Textures and Radiosity:

Controlling Emission and Reflection with Texture Maps

Reid Gershbein Peter Schröder Pat Hanrahan

Research Report CS-TR-449-94
March 1993

Abstract

In this paper we discuss the efficient and accurate incorporation of texture maps into a hierarchical Galerkin radiosity algorithm. This extension of the standard algorithm allows the use of textures to describe complex reflectance and emittance patterns over surfaces, increasing the realism and complexity of radiosity images. Previous approaches to the inclusion of textures have either averaged the texture to yield a single color for the radiosity computations, or exhaustively generated detail elements—possibly as many as one per texture pixel. The former does not capture important lighting effects due to textures, while the latter is too expensive computationally to be practical.

To handle texture maps requires a detailed analysis of the underlying operator equation. In particular we decompose the radiosity equation into two steps: (i) the computation of irradiance on a surface from the radiosities on other surfaces, and (ii) the application of the reflectance operator ρ to compute radiosities from irradiances. We then describe an algorithm that maintains hierarchical representations of both radiosities and textures. The numerical error involved in using these approximations is quantifiable and a time/error tradeoff is possible. The resulting algorithm allows texture maps to be used in radiosity computations with very little overhead.

1 Introduction

Radiosity methods compute the illumination, both direct and indirect, from environments consisting of perfectly diffuse (Lambertian) reflecting and emitting surfaces [9, 14, 6, 5]. The resulting pictures contain subtle but important lighting effects, such as soft shadows and color bleeding, that enhance their realism. However, radiosity algorithms are still complex and far from general. For example, most computer graphics radiosity algorithms are limited to environments consisting of polygonal elements whose radiosities and reflectances are constant. Naturally such assumptions limit both the realism of the resulting pictures, and the utilization of the radiosity method in many applications. This immediately leads to the question of what additional rendering and modeling techniques can be combined with the radiosity method to further enhance the realism of the final imagery.

One of the oldest rendering techniques is the use of texture maps to modulate surface color [3]. In the case of radiosity, texture maps may be used to spatially modulate the emittance and reflectance of the surface. The advantage of texture maps is that the apparent surface complexity increases without the increase in geometric complexity that arises if the surface were to be subdivided into many small surface polygons. This last distinction is particularly important in the case of radiosity, since the computational cost of a radiosity simulation may grow quadratically with the number of primitives in the scene. Consequently, it is desirable to be able to include texture maps in radiosity computations, and preferably to do so at a modest increase in cost.

These difficulties notwithstanding earlier researchers have incorporated emissive as well as reflective textures into radiosity sytems to increase the realism of the generated images. Cohen *et al.* [5] incorporated textures as a post process. During the radiosity computations a texture mapped polygon would have a constant reflectance equal to the average texture map reflectance. For final renderings the textures were incorporated at the resolution of the original texture. This was done by dividing the computed radiosities by the average reflectance and then multiplying them by the true reflectance. This yields visually complex images, but does not accurately account for lighting effects because the spatially varying emission and reflection across a surface is not considered during the radiosity calculation. Dorsey *et al.* [7] also incorporated texture maps. They were motivated by the need to simulate lighting effects for opera lighting design. Consequently the simple average/post-process technique was not applicable. Instead they created many small polygons to get a reasonable approximation of the original texture. As mentioned above this quickly leads to excessive computation times, but does lead to extraordinary pictures.

To summarize, the averaging technique does not compute the correct illumination effects, and the exhaustive approach is expensive in both time and storage. An approach that captures the indirect illumination effects due to texture maps to within some accuracy while having low computational cost is clearly desirable, and is the motivation behind this paper.

Two developments in recent radiosity research are relevant for our present discussion, hierarchical radiosity, and higher order Galerkin methods. Hierarchical approaches, first proposed by Hanrahan *et al.* [11], introduce hierarchical subdivision and interaction schemes into radiosity computations. Objects are allowed to exchange energy at different levels of detail. This is based on the observation that interactions between relatively far away primitives can be approximated with a coarse subdivision, while only close interactions require a fine subdivision. The level of detail at which two objects (or parts of objects) interact is determined by their ability to capture an interaction correctly within a specified error criterion. An asymptotically faster radiosity algorithm results.

Galerkin approaches, first introduced by Heckbert [12] and consequently elaborated by Zatz [20], and Troutman and Max [18], use classical finite element techniques to solve the underlying radios-

ity integral equation. The main goal is to compute smoother answers than classical radiosity. An attendant benefit arises from faster convergence, further increasing the speed of radiosity algorithms. In any Galerkin approach all functions (emittance, reflectance, and the geometric kernel) are written out with respect to some set of basis functions. It has been pointed out by Zatz that this trivially allows for emittances which vary in some complex way over a surface so long as this variance can be described with the chosen set of basis functions.

The benefits of hierarchical radiosity and Galerkin methods have recently been unified under the framework of *wavelets* by Gortler *et al.* [10] and Schröder *et al.* [17].

Starting with a wavelet radiosity system we extend it to handle both emission and reflection textures. The theory and the resulting algorithm is the subject of this paper. We first carefully examine the structure of the underlying operator equation. In particular we show that the usual computation of radiosity as the product of reflectance and irradiance (and possibly active emission) is complicated by the hierarchical framework, which aims to preserve the computational advantage afforded by multi level descriptions. In fact, we formally and algorithmically separate the computation of irradiance from that of radiosity. This allows us to exploit economies in each of the computational steps which would be hard to take advantage of in the usual combined framework. In the resulting algorithm, the cost of performing a radiosity computation with texture maps is very similar to the cost without texturing.

2 Theory

2.1 The radiosity equation

The radiosity equation can be written as a simplification of the rendering equation [13] by assuming that all participating surfaces are perfectly diffuse (Lambertian), giving rise to the classical radiosity integral equation

$$\begin{aligned} B(x) &= B^e(x) + \rho(x) \int dx' G(x,x') B(x') \\ &= B^e(x) + \rho(x) E(x) \end{aligned} \tag{1}$$

where $B(x) = B^e(x) + B^r(x)$ is the radiosity at point x, consisting of the emitted radiosity $B^e(x)$ and the reflected radiosity $B^r(x)$. $\rho(x)$ is the reflectance, and $G(x,x')$ characterizes the radiant coupling between points x and x'. The integral is taken over the hemisphere about x and represents the amount of energy per unit area received from other surfaces. This is the irradiance $E(x)$. Once this irradiance is reflected (multiplication with $\rho(x)$) it contributes to the radiosity of point x.

