Linear Radiosity with Error Estimation

Sumanta N. Pattanaik and Kadi Bouatouch
IRISA, Campus Universitaire de Beaulieu, 35042 RENNES CEDEX, France

February 28, 1996

Abstract. We present a simple and inexpensive method for computing the estimates of error in a hierarchical linear radiosity method. Similar to the approach used in [1] for constant radiosity method, we compute lower and upper linear bounds of the actual radiosity function over the surface elements. We carry out this by computing linear upper and lower bounds of the kernel of the radiosity equation. Also we compute these bounds in a form which makes trivial the effort of projecting the integral equation involving such kernels. We provide the hierarchical algorithm for computing the radiosity bounds. We derive the expression for computing error-estimates from these bounds. Finally we propose a refinement indicator for carrying out the link refinement.

1 Introduction

Radiosity in an environment is governed by an integral equation of the following form [2]:

$$B^{(i)}(s,t) \quad = \quad E^{(i)}(s,t) + \rho^{(i)} \sum_{j=1}^{n} \int \int K^{(i \leftarrow j)}(s,t,u,v) B^{(j)}(u,v) du dv \quad (1)$$

where i and j are the indices of the surface elements, n is the total number of surface elements in the environment, (u,v) and (s,t) are respectively the *parametric* coordinates of a point on i and a point on j, and $K^{(i \leftarrow j)}(s,t,u,v)$, the kernel of the integral equation, gives the radiosity contribution of the differential area around (u,v) on element j towards the differential area around (s,t) on element i, and has an expression as given below.

$$K^{(i \leftarrow j)}(s,t,u,v) = \frac{\cos\theta_{\bar{x}_{s,t}} \cos\theta_{\bar{x}_{u,v}}}{[r(\bar{x}_{s,t}, \bar{x}_{u,v})]^2} Vis(\bar{x}_{s,t}, \bar{x}_{u,v}) A_j(u,v).$$

where $\bar{x}_{s,t}$ and $\bar{x}_{u,v}$ are the points on the surface element i and j, $\theta_{\bar{x}_{s,t}}$ and $\theta\_\bar{x}_{u,v}$ are respectively the angles made by the surface normals at the above points with the line joining them, $r(.,.)$ is distance and $Vis(.,.)$ is the visibility between the points $\bar{x}_{s,t}$ and $\bar{x}_{u,v}$, and $A_j(u,v)$ is the area function of the surface j and has the following expression:

$$A_j(u,v) = \left\| \frac{\delta \bar{x}_{u,v}}{\delta u} \times \frac{\delta \bar{x}_{u,v}}{\delta v} \right\|$$

One of the widely followed approach of solving this integral equation is the projection of the continuous equation onto a finite set of basis functions. This projection gives rise to a discrete system of linear equations which can be solved to compute an approximation of the unknown radiosity function. As in any numerical method, there is bound to be a difference between the computed solution and actual solution. One wants that this difference (or *error*) remains smaller than a predefined amount, called *threshold*. If under the given setup the error is not under this threshold then one has to refine the setup and incrementally recompute the solution. However, error computation requires the knowledge of the actual solution which in general does not exist. Hence, instead of trying to compute the actual error, one aims to compute an estimate of its upper bound. Though theoretical expressions of error estimates for integral equation solution methods exist [3], they do not lend themselves to practical use. That is why the adaptive refinement of radiosity via error estimation has not been a common practice. More recently we have come across the interesting work of Lischinski *et al* [1] where they have proposed a practical method of estimating this error. They have applied this to the constant radiosity computation method, and have demonstrated that the error estimation approach can lead to an improved method for computing radiosity solution. Our effort in this paper has been to extend such error estimation work to linear radiosity computation.

