluders although they do not actually cast a shadow on the pot.

Figure 19b gives an example of the mesh produced by our refinement algorithm on the *Trees* scene (The trees have been removed from the image for clearer display). On this example we clearly see that the mesh actually serves the global energy balance while radiosity textures focus on shadow details at a much finer scale.

Figure 20 is a view of the stairs in the *Bar scene*. Although the shadow cast by the handrail has been formed without blocker subdivision, the shadow seems to be smoother on the bottom of the wall. This effect is due to the large angle between the direction of projection used to compute the blocker image and the normal of the wall itself, during the application of the convolution method. The projection of the texture is thus wider in the regions of the wall that lie at larger distance to the source.

9. CONCLUSIONS AND FUTURE WORK

We have presented a new hierarchical radiosity algorithm with clustering in which two different representations of radiosity contributions coexist: textures, computed by the convolution method and associated to *convolution links*, and standard mesh elements for which energy transfers are computed by classical hierarchical radiosity links. The new algorithm automatically chooses the kind of link for each energy transfers with respect to computation cost and visual quality for each configuration, as well as the refinement depth of the

(a) Scene with reflectance textures	(b) Uniform reflectance values

Fig. 18. *Example of a scene with reflectance textures* (a)*. In* (b)*, the reflectance textures have been replaced by a uniform reflectance value, which increases the visual definition in the radiosity textures.*

(a) Convolution link on cluster	(b) Mesh for scene *Trees*

Fig. 19. *(a) this image shows a convolution link arriving on the cluster containing the plant pot, as an example of establishment of a convolution link at a high level of the hierarchy. It can be seen that the selected blocker (the chair, in the pink cluster) casts a shadow on the pot. (b) example of a typical mesh obtained with a small energy error threshold. However, at the most refined locations, the resolution of the mesh is still four times larger than the equivalent resolution of the radiosity textures containing the shadows of the leaves, which illustrates the de-correlation of visibility and global illumination calculation in our algorithm.*

source, receiver or blockers, depending on the user-defined quality threshold for global energy balance and shadow accuracy. We have also presented the method to display the solution and various examples to illustrate this technique.

Our results on scenes of varying complexity show that the conjunction of a specific method for on-the-fly computation of soft shadow masks, with the traditional refinement of hierarchical radiosity, opens the way to computing accurate yet visually pleasant radiosity simulations within a range of very fast computation times. The robustness of the convolution method, indeed, allows to compute shadows in a very wide range of configurations.

Since we de-correlate visibility from the computation of the radiosity kernel, our algorithm does not necessarily converge toward the exact solution of the radiosity equation. However, it should be noted that this kind of error is hardly perceptible for the human eye and only concerns the gradient of the illumination in penumbra regions of the solution.

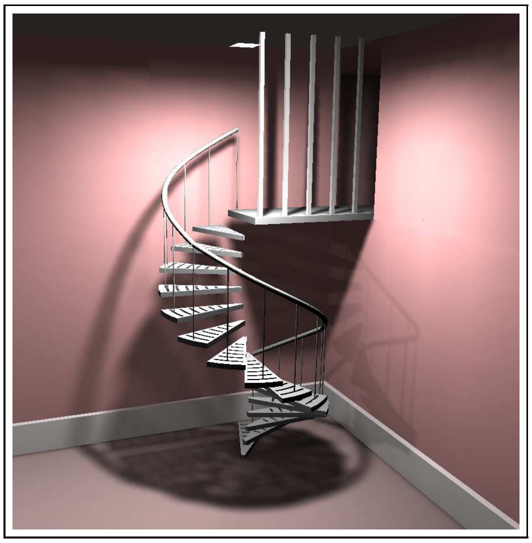


Fig. 20. *Image of the stair of the* Bar *scene.*

Our method has been presented as based on piecewise-uniform hierarchical radiosity. Nothing prevents our algorithm to be transposed in the context of higher order hierarchical radiosity. Such an implementation would simply increase the cost of the computation of form factors for standard links since more than one coefficient are needed for each energy transfer. This would consequently require changing the optimal balance of convolution and standard links.

To extend the discussion in Section 8.2 about memory optimization, texture compression schemes should be investigated. Future efforts should also be directed towards the creation of a more user friendly set of interface parameters to control the accuracy of the solution. Instead of separately fixing the accuracy of shadow masks and the equilibrium of light energy, it would be more practical to have a single error threshold influencing both computations. As discussed in Section 8, this may imply a gap in computation times when suddenly replacing a shadow mask by mesh elements unless its computation time becomes comparable to that of the mesh elements due to very high blocker refinement.

ACKNOWLEDGMENTS

This work was supported in part by the European Union under Esprit LTR project #24944 ARCADE "making radiosity usable". We also would like to thank the referees for their useful comments and suggestions.

REFERENCES

BASTOS, R., GOSLIN, M., AND BADLER, N. I. 1997. Efficient rendering of radiosity using texture and bicubic interpolation. In *1997 Symposium on Interactive 3D Graphics* (April 1997), pp. 71–74. ACM SIGGRAPH.

BAUM, D. R., RUSHMEIER, H. E., AND WINGET, J. M. 1989. Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors. In *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, Volume 23 (July 1989), pp. 325–334.

COHEN, M., CHEN, S. E., WALLACE, J. R., AND GREENBERG, D. P. 1988. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, Volume 22 (August 1988), pp. 75–84.

COHEN, M. F. AND WALLACE, J. R. 1993. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA.

DRETTAKIS, G. AND SILLION, F. 1996. Accurate Visibility and Meshing Calculations for Hierarchical Radiosity. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Render-*

ing) (New York, NY, 1996), pp. 269–278. Springer-Verlag/Wien.

