Automatic Calculation of Soft Shadow Textures for Fast, High Quality Radiosity

Cyril Soler and François X. Sillion
iMAGIS [1] – GRAVIR/IMAG

Abstract.
We propose a new method for greatly accelerating the computation of complex, detailed shadows in a radiosity solution. Radiosity is computed using a "standard" hierarchical radiosity algorithm with clustering, but the rapid illumination variations over some large regions receiving complex shadows are computed on the fly using an efficient convolution operation, and displayed as textures. This allows the representation of complex shadowed radiosity functions on a single large polygon. We address the main issues of efficiently and consistently integrating the soft shadow calculation in the hierarchical radiosity framework. These include the identification of the most appropriate mode of calculation for each particular configuration of energy exchange, the development of adequate refinement criteria for error-driven simulation, and appropriate data structures and algorithms for radiosity representation and display. We demonstrate the efficiency of the algorithm with examples involving complex scenes, and a comparison to a clustering algorithm.

1 Introduction

Despite the impressive progress in realistic rendering techniques over the last two decades, the production of very accurate images involving the simulation of light exchanges remains an expensive process, especially for complex scenes (more than a few thousands of objects).

Hierarchical radiosity techniques[4, 9, 7] were very successful in introducing the ability to trade accuracy for speed in a controlled manner. Unneeded form factor calculations can be avoided, and the solution to the global illumination problem can be estimated with the desired accuracy. In practice however, these techniques are mostly used to compute fast approximations of the solution. While possible in theory, the calculation of very high quality solutions using hierarchical radiosity is still expensive for complex scenes.

Of course this depends on the definition of "quality". For many applications, including lighting design, it may suffice to correctly represent the overall balance and flow of light. For applications in realistic rendering, however, shadows should be of very high quality[6]. This goal pushes hierarchical radiosity to its limits, because it requires a very fine subdivision of shadow variation areas. This is a direct consequence of the simple shape assumption of the radiosity function –constant or linear–across each mesh element.

In this paper, we introduce a new approach where complex shadowed radiosity functions can be represented across large elements using textures. Textures have previously

[1] iMAGIS is a joint research project of CNRS, INRIA, INPG and UJF within the GRAVIR/IMAG laboratory. Contact: csoler@imag.fr, http://W3imagis.imag.fr/Membres/Cyril.Soler/csoler.gb.html

been used for the representation of precomputed radiosity[6], or the storage of sampled illumination values[5, 1], but in our approach, textures are computed as a whole, and on the fly as an alternative method for simulating the energy transfer between two hierarchical elements. Basically, the textures are obtained by modulating the direct irradiance calculation by the "soft shadow map" that accounts for the portion of the source visible from any point on the receiver. This idea was used in[11, 8] but at the expense of computing time due to the large number of visibility tests necessary to obtain the shadow map.

We have recently introduced a new tool for the calculation of soft shadow maps, based on the convolution between images of a source and a blocker[10]. This tool has definite strengths: it intrinsically captures fine soft shadow details, integrates an error control tool and produces artifact-free soft shadow maps at interactive rates. It also has an associated cost, and should only be used in appropriate configurations. In a general radiosity simulation, we encounter a variety of situations, some of which may be suited for the use of this tool. In this paper we show

- that the convolution tool can dramatically accelerate the calculation of slightly approximate soft shadows across surfaces, consistently with hierarchical radiosity.
- how to select configurations where the convolution tool can/should be used
- how to choose appropriate convolution parameters automatically
- how to combine soft shadow textures with traditional hierarchical radiosity representations
- how to control the hierarchical simulation process, using and balancing the variety of possible refinement techniques

The resulting hierarchical radiosity system generates global illumination solutions, including very accurate soft shadows, in a matter of seconds for scenes containing several tens of thousands of polygons.