2.2 The radiosity operator

The radiosity equation may be decomposed into two steps as shown in Figure 1:

1. *Compute irradiance from scene radiosities.*

   Define the operator $\mathcal{G}$ for a general function f

   $$\mathcal{G}(f)(x) = \int dx' G(x,x') f(x')$$

   Then this first step is

   $$E(x) = \mathcal{G}(B)(x)$$

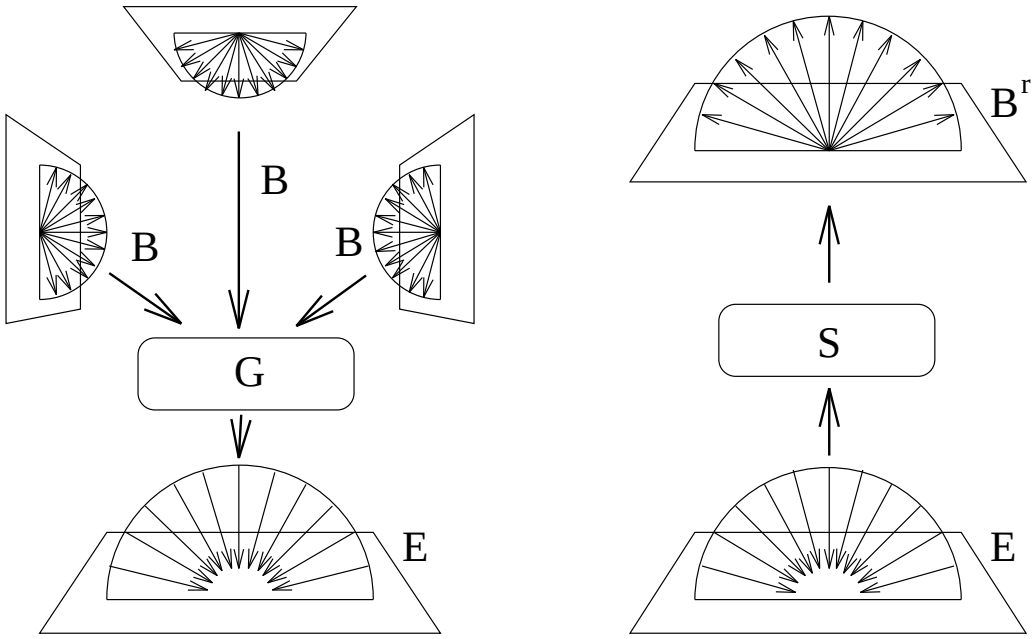

Figure 1: The two step computation of reflected radiosity, B^r. First the computation of irradiance, denoted by G, second the computation of the scattered irradiance, denoted by S.

Note that $\mathcal{G}$ is a geometric operator that takes into account the radiant exchange between a receiver surface and all other source surfaces in the scene. It contains the visibility relationships, the relative surface orientations, as well as the inverse square distance falloff.

$\mathcal{G}$ is a dense non-local operator, but it is smooth (in the absence of shadows). It is dense in that $\mathcal{G}$ may potentially couple any part of the environment to any other part of the environment. It is smooth in that $\mathcal{G}$ varies slowly as a function of the positions of the source and the receiver. In the parlance of integral equations we say that $\mathcal{G}$ has a very smooth kernel [2].

2. *Compute radiosities from irradiances and reflectivities.*

   Define the operator $\mathcal{S}$

   $$\mathcal{S}(f)(x) = \rho(x)f(x)$$

   Then this second step is

   $$B^r(x) = \mathcal{S}(E)(x)$$

   Note that $\mathcal{S}$ is completely determined by surface properties and as such does not depend on the environment. In contrast to $\mathcal{G}$, $\mathcal{S}$ is a local operator, but is, in general, not smooth. This is because the texture map representing the reflectance may contain high frequencies.

Using this operator notation we can rewrite Equation 1 as

$$\begin{aligned} B(x) &= B^e(x) + \mathcal{K}(B)(x) \\ &= B^e(x) + \mathcal{S} \circ \mathcal{G}(B)(x) \end{aligned} \tag{2}$$

The separation into $\mathcal{G}$ and $\mathcal{S}$ allows us to pursue different strategies to efficiently represent each operator individually. Sparse operators such as $\mathcal{S}$ are easy to represent efficiently. In contrast, dense operators such as $\mathcal{G}$ are normally difficult to represent efficiently. However, methods exist to efficiently represent dense, smooth integral operators [2]. In particular, hierarchical or wavelet

radiosity algorithms provide an efficient method for representing $\mathcal{G}$ [11, 10, 17]. They work because the smoothness of $\mathcal{G}$ allows it to be approximated efficiently using a hierarchy of levels of detail. Unfortunately, combining $\mathcal{S}$ with $\mathcal{G}$ creates an operator which is dense, but no longer smooth, causing difficulties for the hierarchical methods.

2.3 Projection methods

In order to make a radiosity system tractable it must be approximated by using a finite dimensional function space. This is done by projecting the radiosity, emittance, and reflectance functions onto a finite set of basis functions $\{N_i\}_{i=1,\ldots,n}$. For example, these basis functions might be piecewise constant, as in classical radiosity, they might be higher order polynomials as proposed by Zatz [20] for Galerkin radiosity, or wavelets [10]. In all of those cases the result is

$$\begin{aligned} B(x) \approx \hat{B}(x) &= \sum_i B_i N_i(x) \\ \rho(x) \approx \hat{\rho}(x) &= \sum_l \rho_l N_l(x) \end{aligned}$$

Limiting everything to these basis functions the solution to the original radiosity integral equation can be approximated by solving the following linear system

$$B_i = B_i^e + \sum_j K_{ij} B_j$$

where[1]

$$K_{ij} = \int dx\, \hat{\rho}(x) \int dx'\, G(x,x') N_j(x') N_i(x)$$

Suppose now we have computed irradiance with respect to our chosen basis. The next step is to multiply with $\rho(x)$ to compute the reflected radiosity. $\rho(x)$ too is given in some basis (typically the same as that used for E). Writing this out we get

$$B^r(x) = \hat{\rho}(x) E(x) = \left(\sum_l \rho_l N_l(x) \right) \left(\sum_i E_i N_i(x) \right) \tag{3}$$

If our chosen basis is a piecewise constant nodal basis, as is typically used in classical radiosity, this multiplication becomes very easy. For such a basis we have $\mathrm{supp} N_i \cap \mathrm{supp} N_j = \emptyset$ if $i \neq j$, i.e. the support of two basis functions with differnt index does not overlap. Therefore the application of the operator reduces to a simple multiplication of the E_i's with the ρ_i's. Consequently in this basis $\mathcal{S}$ is a diagonal operator. Suppose instead that our basis was a piecewise polynomial basis of some higher order, but still with local support. In this case the matrix corresponding to $\mathcal{S}$ would be block diagonal. Each block couples the basis functions with overlapping support. The number of overlaps is a constant subset of the set of all functions, and thus this block diagonal operator is also sparse.