DURAND, F., DRETTAKIS, G., AND PUECH, C. 1999. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics 18*, 2 (April), 128 – 170. See http://www-imagis.imag.fr/˜Fredo.Durand.

FERWERDA, J. A., PATTANAIK, S. N., SHIRLEY, P., AND GREENBERG, D. P. 1996. A Model of Visual Adaptation for Realistic Image Synthesis. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)* (1996), pp. 249–258.

GERSHBEIN, R., SCHRÖDER, P., AND HANRAHAN, P. 1994. Textures and Radiosity: Controlling Emission and Reflection with Texture Maps. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)* (1994), pp. 51–58.

GIBSON, S. AND HUBBOLD, R. J. 1997. Perceptually driven radiosity. *Computer Graphics Forum 16*, 2 (June), 119–128.

GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. 1984. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, Volume 18 (July 1984), pp. 212–222.

GRANIER, X. AND DRETTAKIS, G. 1999. Controlling memory consumption of hierarchical radiosity with clustering. In *Graphics Interface* (June 1999), pp. 58–65. available at http://www.dgp.toronto.edu/gi99/papers/146.

HAINES, E. AND WALLACE, J. 1991. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering* (1991), pp. 122–138.

HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, Volume 25 (July 1991), pp. 197–206.

HECKBERT, P. 1990. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, Volume 24 (August 1990), pp. 145–154.

HECKBERT, P. 1992. Discontinuity Meshing for Radiosity. In *Third Eurographics Workshop on Rendering* (Bristol, UK, May 1992), pp. 203–226.

HECKBERT, P. S. AND HERF, M. 1997. Simulating soft shadows with graphics hardware. Technical report (Jan.), CS Dept., Carnegie Mellon U. CMU-CS-97-104, http://www.cs.cmu.edu/ ph. See also the Sketch in Siggraph'96 visual proceedings.

KAJIYA, J. T. 1986. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, Volume 20 (August 1986), pp. 143–150.

KELLER, A. 1997. Instant radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, Volume 31 (1997), pp. 49–56.

LISCHINSKI, D., SMITS, B., AND GREENBERG, D. P. 1994. Bounds and error estimates for radiosity. In A. GLASSNER Ed., *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series (July 1994), pp. 67–74. ACM SIGGRAPH: ACM Press. ISBN 0-89791-667-0.

LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. 1992. Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications 12*, 6 (November), 25–39.

MARTIN, I., PUEYO, X., AND TOST, D. 1998. A two-pass hardware-based method for hierarchical radiosity. *Computer Graphics Journal (Proc. Eurographics '98) 17*, 3 (September), C159–C164.

MAX, N. L. 1991. Unified Sun and Sky Illumination for Shadows Under Trees. *CVGIP: Graphical Models and Image Processing 53*, 3 (May), 223–230.

MOLLER, T. 1996. Radiosity techniques for virtual reality - faster reconstruction and support for levels of detail. In N. M. THALMAN AND V. SKALA Eds., *WSCG 96 (Fourth International Conference in Central Europe on Computer Graphics and Visualization)*, Volume 1 (Plzen, Czech Republic, 1996), pp. 209–216. University of West Bohemia.

MYSZKOWSKI, K. AND KUNII, T. L. 1994. Texture Mapping as an Alternative for Meshing During Walkthrough Animation. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 375–388.

NEIDER, J., DAVIS, T., AND WOO, M. 1993. *OpenGL Programming Guide*. Addison-Wesley, Reading MA.

PRIKRYL, J. AND PURGATHOFER, W. 1998. *Eurographics 98 State of the Art Reports*, Chapter State of the Art in Perceptually-Driven Radiosity, pp. 89–104. Eurographics Association. ISSN 1017-4656.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In M. C. STONE Ed., *Computer Graphics (SIGGRAPH '87 Proceedings)*, Volume 21 (July 1987), pp. 283–291.

REICHERT, M. C. 1992. A Two-Pass Radiosity Method Driven by Lights and Viewer Position. Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, NY.

SCHRÖDER, P., GORTLER, S. J., COHEN, M. F., AND HANRAHAN, P. 1993. Wavelet Projections for Radiosity. In *Fourth Eurographics Workshop on Rendering* (Paris, France, June 1993), pp. 105–114.

SILLION, F. 1995. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. *IEEE Transactions on Visualization and Computer Graphics 1*, 3 (September), 240–254.

SILLION, F. AND DRETTAKIS, G. 1995. Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)* (1995), pp. 145–152.

SILLION, F. AND PUECH, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA.

SMITS, B., ARVO, J., AND GREENBERG, D. 1994. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)* (1994), pp. 435–442.

SOLER, C. 1998. Hierarchical Representations of Visibility for Error Control in Lighting Simulation. *Available in french with introdutions, conclusions, captions and important articulations translated in english, at* http://www-imagis.imag.fr/Public/Membres/Cyril.Soler/. Ph. D. thesis, Université Joseph Fourier, Grenoble I.

SOLER, C. AND SILLION, F. X. 1998a. Automatic calculation of soft shadow textures for fast, high quality radiosity. In G. DRETTAKIS AND N. MAX Eds., *Rendering Techniques'98 (Proceedings of Eurographics Rendering Workshop '98)* (New York, NY, 1998), pp. 199–210. Springer Wien.

SOLER, C. AND SILLION, F. X. 1998b. Fast calculation of soft shadow textures using convolution. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)* (1998), pp. 321–332.

WANGER, L. R., FERWERDA, J. A., AND GREENBERG, D. P. 1992. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications 12*, 3 (May), 44–58.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, Volume 12 (Aug. 1978), pp. 270–274.

ZATZ, H. R. 1993. Galerkin Radiosity: A Higher Order Solution Method for Global Illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)* (1993), pp. 213–220.