The paper is organized as follows: we begin with a rapid summary of the convolution technique for the generation of soft shadows (Section 2). Section 3 discusses the issues raised by the combination of hierarchical radiosity and the convolution method. In Section 4 we propose a number of solutions to these issues, in particular for the combination of different radiosity representations. Refinement criteria for the hierarchical simulation are discussed in Section 5. We then present some results (Section 6) and conclude.

2 Creating soft shadow textures using convolution

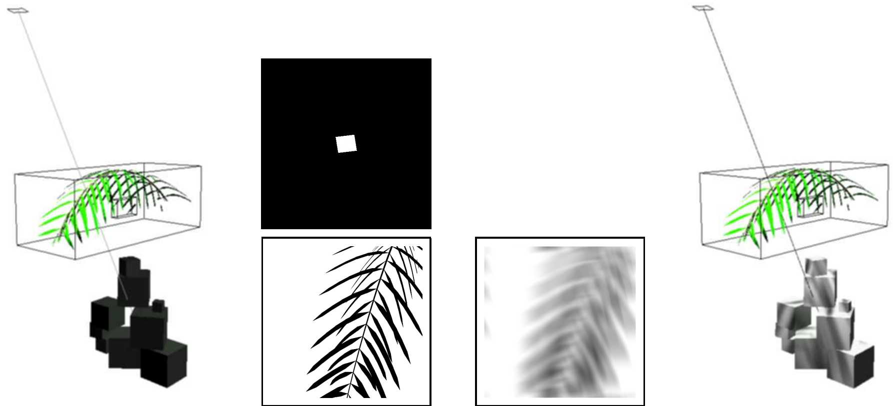
This section is intended to provide the reader with the essential principles for the computation of soft shadow maps using a convolution operation. A detailed presentation of this technique can be found in [10].

2.1 Basic principle

The *convolution method* provides a means to compute the illumination on a selected receiving object, subject to the light emitted by an extended light source, in the presence of a set of occluding objects. We refer to these occluders collectively as "the blocker" since it is considered as a cluster.

We first obtain an image of the source and the blocker by rendering the associated geometry in an offscreen buffer. For this projection, the source is viewed using an orthogonal transformation and the blocker is viewed using a perspective projection from the center of the source (See Fig.1)

Secondly, we compute the convolution of the two images by multiplication in the spectral domain, using *Fast Fourier Transforms*.



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

Fig. 1. Three basic steps of the convolution method: To compute the shadow cast by a 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 on the receiver (d)

The resulting image can be shown to be a valuable approximation of the portion of the source that is visible from any given point on the receiver. We call such an image a *soft shadow map*. We simulate the illumination on the receiver by using this image as a texture that we modulate by the reflectance of the receiver and the unoccluded form factor to the source.

During offscreen rendering of the source and blocker, both projection viewing frustum are adjusted so as to achieve a particular aspect ratio between the two images. This ratio accounts for the relative distance of the three components: the further the source, the smaller its size in the image, and the sharper the shadow.

This principle can be extended to non planar or non uniform sources, and more general receivers such as object clusters [10].

2.2 Hierarchical combination of soft shadow textures from different blockers

The above convolution method only provides a mathematically exact soft shadow map when the source, blocker and receiver are planar and parallel. In this particular case, using the convolution to compute the illumination on the receiver only amounts to neglecting the correlation between the radiosity kernel and the visibility terms.

To better understand the approximation involved when the blocker is non-planar, note that two blockers that look identical as viewed from the source when generating the blocker image will produce the same shadow on the receiver, if their relative distances from the source and the receiver are identical. Thus the actual position and size of the blocker between the source and receiver is not totally accounted for. The most noticeable consequence is that in a large blocker, all polygons produce shadows of equal sharpness, though their relative distance from the source and the receiver may vary.

In a general configuration, the thicker the blocker, the larger is the error. We give in [10] an estimator of this error, and a way of controlling it: to obtain a more accurate result, we separately compute soft shadow maps for N child clusters in the blocking cluster, using the convolution method with specific parameters, and combine them to produce a soft shadow map suitable for the whole blocker. The main cost of this operation is that of $2N$ additional FFT operations. This means that we are able to compute more accurate images at a larger computation cost.