We shall here briefly discuss the implication of the terms constant or linear radiosity. As said earlier, radiosity solution proceeds by projection onto a set of basis functions. Piecewise polynomial functions are oftenly used for this projection. Among the various polynomials, piecewise constants, the lowest degree polynomials, have been the widely used basis functions[4] for radiosity computations. Use of piecewise constants leads to constant radiosity solution over surface elements. Radiosity functions are in general continuous and it is well known that piecewise constants make poor approximations to a continuous function. In general, higher the degree of polynomial the better is the approximation. However, the expenses involved in the projection and in subsequent solution also grow with the degree of polynomial. Though there have been various attempts with piecewise polynomials of different degrees [5, 6, 7, 8, 2], it is not very clear what is the most optimal degree of choice for radiosity computation. In this paper we have preferred to use a piecewise linear basis set. A piecewise linear basis set leads to linear (more correctly, bi-linear) radiosity over surface elements. Thus, in this paper we describe a method for efficiently estimating the error between this computed linear function and the actual radiosity function and using this error estimate we propose an adaptive refinement strategy for improving the solution.

The key to estimation of error is the computation of the lower and the upper bound of the radiosity function over the elements. Lischinski *et al* [1] have chosen to use the lower and upper bounds of the form-factor between the elements, *i.e.* the double integral of the kernel function of the integral equation, to compute the bounds of the radiosity function. In this paper we have chosen, instead, to use the lower and upper bounds of the kernel function itself for the computation of the radiosity bounds. Particularly, we use the linear (perhaps better called *tetra-linear*) upper and lower bounds of the kernel function. Approaching in this fashion, we reduce the subsequent complexity of the projection process. In fact, in our method the effort involved in the bound computation process is very much compensated by the gain during the projection process. Our initial findings show that the resulting linear radiosity method promises to outperform the constant radiosity method in the overall cost and quality measure.

The organisation of the paper is as follows. We first briefly describe the basis functions and the projection method. Then we describe the method of computing the kernel bounds. We follow it with the algorithm for computing radiosity bounds from the kernel bounds. We then derive the expression for the estimation of error from the bounded radiosity values and the expression of a refinement indicator to carry out error-driven refinement in the context of hierarchical radiosity. Finally we show some results of the application of our algorithm to two simple environments. We compare the results of the linear radiosity method with that of the constant radiosity to emphasize the improvements.

2 Basis Functions and Projection

The work presented in this paper is based on the use of Legendre polynomials (polynomials of degree 0 and 1) as the piecewise linear basis set. That means we use two 1D basis functions

$$N_0(u) = \frac{1}{\sqrt{2}} \quad \text{and} \quad N_1(u) = \sqrt{\frac{3}{2}}u,$$

which are only defined over the parametric domain $-1 \leq u \leq +1$ and undefined outside.

These functions are orthonormal and the 2D function set

$$\left\{\mathbf{N}_k(u,v) \mid k = 1 \ldots 2^2,\ \mathbf{N}_1(u,v) = N_0(u)N_0(v), \ldots, \mathbf{N}_4(u,v) = N_1(u)N_1(v)\right\}, \quad (2)$$

formed by combining these functions, is also orthonormal. If we assume that u, v in the range $[-1, +1]$, are the parameters defined over a surface element, then we can construct the following orthonormal basis set for the whole environment:

$$\left\{\mathbf{N}_k^{(i)}(u,v) \mid k = 1 \ldots 2^2\ i = 1 \ldots n\right\} \quad (3)$$

where $\left\{\mathbf{N}_k^{(i)}(u,v)|k=1\ldots 2^2\right\}$ is the basis set over the surface element i, and n is the total number of elements.
Projecting the radiosity equation given in equation 1 on this basis we will get the following linear equations:

$$\begin{aligned}
B_k^{(i)} &= E_k^{(i)} + \rho^{(i)} \sum_{j=1}^{n} \sum_{l=1}^{2^2} K_{k,l}^{(i\leftarrow j)} B_l^{(j)} \quad \text{for} \quad k = 1\ldots 2^2 \text{ and } i = 1\ldots n \qquad (4)\\
&\text{where} \quad E_k^{(i)} = \int_{-1,-1}^{1,1} E^{(i)}(s,t)\mathbf{N}_k^{(i)}(s,t)dsdt \\
&\text{and} \quad K_{k,l}^{(i\leftarrow j)} = \int_{-1,-1}^{1,1}\int_{-1,-1}^{1,1} K^{(i\leftarrow j)}(s,t,u,v)\mathbf{N}_l^{(j)}(u,v)\mathbf{N}_k^{(i)}(s,t)dudvdsdt.
\end{aligned}$$

The equations can be set up by evaluating $E_k^{(i)}$ and $K_{k,l}^{(i\leftarrow j)}$ values. From the solution of these equations, *i.e.* $B_k^{(i)}$'s, one can construct an approximation to the unknown radiosity function as

$$B^{(i)}(s,t) \approx B_{computed}^{(i)}(s,t) = \sum_k B_k^{(i)}\mathbf{N}_k^{(i)}(s,t).$$

As the basis functions are polynomials of degree ≤ 1 the resulting approximation $B_{computed}^{(i)}()$ will be a bilinear function over the surface element i.