When multiplying basis functions a small difficulty arises. To see this consider the use of a piecewise linear basis set and the case depicted in Figure 2. A linearly varying irradiance is multiplied with a linearly varying reflectance resulting in a quadratically varying radiosity. The latter cannot be represented in our chosen basis. The radiosity is of higher polynomial order

[1] We will ignore all scaling constants (e.g. for classical radiosity $\rho_i F_{ij} = \frac{K_{ij}}{A_i}$) to not burden the following derivations but rather to expose the fundamental connections between basis functions. Furthermore we will not deal with the questions of primal and dual bases but assume an orthogonal basis. Again, making the distinction would not engender any new insights but clutter our notation considerably.

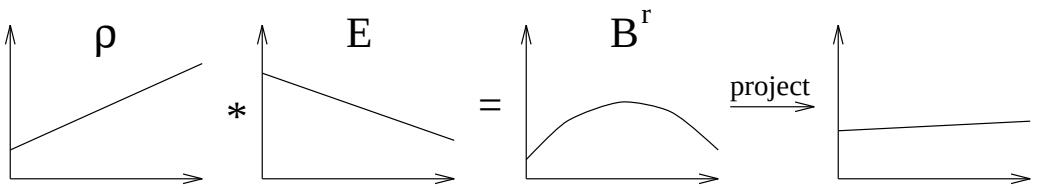

Figure 2: Multiplying an irradiance and reflectance which are representable by a given basis (e.g. piecewise linear) does not necessarily yield a reflected radiosity which can be represented in the same basis. Therefore, a finite-element algorithm may require a reprojection step.

than the individual component functions. This difficulty can be addressed by following up the multiplication with a reprojection step (see the rightmost step in Figure 2).

The above observations can be turned into a very straightforward algorithm. Whichever basis is used, we proceed in three steps

1. transform irradiance from its preferred basis to a nodal basis
2. apply (block-) diagonal $\mathcal{S}$
3. reproject onto the nodal basis
4. transform back to preferred basis for radiosity

This algorithm has the advantage that $\mathcal{S}$ is represented efficiently. However, it has the disadvantage that it requires an intermediate transformation between the preferred basis and the nodal basis. In general, this transformation may require n^2 operations. Fortunately, if the preferred basis is the wavelet basis, the transformation to and from the nodal basis requires only a linear number of operations.

An obvious improvement would be to express $\mathcal{S}$ directly in the wavelet basis. Unfortunately, the multiplication of the representations of ρ and E is more expensive. The support of many of the functions may overlap, resulting in many non-zero terms. Hence, the representation of $\mathcal{S}$ is no longer sparse. Fortunately, if we are only computing to within some chosen accuracy it may turn out that many of the products in the above expression make a negligable contribution to the total. It would be desirable to *only* compute the components that matter.

Two observations suggest methods for producing such an algorithm:

1. $\mathcal{S}$ may be smooth and if represented in the wavelet basis it may have a sparse representation. If, for example, the texture has regions with fairly little high frequency detail we will be able to adequately represent the texture in those regions with few coefficients. Wavelets afford us one way to optimally take advantage of smoothness if present.
2. The operator $\mathcal{S}$ will be followed by the operator $\mathcal{G}$ in the next iteration. Hierarchical radiosity systems [11, 10] use different levels of detail to compute the irradiance from the radiosity. In other words, the decision as to how far to subdivide the surfaces is based on the smoothness of $\mathcal{G}$, and as a result there is no need to compute the radiosity at a finer level of detail than it is needed in the next iteration.

These observations can be formulated into two variants of an algorithm which works directly with the given basis

1. exact

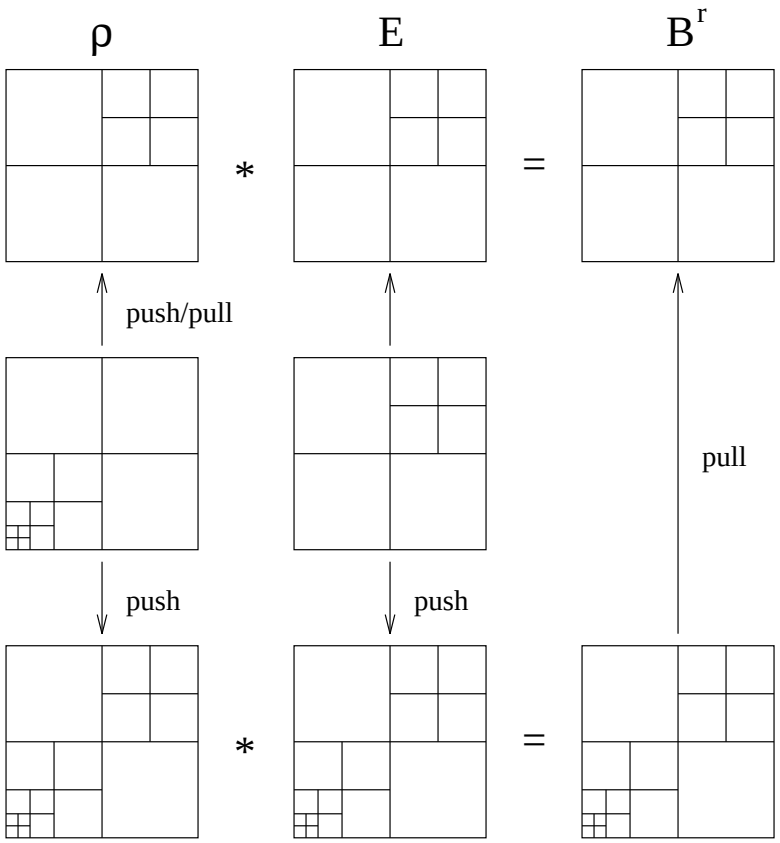

Figure 3: Two possible algorithms for applying $\mathcal{S}$ (ρ) to the irradiance E. The center row shows an example of a subdivision hierarchy as it might result when considering ρ and E separately. The bottom depicts an approach in which both representations are pushed to the finest detail before multiplication occurs. The result is then pulled up to the level of the reflected radiosity for the next iterations. Alternatively one can use the level of the radiosity for the multiplication with ρ without requiring any push/pull operations.