2.3 Motivation for using the convolution method in a hierarchical radiosity system

The typical computation cost for a soft shadow texture using our method is less than 400*ms* for 256×256 textures on both SGI O^2 and *Onyx*2 computers, for blockers ranging from 100 to $40,000$ polygons. As a general rule, 256×256 textures are only necessary for very large surfaces such as walls. On most ordinary objects, a 64×64 texture is sufficient. Because the offscreen rendering of the blocking polygons uses graphics hardware, the cost of the convolution method essentially depends on the cost of the FFT, e.g. on texture size.

In a hierarchical radiosity system, energy transfers are usually computed between patches or clusters of any size. To achieve a given accuracy on a receiving polygon, complex visibility events may impose a fine subdivision to this polygon, and therefore the computation of many form factors including visibility tests.

The preceding arguments suggest that computing the energy transfers on sufficiently large receivers using the convolution method can be far cheaper than performing a classical subdivision, in the case where a large number of complex blocking objects participate to the shadow on the receiver.

We discuss in the next section the main avenues of incorporating the convolution method into a hierarchical radiosity system.

3 Principles for using soft shadow maps in a hierarchical radiosity algorithm

Considering the potential benefits of using the convolution method for the calculation of shadow maps, we would like to leverage the power of the technique in the context of hierarchical radiosity. This should allow a hybrid system to automatically select the convolution method whenever appropriate, while maintaining the benefits of radiosity (global energy balance, indirect illumination effects, ability to provide solutions with various cost/accuracy trade-offs, generation of view-independent solutions for diffuse scenes, etc.).

Therefore we wish to be able to compute the illumination on some potentially large receivers using the convolution method, during a hierarchical radiosity iteration. This can be accomplished by introducing two distinct types of *links* in a hierarchical radiosity system. In addition to the usual type, carrying a form factor estimate between two objects, *convolution links* are created for appropriate configurations of the source and receiver. The radiosity contribution of a convolution link on the receiver is represented by the association of a soft shadow texture and the kernel values used to modulate it (representing unoccluded irradiance estimates at selected points on the receiver).

Specific algorithms will be presented in the next section: In particular, care must be taken to derive a consistent algorithm for the representation and combination of energy contributions, coming either from “normal” radiosity calculations or from the convolution tool. Therefore, the gathering operation and the Push/Pull step must be modified to ensure consistency and efficiency.

Proper rendering of the computed solution also requires some care, as we must merge uniform or interpolated radiosity values with several textures modulated by arbitrary factors.

Since we are introducing an alternative method for the representation of light exchanges, the hierarchical refinement phase must be extended to properly incorporate this additional choice. We have designed an algorithm for link refinement that estimates the error associated with a given link and then chooses the best way to reduce this error among the different options for each kind of link. This algorithm will be presented in detail in Section 5

4 A complete hierarchical radiosity system with accurate shadows

We now describe a possible implementation of the above ideas, realizing a consistent combination of hierarchical radiosity with clustering and soft shadow maps. We explain how to store and gather energy across convolution links, how to perform a consistent push/pull using the information computed during the gathering operation, and how to display the resulting radiosity functions.

4.1 Gather

For standard links, gathering means multiplying a single radiosity value of the source element by the form factor associated to the link, which gives its irradiance contribution. As shown in the push/pull section, the source radiosity value is the sum of standard radiosity and average values of all textures assigned to that source.

For a convolution link, the source polygon is rendered with all its radiosity textures to form the source image, which accounts for the correlation between the source radiosity fonction and the visibility through the blocker. For that kind of link, a complete gathering operation would ideally consist in computing the soft shadow texture and modulate 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 at the leaves of the hierarchy only. One reason for this is that the illumination values strongly depends on the geometry of the leaf itself, rather than on the parent on which the convolution link is attached, which in certain cases may be a cluster.

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 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, whereas is can be achieved when rendering, using a two-pass shadowing technique.