   (a) push the irradiance to the resolution level of ρ

   (b) apply $\mathcal{S}$

   (c) reproject

   (d) pull to level of outgoing radiosity

2. approximate

   (a) push and/or pull the irradiance to the resolution level of the radiosity[2]

   (b) apply $\mathcal{S}$

   (c) reproject

[2]We say push "and/or" pull since in some regions of the reflectance texture one may need to pull while in other regions one may need to push. For an example, see the lower left quadrant and upper right quadrant of the ρ texture in Figure 3.

These two algorithms are illustrated in Figure 3. In the center row we see graphical depictions of irradiance computed at some level of detail and the reflectance which is adequately represented at some other level of detail. We could push both irradiance and reflectance to a common representation, but not the finest level, and apply $\mathcal{S}$ (bottom row), pulling the result back to the level at which it is needed for the next iteration. This is contrasted with an approximate scheme (shown at the top) which computes $\mathcal{S}$ at the level of the radiosity. In this approximate scheme, less pushing or pulling is required, but a certain error will result.

In order to facilitate the following error analysis we define all the terms in the matrix realizations of the operators $\mathcal{G}$ and $\mathcal{S}$. Replacing $\mathcal{K} = \mathcal{S} \circ \mathcal{G}$ in the traditional form of the linear system we have

$$B_i = B_i^e + \sum_j K_{ij} B_j = B_i^e + \sum_k S_{ik} \sum_j G_{kj} B_j$$

Using $\langle f, g \rangle$ to denote the inner product, we can write

$$\begin{aligned} K_{ij} &= \langle \mathcal{S} \circ \mathcal{G}(N_j), N_i \rangle \\ G_{kj} &= \langle \mathcal{G}(N_j), N_k \rangle \\ S_{ik} &= \langle \mathcal{S}(N_k), N_i \rangle \\ &= \langle \sum_l \rho_l N_l N_k, N_i \rangle \\ &= \sum_l \rho_l \langle N_l N_k, N_i \rangle \end{aligned} \tag{4}$$

We now analyze the error introduced by the approximate algorithm. In what follows let $\mathcal{P}$ denote the projection operator onto the level of the outgoing radiosity, and let $\rho_d(x) = (\rho - \mathcal{P}(\rho))(x)$ denote the difference between the ρ and its projection at a particular level of detail. This corresponds to the finer levels of detail below the given level. We may then write

$$\mathcal{P} \circ \mathcal{S}(E) = \mathcal{P}(\mathcal{P}(\rho)E) + \mathcal{P}(\rho_d E)$$

Examining the right hand side we see two terms. The first, $\mathcal{P}(\mathcal{P}(\rho)E)$, arises from multiplying the approximation of ρ at the chosen subdivision level with the irradiance computed at that level, followed by a reprojection. The second term, $\mathcal{P}(\rho_d E)$, describes the effect of multiplying the finer detail with the irradiance and again reprojecting. Both are potential sources of error. If we only compute $\mathcal{S}$ at the level of the outgoing radiosity we are completely dropping the second term, $\mathcal{P}(\rho_d E)$. Using the fact that the magnitude of a linearly transformed vector is bounded above by the product of the magnitudes of the vector and the linear operator, $\|Ax\| \leq \|A\| \, \|x\|$, we may write

$$\|\mathcal{P}(\rho_d E)\| \leq \|\mathcal{P}\| \, \|\rho_d\| \, \|E\|$$

For orthogonal projections $\mathcal{P}$ we have $\|\mathcal{P}\| = 1$ and consequently the error in this term can be bounded by the product of the irradiance magnitude and the magnitude of the ignored texture detail. In the standard wavelet radiosity algorithm an *oracle* function is used to decide whether a given interaction needs subdivision to meet the error criterion [10]. Similarly, an oracle to decide at what level of detail $\mathcal{S}$ needs to be applied is needed. It can be based on a test whether $\|\rho_d\| \, \|E\|$ is less than some error threshold. This is very similar to the BF refinement [11] in ordinary hierarchical radiosity. In BF refinement an interaction between two elements ($\mathcal{G}$) is refined if the product of the magnitudes of radiosity and the form factor error estimate is above the error threshold.

The other source of error arises from the fact that some information is lost in the reprojection step after $\mathcal{P}(\rho)$ and E have been multiplied. This error however, is already implicitly taken into account by the oracle which governs the expansion of $\mathcal{G}$. This oracle decided that the radiosity at the receiving element is only needed to the level of detail at which we computed E.

```
Gather( QuadNode p )
  p.E = 0
  ForAllElements( q, p.Interaction ){
    p.E += Interaction[i].K * q.B
  }
  Gather( p.sw )
  Gather( p.se )
  Gather( p.nw )
  Gather( p.ne )

PushPull( QuadNode p )
  if( !p.Leaf() ){
    p.children.E += WaveletTransformDown( p.E )
    PushPull( p.sw )
    PushPull( p.se )
    PushPull( p.nw )
    PushPull( p.ne )
    p.B = WaveletTransformUp( p.children.B )
  }else{
    p.B = p.Be + ApplyOpS( p.rho, p.E )
  }
```

Figure 4: Pseudo code for the `Gather` and `PushPull`. Note that `B`, `E`, `rho`, and `Be` are two dimensional arrays of coefficients and the operations on them are matrix operations. The function `ApplyOpS` is given by Equation 5.

3 Implementation

We have added reflectance and emittance textures to a wavelet radiosity system as described in [10, 17]. Pseudo code for the two functions which changed from previous hierarchical radiosity systems, `Gather` and `PushPull`is given in Figure 4.

In the paragraphs below we give some more details regarding the following three parts of the overall system

1. functions to build wavelet representations of the texture (image) input files
2. extending the push/pull algorithm to include non-constant emissions and a new function `ApplyOpS`
3. implementation of renderman shaders for final rendering output

3.1 Wavelet encoding of textures

For the encoding of texture maps we used the same basis functions as were used in the rest of the system. These are the multi-wavelets introduced by Alpert [1], which are based on Legendre polynomials up to some order (we used order 0 through 3). The mathematics do not require us to use the same wavelet basis for the textures as for the other parts of the system, but it is a natural choice with the least modifications to the system overall.

After reading in a texture, which is currently limited to have a power-of-2 u and v extent, we pyramid transform it using the low pass filter sequence associated with the multi-wavelets [17]. This results in averaged representations of the original texture from the finest (input) level all the way to the overall average.

To start the transform we treat each pixel value as the coefficient of a constant basis function whose domain is the pixel's area. These constant functions are then projected, i.e. taking the inner product with the multiwavelets, onto the finest level of the hierarchy. This was chosen for simplicity, but other choices are possible. For example, one might use a cubic spline interpolant of the texture values at the finest level and evaluate the integrals of this function with the multi-wavelet bases at the finest level, before beginning the pyramid transform. The rest of the hierarchy is created by running the `PyramidUp` algorithm described in [10, 17].

At this point we have a full quadtree representation of the texture. For space reasons it may be desirable to prune subtrees in regions which do not have enough spatial detail to warrant finer level descriptions. This is akin to a lossy compression scheme applied to the original texture. We have not yet explored this possibility, but it would be useful for a production system, where one would like to allow large numbers of textures simultaneously.

3.2 Extending push/pull

The modification to the push/pull algorithm only occurs when it reaches the leaves of the hierarchy. After gathering irradiance, which is the $\mathcal{G}$ step, we need to push the irradiance to the leaves of the subdivision as induced by the $\mathcal{G}$ oracle. Ordinarily we would now multiply the coefficients of the basis functions with the scalar ρ before executing the pull to get ready for the next iteration. Instead we apply $\mathcal{S}$ as given by Equation 4.

So far we have used a single index on all coefficients to indicate which basis function they belong to. In so doing we have abstracted the fact that all functions are defined with respect to surfaces, i.e. with 2 parameters. We will be explicit about the 2 parameter nature of our basis functions for a moment to give the exact expression for the application of $\mathcal{S}$. Let $\{N_i(u)\}_{i=1,\ldots,M}$ be the multi-wavelets of order M. Surface basis functions are given by $\{N_i(u)N_j(v)\}_{i,j=1,\ldots,M}$ and consequently we use double indices on B, E, and ρ. For example, if we use cubic basis functions, $M = 4$, and all coefficients can be thought of as 4×4 matrices. We still need the reprojection coefficients $R_{ilk} = \int du\ N_l(u)N_k(u)N_i(u)$. With this notation we have

$$B^r_{ij} = \sum_{lm} \sum_{kn} R_{ilk} R_{jmn} \rho_{kl} E_{mn} \tag{5}$$

Note that R_{ilk} can be precomputed once for a given choice of basis. Taking the example of $M = 4$ again we see that the 4×4 coefficients B^r_{ij} are a sum over all combinations of the 4×4 irradiance coefficients with the 4×4 reflectance coefficients. The weights in this sum are given by the $4 \times 4 \times 4$ reprojection coefficients. Only emittance is left to be added in before executing the pull and starting the next iteration

$$B_{ij} = B^e_{ij} + B^r_{ij}$$

3.3 Final output

As the renderer for final output we use RenderMan [19]. This requires the creation of a RenderMan Interface Bytestream (rib) file containing descriptions of the geometry of each leaf element generated during subdivision, as well as the computed coefficients for the basis functions associated with the element. We have written RenderMan shaders which accept these coefficients as parameters and evaluate the radiosity function at a given point (u, v) in parameter space of the surface element.

In the case of multi-wavelets this is achieved by implementing the Legendre basis functions in the shading language. Note that we do no post processing, such as, for example, vertex averaging, as is done in many traditional radiosity systems. The rendered images show the functions exactly as computed by the radiosity system, because the color of each sample is interpolated using the appropriate basis function during the rendering process.

For surfaces which do not have texture maps associated with them, we write out the coefficients of the *radiosity*. For surfaces with associated texture maps (emittance and/or reflectance) we write out the coefficients of the *irradiance* together with texture coordinates and references to the original texture files. The shaders take this description and for a given (u, v) evaluate the irradiance, multiply with the reflectance texture value, and add the emittance texture value.

Although our final renderings were done with RenderMan, the texture mapping and basis function interpolation scheme used could be easily added to any rendering system.

4 Results

Figure 5 shows the difference between using averaged textures and true textures during the radiosity computation:

- On the top we demonstrate the case of a reflectance texture mapped onto the floor. The texture map has equal amounts of red, green and blue, and these were chosen so the average texture value is gray and colorless. The left side shows the result of using the average texture color and adding the texture as a post process; the right shows the result achieved using the true texture map. Below each of the two pictures is a blowup of the region just above the floor to show the difference between the two computations more clearly. Note the strong color bleeding near the floor. These colors are absent if the average value is used.

- On the bottom we see the results of a similar experiment for emittances. In this case there are 4 ceiling lights with the same texture map, this time used for emittance instead of reflectance. Once again the left side shows the solution when using only the average color, and the right when using the true colors during the simulation itself. In both cases we also have a blowup of the region just below the ceiling lights where the differences are most pronounced. Again, with averaging no colored light is cast onto the ceiling and walls near the lights.

For these examples we also measured the execution time difference between the two approaches, excluding the initial build of the tree representation of the textures. For the emittance textures there was essentially no difference in runtimes between the use of averages and the proper texture. This is mostly due to the fact that the inclusion of emittance textures does not change the amount of computation necessary to account for the emittance. Whether constant emittance or textured emittance is used, in either case we only need to add the coffecients to the total. This is in contrast to the reflectance textures which do require a considerable number of floating point operations in the application of $\mathcal{S}$ (see Equation 5). We used cubic basis functions and a very small error threshold. The initial iteration is dominated by the computation of $\mathcal{G}$ which is identical for the averaged texture and the full texture. After that the cost of each full Jacobi sweep (complete gather) goes up by about 10%.

We argued earlier that it is in fact very easy to control the error of the application of the operator $\mathcal{S}$ and that we generally do not need to go lower than the leaf level of the element subdivision. To test the error introduced by using this technique we computed $\mathcal{S}$ in two different ways as depicted in Figure 6. On the left is an image of the irradiance computed for the back wall due to a ceiling light source. The reflectance texture for the back wall is shown underneath. The