4.2 Push/Pull

During push/pull, uniform irradiance values coming 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 textures stored during the gathering are collected on a *texture stack* and pushed down to the leaves of the hierarchy.

Once on a leaf, for each of the collected soft shadow textures, a proper set of modulation values is computed from reflectance and unoccluded form factor values. Practically, these values are computed at the vertices of a raw *display mesh*, which size depends on the geometry of the leaf. We also store of each of these points adequate texture coordinates in the plane of the soft shadow texture, that are used when rendering. We call the resulting object, combining the soft shadow map, the irradiance values and texture coordinates, a *complete* texture. Thus the Push/Pull operation essentially transforms the soft shadow maps into complete textures at the leaf nodes of the hierarchy.

Using the information computed at the same mesh nodes, we also compute an approximative 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.

The pseudo-code presented in Fig.2 shows the entire push/pull process.

4.3 Display

To render the computed solution, we need to display the total radiosity value of each leaf of the hierarchy. This means that we must add to the standard radiosity of each leaf, the list of soft shadow textures that can be stored on it.

We first draw the polygon with its uniform radiosity value and then the different radiosity textures using $OpenGL$ blending capabilities (with an add operation).

When rendering the texture modulated by the direct illumination values at each node of the texture mesh, $OpenGL$ clamps these color values to the range $[0,1]$ before using them to modulate the texture. This means that in a situation where the direct illumination exceed 1.0 but the modulated texture still lies into $[0,1]$, the complete texture won't be rendered correctly. To achieve the adequate texture modulation by an illumination map L whose maximum value M is larger than 1.0, we render the texture n times and modulate it by $\frac{1}{n}L$, where n is the smallest integer larger than M. This technique partially solves the problem for it also scales by a factor n the steps in the shadow texture (These steps come from the fact that textures are stored in 8 bits arrays). Except for very high illumination gradients due to extreme light values, this effect is not noticeable.

Note that because each texture is drawn on the same polygon that the one used to display the uniform radiosity values (i.e leaves of the hierarchy), setting the $z-$buffer comparison function to *less or equal* is sufficient to avoid $z-$buffer comparison artifacts in the resulting image.

We summarize in Fig.3 the different steps of rendering algorithm.

```
PushPull(node H, Spectrum IrradDown, Stack TextureStack, Spectrum RadUp)
    RadUp= 0
    TextureStack→Push(H.SoftShadowTexList())

    if(H.Is_A_Leaf())
        RadUp = (H.Irradiance() + IrradDown) * H.Reflectance()
        ForAll  T∈ TextureStack
            RadUp += ComputeTexAverage(H,T)
    else
        TextureStack→Push(H.CompleteTextures())
        Spectrum R
        ForAllChildren c of H
            PushPull(c,IrradDown+H.Irrad(),TextureStack,R)
            RadUp += R * c.Area()
        TextureStack→Pop(H.CompleteTextures())
        H.DeleteCompleteTextures()
        RadUp /= H.Area()

    H.SetRadiosity(RadUp)
    TextureStack→Pop(H.SoftShadowTexList())
    H.ResetSoftShadowTexList()
```

Fig. 2. Pseudo code for a Push/Pull with radiosity textures

5 Controlling refinement

We present in this section the refinement algorithm we use in our method.

5.1 Principle

The general purpose of a refiner is to decide if computing an energy transfer between a given source S and receiver R can be done while keeping the approximation errors under a user-defined bound ε. If so, it establishes a link, that contains the necessary information to compute the transfer (for instance form factors). If not, it assumes that a more precise result can be obtained when computing transfers between sub-regions of the source and/or the receiver, and subdivides the element that most reduces this error.