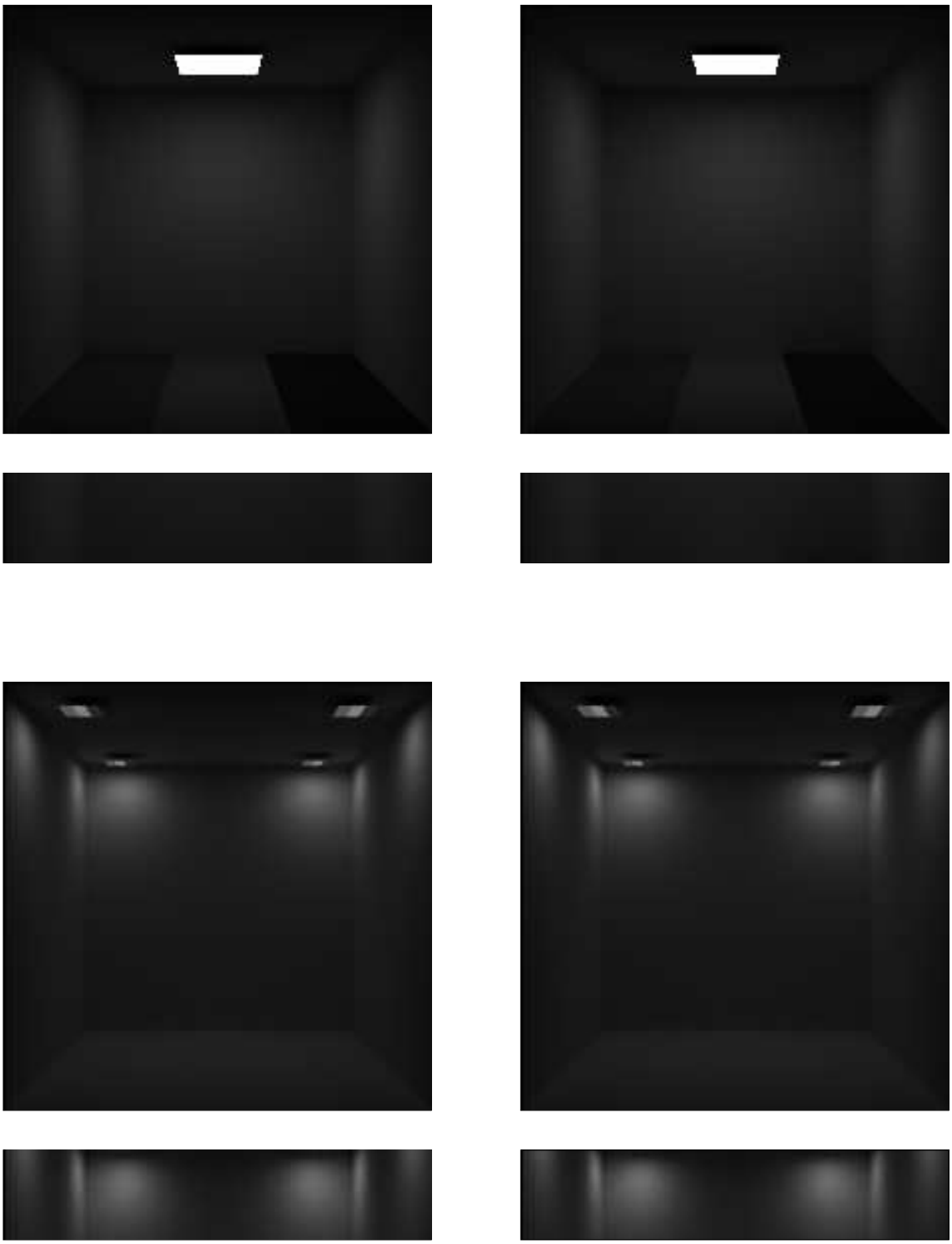

Figure 5: These pictures illustrate the difference in radiosity solutions produced by using true texture maps and approximating them with average values. The top row shows the effect of a reflectance texture with averaging (left) and without averaging (right). Similarly the bottom row shows this difference for emittances. These images were computed with cubic basis functions (M_4 multi-wavelets).

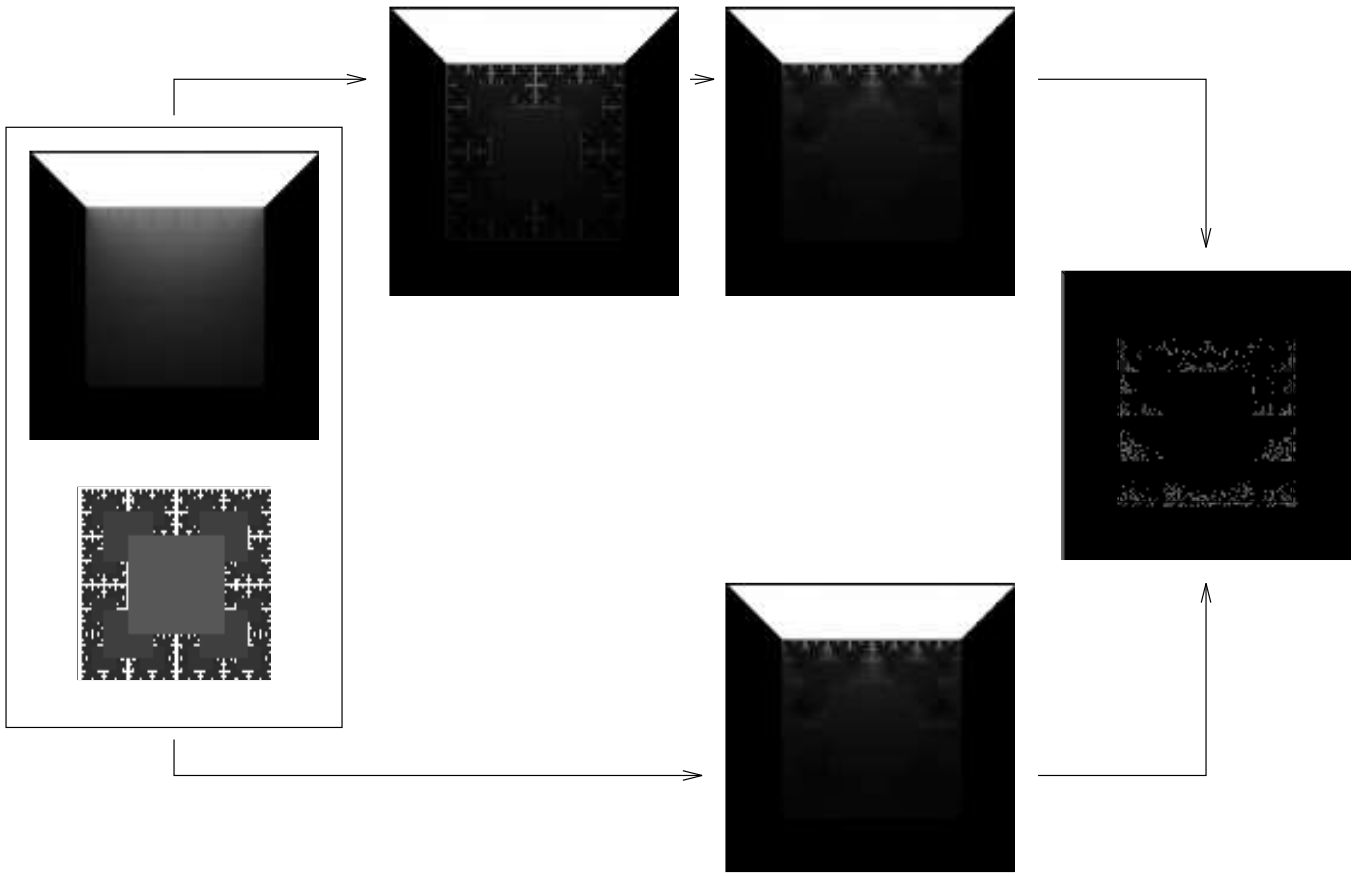

Figure 6: Errors in the application of $\mathcal{S}$. On the left is the irradiance created on the back wall by the light source at the top. The wall has a reflectance texture shown on the bottom left. The top path shows the sequence of steps when pushing irradiance to the bottom level and applying $\mathcal{S}$ at that level. The bottom path shows the application of $\mathcal{S}$ at the same level at which irradiance was computed. The final image shows the difference between the two paths. The differences are no worse than 1 part in 256 and are scaled up to be visible.

figure shows the two methods for computing the radiosity assuming the radiosity will be used to transfer energy to the floor and ceiling. The outgoing radiosity distributions are shown using both algorithms. The top path in the figure shows the exact calculation: the radiosity is computed at the finest level of the texture by interpolating the irradiance to finest level in the texture hierarchy, multiplying the irradiance by the textured reflectance to yield the radiosity, and then pulling the radiosity up to the level required for subsequent application of the irradiance operator in the next iteration. The bottom path computes an approximate $\mathcal{S}$, using information about the resolution at which radiosity will be used in the next iteration. This method is faster because the reflectance is applied at the highest possible level in the hierarchy. Note that the differences between the two radiosities computed using the different approaches are visually indistinguishable. In fact the difference between the two radiosities is never greater than 1 part in 256. The difference image is shown on the far right (the magnitude of the difference is scaled so that the patterning is visible).

The texture was designed to contain a repeating self-similar pattern with many different levels of detail. The outgoing radiosity must be represented at great detail near the edges of the wall because of their interaction with the floor and ceiling. This is because the size of the elements must be small if the distance between them is small. As a result the figure shows the high frequencies of

Figure 7: This scene illustrates the visual complexity added to a radiosity simulation by the inclusion of reflectance textures (11) and emission textures (1). Total simulation time 23.5 minutes.

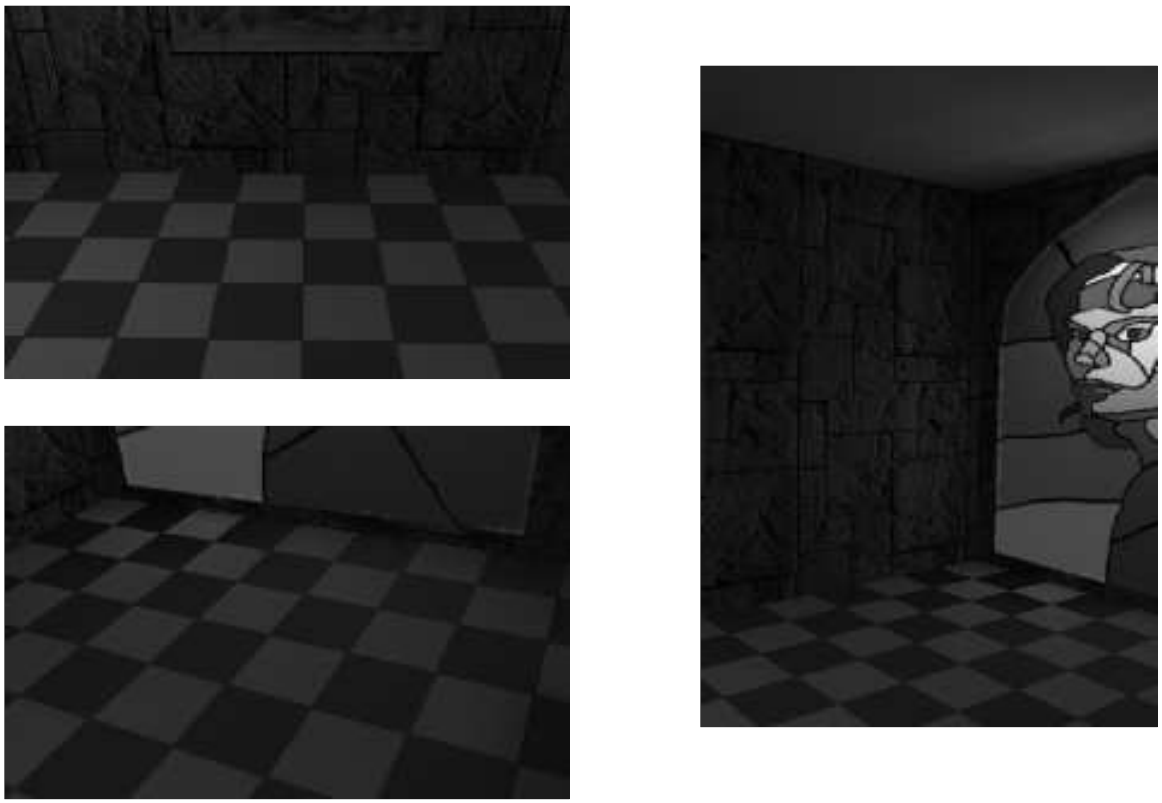

Figure 8: Three detail pictures of the room scene, showing the effects due to inclusion of textures during the simulation. On the upper left note the color bleeding of the floor texture onto the wall. The lower left shows the colored illumination on the floor due to the stained glass texture. The right picture shows another example of colored illumination on the left wall due to the emissive texture.

the texture. However as we move towards the center of the wall, bigger and bigger elements may be used because they are farther from the other surfaces. This implies that only low frequency variations in radiosity matter.

Figure 9: The stained glass window was created with the emissive texture on the left and the reflective texture (matte) on the right.

Finally, to test the robustness of the system, and to measure the impact of texturing on the running time of the radiosity algorithm, we computed the more complicated scene shown in Figure 7. It was modeled with 28 quadrilaterals and 12 texture maps. The stained-glass window is a single quadrilateral with the emittance and reflectance maps shown in Figure 9. We solved the system using linear multi-wavelets. A total of 29 seconds was spent on hierarchy creation for all texture maps (each with a resolution of 128^2 pixels). 6 complete gather sweeps were performed over 4024 elements with 46308 interactions. The size of the system reached 17.37Mb and ran in 23 minutes on an SGI Indy (R4000SC 100MHZ) computer. Figure 8 shows close-up views of some of the areas that were most influenced by the texture maps. The color changes around the stained-glass window and the reflection off the floor tiles onto the wall are effects that would have been difficult to model without the use of texture maps. The texture maps also allowed the system to cluster many floor tiles or window panes together when determining interactions by using the standard hierarchical error criteria [11, 10]. The fact that the achieved the high visual complexity with only a few quadrilaterals attests to the power of the texture mapping paradigm in the radiosity context.