Since we deal with two kinds of links, the refiner must be able to decide where it can use a convolution link or not. This requires a criterion for the identification of source/receiver pairs for which using a convolution is possible and useful. We propose that the configuration must fulfill the following conditions:

1. The energy transfer between the source and the receiver must be “significant”.
2. The receiver must be large enough to justify the use of a texture with non-trivial size.
3. Blockers must be present between the source and receiver, and cast noticeable shadows: too smooth a shadow can be easily represented by smooth shading a small collection of large sub-areas of the receiver.

```
DrawPoly(Polygon P)
    SetColor(P.Radiosity() - P.Textures().MeanValue() )
    DrawSimplePoly(P)
    Enable(BLENDING)
    SetZBufferFunction(≤)
    ForAll T∈ P.Textures()
        n = SmallestIntegerLargerThan(T.MaximumLuminance)
        for i = 1to n
            DrawPolyWithTexture(P,T,n)
    Disable(BLENDING)
```

Fig. 3. Pseudo code for rendering a polygon with radiosity textures

4. The computation of the shadow map using convolution must reach a given accuracy at a reasonable cost.

We have seen in Section 2.2 that when computing a transfer using the convolution method, an alternative way of reducing the error is to subdivide the blocker. Therefore, in the general case, the refiner must answer the following questions:

- do I establish a link at this level, and what kind of link do I use ?
- if not, I choose between the three following refinement alternatives: source refinement, blocker refinement, or receiver refinement.

We have combined these requirements to form the refinement algorithm described in the next section.

5.2 Refinement algorithm

Our strategy is to use a convolution whenever possible, provided that the approximation error and the computation cost lie some user-defined intervals.

Thus, our refinement algorithm consists in successively checking for all the convolution site selection conditions in order of their computation cost. If using a convolution is not technically possible, or can't satisfy the refinement error condition, we transfer the refinement control to a standard refiner that tries to establish a standard link or split the source or receiver (See Fig.4 for a general overview of this). As a standard refinement algorithm, we use a method similar to the algorithm described in [2].

We first compute an upper bound B on the energy transfer from the source to the receiver, If B is smaller than a user-defined threshold, we do not refine the current link (or we create a standard link, if no link is already present).

The next test concerns the size of the receiver: complex scenes are usually filled with fine polygonal models, and we want to avoid computing too large a number of convolutions for very small polygons. For more generality, we want this test to apply to clusters as well as surfaces. Thus, we compare the dimensions of the object bounding box to a fixed value.

At that point, a little more work must be done to examine the blockers that occlude visibility between the source and receiver. The blocking clusters are first selected using an algorithm described below. If no blockers are found, we do not apply a convolution.

Otherwise, we compute the number of sub-blockers to use in the convolution method necessary to be able to compute an illumination map whose error (Given by the error criterion from [10]) is below an error threshold E_{max}. We set the variable E_{rr} to the best error estimation reached. In both cases N is the number of sub-blockers to use.

If $E_{rr} > E_{max}$, no convolution can be used. We transfer control to the standard refiner. Otherwise, the required precision can be satisfied using N blockers, i.e $2+2N$ FFT transformation. We have experimentally determined a constant C for which the computation time for a FFT on a $n \times n$ image is approximatively

$$t(n) = C\,n^2 \log(n)^2$$

Thus, the computation time for the illumination map can be estimated as

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

If t is larger than a given computation time threshold, we give up the convolution. In the case where t is small enough, we create a convolution link between S and R, using N sub-blockers.

When one of the convolution sites identification criteria fails, some information can nevertheless be extracted from the various computations: The blocker selection process helps classifying the visibility configuration (full or partial visibility). While computing E_{rr} and N, a bound α_{min} on the sharpness of the penumbra regions on the receiver can be computed. We use this bound for determining a dimension δ under which it is not interesting to split the receiver.

In summary, we make use of four user defined thresholds to control refinement:

a_{min} : minimum receiver area for a convolution link.
E_{max} : maximum blocker error tolerance for a convolution link.
t_{max} : maximum computation time allowed for a single convolution.
ε : minimum energy transfer value for a link.