5 Summary and conclusion

We have presented a technique which extends hierarchical radiosity algorithms to include reflectance and emittance textures during the solution process. This extension was facilitated by a careful examination of the radiosity integral equation. We separate the computation of irradiance, a global process, from the computation of reflected radiosity, a local process. In so doing we can take advantage of efficiencies in each step individually which cannot be readily taken advantage of if the two steps were merged, as is normally done. The new method has been implemented in a wavelet radiosity system. The resulting images demonstrate the desired lighting effects with little penalty in performance. As a result increasing the visual complexity of radiosity computations with texture maps is now practical.

Although the computational cost of using texture maps is now quite small, the storage and memory costs still may be quite large. An interesting research question is how to manage the texture data. Since only limited resolutions are ever needed, and since many radiosity runs may be performed using the same input textures, it is reasonable to precompute and store the wavelet representation of the texture. The radiosity program then need only read in the resolution sets that are required. This also suggests the intriguing possibility of compressing the texture maps using wavelets, both to minimize the cost of reading the textures, and to reduce the amount of disk space needed to store them.

The use of texture mapping techniques in computer graphics has a long tradition and many complex effects can be achieved. One such example has already been demonstrated in Figure 7, where two texture maps, one for emittance and another for reflectance are combined with a matte. As a result the odd shaped stained glass window is created from texture maps and a single quadrilateral. Other uses for texture maps include environment mapping and bump mapping [4]. The algorithm described in this paper may be easily modified to use environment maps; an interesting application of this would be to the modeling of skylighting, which is currently very expensive. A clever algorithm for approximating bump mapping in the context of radiosity is described in [4]; similar approximations could likely be used in a hierarchical radiosity algorithm, although this would require further research.

Many texture map applications do not use image files as a basis for texturing, but rather procedural textures [16, 15]. These could also be used in the context of wavelet radiosity. Two possibilities arise: procedural textures could be defined directly in the wavelet domain, similar to noise texture directly defined in the Fourier domain. Alternatively, one could take an arbitrary procedural texture and compute its wavelet expansion.

Some of these techniques may also be useful for the clustering problem. Imagine a surface described by a displacement map over a regular grid. Now we have actual geometric complexity, yet a multi resolution description still appears promising and could be incorporated in a straightforward manner into the texture mapping system. In fact one might imagine replacing entire complex ensembles of geometry with a "picture" during parts of the radiosity solution process. Another use for this system is in the area of inverse problems. Consider taking a photograph of a diffuse surface, i.e. capturing its radiosity, and using this as a texture in a radiosity simulation. This could be used to match real scenes with computed scenes [8]; and more generally might be useful for problem arising in computer vision, such as the computation of the reflectance map.

The technique we applied in our analysis, separating out the global $\mathcal{G}$ part of the computation from the surface local $\mathcal{S}$ part, may also be a fruitful analysis technique for other rendering problems. For example, in volume rendering it may yield new insights to separate the representation of the attenuation operator from the representation of the scattering operator. Another example is the computation of radiance, where one might gain new efficiencies from separating the incident radiance operator from the scattering operator.

Acknowledgements

The authors would like to thank James Shaw for permission to use his design for the stained glass window; Seth Teller for major contributions to the radiosity system; Don Mitchell and Michael Cohen for many useful discussions. The research reported here was partially supported by Apple, Silicon Graphics Computer Systems, and the National Science Foundation (CCR 9207966).

References

[1] Alpert, B., Beylkin, G., Coifman, R., and Rokhlin, V. Wavelet-like Bases for the Fast Solution of Second-kind Integral Equations. *SIAM Journal on Scientific Computing 14*, 1 (Jan 1993).

[2] Beylkin, G., Coifman, R., and Rokhlin, V. Fast Wavelet Transforms and Numerical Algorithms I. *Communications on Pure and Applied Mathematics 44* (1991), 141–183.

[3] Blinn, J. F., and Newell, M. E. Texture and Reflection in Computer Generated Images. *Communications of the ACM 19*, 10 (October 1976), 542–547.

[4] Chen, H., and Wu, E.-H. An Efficient Radiosity Solution for Bump Texture Generation. *Computer Graphics 24*, 4 (August 1990), 125–134.

[5] Cohen, M., Chen, S. E., Wallace, J. R., and Greenberg, D. P. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics 22*, 4 (August 1988), 75–84.

[6] Cohen, M. F., and Greenberg, D. P. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics 19*, 3 (July 1985), 31–40.

[7] Dorsey, J. O., Sillion, F. X., and Greenberg, D. P. Design and Simulation of Opera Lighting and Projection Effects. *Computer Graphics 25*, 4 (July 1991), 41–50.

[8] Fournier, A., Gunawan, A. S., and Romanzin, C. Common Illumination between Real and Computer Generated Scenes. In *Proceedings of Graphics Interface 93* (1993), pp. 254–261.

[9] Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. Modelling the Interaction of Light between Diffuse Surfaces. *Computer Graphics 18*, 3 (July 1984), 212–222.

[10] Gortler, S., Schröder, P., Cohen, M., and Hanrahan, P. Wavelet Radiosity. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 221–230.

[11] Hanrahan, P., Salzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics 25*, 4 (July 1991), 197–206.

[12] Heckbert, P. S. Radiosity in Flatland. *Computer Graphics Forum 2*, 3 (1992), 181–192.

[13] Kajiya, J. T. The Rendering Equation. *Computer Graphics 20*, 4 (1986), 143–150.

[14] Nishita, T., and Nakamae, E. Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection. *Computer Graphics 19*, 3 (July 1985), 23–30.

[15] Peachey, D. R. Solid Texturing of Complex Surfaces. *Computer Graphics 19*, 3 (July 1985), 279–286.

[16] Perlin, K. An Image Synthesizer. *Computer Graphics 19*, 3 (July 1985), 287–296.

[17] Schröder, P., Gortler, S. J., Cohen, M. F., and Hanrahan, P. Wavelet Projections For Radiosity. In *Fourth Eurographics Workshop on Rendering* (June 1993), Eurographics, pp. 105–114.

[18] Troutman, R., and Max, N. Radiosity Algorithms Using Higher-order Finite Elements. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 209–212.

[19] Upstill, S. *The RenderMan Companion.* Addison Wesley, 1992.

[20] Zatz, H. R. Galerkin Radiosity: A Higher-order Solution Method for Global Illumination. In *Computer Graphics Annual Conference Series 1993* (August 1993), Siggraph, pp. 213–220.