Blocker selection. The selection of blocking clusters between the source and the receiver is done using a shaft culling method [3]. We first build the shaft between the source and the receiver. Then, a recursive procedure selects the largest clusters that intersect the shaft without containing the source or the receiver. Generally, a single blocker is selected. When two or more clusters are selected, we pack them in a single virtual blocker.

The non-containing test ensures that neither the source nor the receiver would participate to the resulting umbra, as a part of the blocker hierarchy. A more careful examination of the problem leads to turning this test into a non-overlapping test, to also discarding possible blocking objects that hide behind the receiver with respect to the source or hide behind the source with respect to the receiver.

As a consequence, this test must be sufficiently numerical tolerant to allow a cluster lying on a polygon (Typically an object lying on the floor) to remain selected as a occluder/receiver configuration.

6 Results

We show on the left hand side of Figure 6 (See also color plates) an image of a classical interior scene computed with 16 convolution links (from each source to each of the

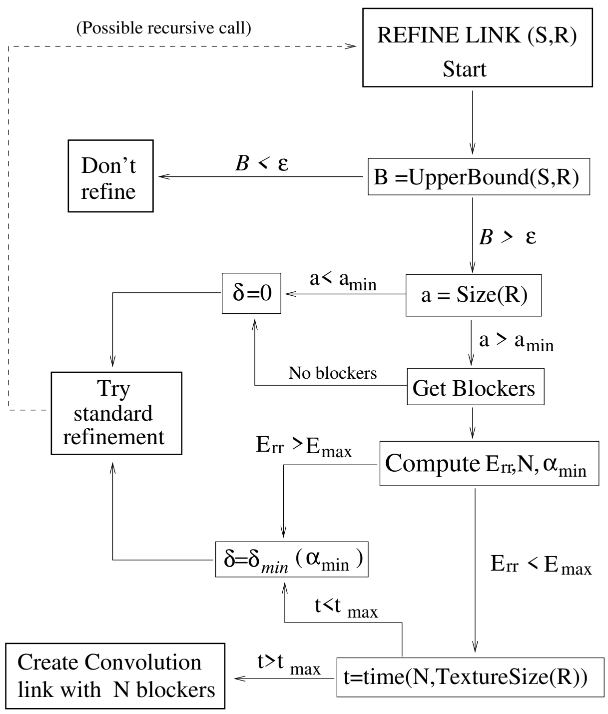
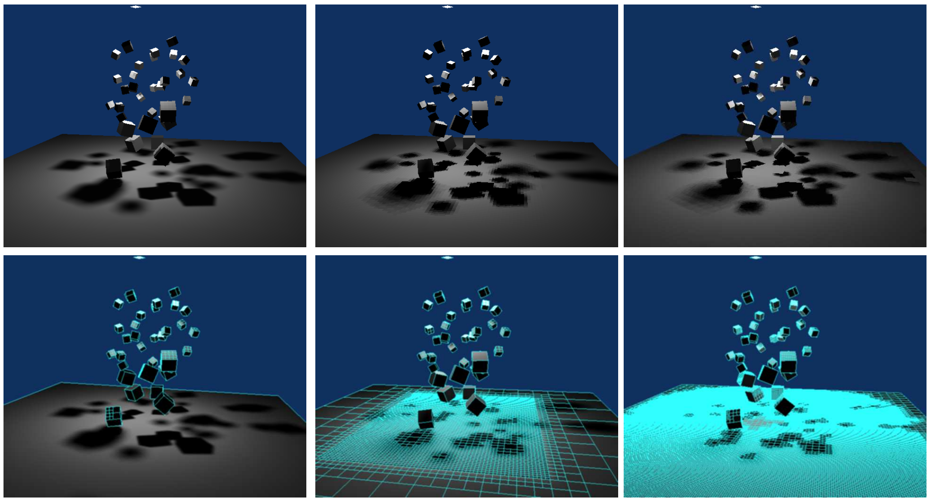


Fig. 4. Overview of the refinement algorithm for a link between a source *S* and receiver *R*

two walls and the floor, and from some of the sources to clusters containing the desks). Three iterations were performed for a total computation time of 20.2 seconds on an *Onyx2 workstation - Reality engine*. On the right hand side is a solution obtained with only classical hierarchical radiosity with clustering in the same computation time. We show the HR mesh and the smooth shaded version of the result as a comparison to the previous image. Very fine shadows have been captured by our method with only a few convolution links in a short time, resulting in a high quality radiosity solution.

Figure 5 (See color plates) shows the effect of changing the refinement parameters on the proportion of convolution links (e.g textures) used by the refiner. The blocker time parameter t_{max} is fixed at $1sec$. In (a) the use of convolution links is favored by a large blocker error E_{max}. The floor receives a convolution link (512×512 texture, 6 sub-blockers) at the highest level. In (b) we see that reducing the blocker error tolerance forces the refiner to switch to standard links in the center of the floor, where textures can no longer satisfy the error criteria E_{max} in less time than t_{max}. The progressive meshing around this region is a constraint in the quadtree. Convolution links arrive at the highest possible level in this region. Finally, in (c) too small a blocker error threshold totally prevents convolution links. The minimum receiving area for standard links has been set so as to obtain the same resolution with standard refinement as the texture in (a).

Note that some significant part of the computing time is used to compute transfers between the cubes or clusters that contain them. Some of the cubes also receive textures from convolution links from the source to the cluster containing them.



(a) 12.6 *sec*
$E_{max} = 4.04$, $a_{min} = 0.802$

(b) 29 *sec*
$E_{max} = 2.02$, $a_{min} = 0.108$

(c) 158 *sec*
$E_{max} = 0.18$, $a_{min} = 0.108$

Fig. 5. The effects of changing refinement parameters on the kind of link used and the computation time

7 Conclusions and future work

We have described a new radiosity algorithm based on the use of a convolution technique to compute shadow maps where complex blockers may cause a classical hierarchical radiosity algorithm to get lost in mesh subdivision. The algorithm basically merges two radiosity storage methods: spectrum and textures. It could be used with another texture computation method than the convolution with hardly any modification provided that the replacing algorithm gives a bound on the error incurred and a way to decrease it.

Storing radiosity values as textures is not compatible with *already-textured* environments, unless we manually compute the pixel-by-pixel product of radiosity textures and scene textures, which is a costly operation. Nevertheless, we can reasonably expect that this capability may be provided by future graphics hardware.

The intra-cluster visibility problem is not specifically treated by our method but it can be addressed by refining the receiver for textured links arriving at a cluster. This causes some parts of the cluster to become treated as occluders, thus producing the desired internal shadows.

Finally, a single user defined error threshold for controlling the refinement would be appreciated. The difficulty is here to find comparable error estimators for both kinds of links. We think that this gap can be filled using perceptually based error criteria.

References

1. James R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, August 1986.

Fig. 6. Comparison between our method (*left*) and a standard HR algorithm (*right*) for equivalent computation times (20 *sec*, 5380 input polygons).

2. S. Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, December 1996.
3. Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.
4. Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
5. Paul Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, August 1990.
6. Karol Myszkowski and Tosiyasu L. Kunii. Texture mapping as an alternative for meshing during walkthrough animation. In *Fifth Eurographics Workshop on Rendering*, pages 375–388, Darmstadt, Germany, June 1994.
7. François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995. (a preliminary version appeared in the fifth Eurographics workshop on rendering, Darmstadt, Germany, June 1994).
8. Philipp Slusallek, Michael Schroder, Marc Stamminger, and Hans-Peter Seidel. Smart Links and Efficient Reconstruction for Wavelet Radiosity. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 240–251, New York, NY, 1995. Springer-Verlag.
9. Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
10. Cyril Soler and François Sillion. Fast calculation of soft shadows textures using convolution, 1998. to appear in SIGGRAPH 98 conference proceedings.
11. Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series:* SIGGRAPH '93 (Anaheim, CA, USA), pages 213–220. ACM SIGGRAPH, New York, August 1993.