Using the coefficients $K_{k,l}^{(i\leftarrow j)}$ and the basis functions we can also set up an approximation to the kernel function.

$$K^{(i\leftarrow j)}(s,t,u,v) \approx \sum_k \sum_l K_{k,l}^{(i\leftarrow j)}\mathbf{N}_k^{(i)}(s,t)\mathbf{N}_l^{(j)}(u,v) \qquad (5)$$

Better is this approximation, more accurate is the radiosity solution. In the limit if the approximation can be made exact then we can have the exact radiosity function. The kernel functions defined between a pair of surfaces of the environment is so complicated that to get its exact expansion as given in equation 5, we may have to breakup the surfaces in the environment to infinitesimal elements. This will lead to a infinite linear system and hence will give rise to an impracticable solution method.

3 Computation of Linear Kernel Bounds

We have said in the beginning that the key to the error computation is the computation of the radiosity bounds and we propose to compute these bounds using the bounds of the kernel function. In this section we describe the method for computing the bounds of the kernel function, *i.e.* we wish to compute two tetra-linear functions which will completely bound the kernel function.

In the above paragraph we indicated that if we had an integral equation whose kernel can have an exact expansion using a finite set of basis function then the resulting the solution of the resulting linear system will be exact value for the unknown function. Here we shall find two such kernels and using them we shall compute the exact solution of the integral equation.

These two functions are $\underline{K}(s,t)$ and $\overline{K}(s,t)$, and are defined as follows:

$$\begin{aligned}
\underline{K}^{(i\leftarrow j)}(s,t) &= \sum_{k=1}^{2^2}\sum_{l=1}^{2^2} \underline{K}_{k,l}^{(i\leftarrow j)} \mathbf{N}_k^{(i)}(s,t)\mathbf{N}_l^{(j)}(u,v) \\
\text{and } \overline{K}^{(i\leftarrow j)}(s,t) &= \sum_{k=1}^{2^2}\sum_{l=1}^{2^2} \overline{K}_{k,l}^{(i\leftarrow j)} \mathbf{N}_k^{(i)}(s,t)\mathbf{N}_l^{(j)}(u,v) \qquad (6) \\
\text{where} \quad & \underline{K}^{(i\leftarrow j)}(s,t) \leq K(s,t) \text{ and } \overline{K}^{(i\leftarrow j)}(s,t) \geq K(s,t) \ \ \forall(s,t).
\end{aligned}$$

If we substitute them in the equation 1 and the resulting integral equation will be as follows:

$$\begin{aligned}
\underline{B}^{(i)}(s,t) &= E^{(i)}(s,t) + \rho^{(i)} \sum_{j=1}^{n} \int\int \underline{K}^{(i\leftarrow j)}(s,t,u,v)\underline{B}^{(j)}(u,v)dudv, \\
\overline{B}^{(i)}(s,t) &= E^{(i)}(s,t) + \rho^{(i)} \sum_{j=1}^{n} \int\int \overline{K}^{(i\leftarrow j)}(s,t,u,v)\overline{B}^{(j)}(u,v)dudv. \quad (7)
\end{aligned}$$

If solution exist for each of these integral equations then we can solve them by the projection method. The solution will give us two functions $\underline{B}^{(i)}(s,t)$ and $\overline{B}^{(i)}(s,t)$. They will be bilinear and because of the definitions in equations 6 and 7 they will have the following property:

$$\underline{B}^{(i)}(s,t) \leq B^{(i)}(s,t) \text{ and } \overline{B}^{(i)}(s,t) \geq B^{(i)}(s,t).$$

Or in other words they will be the linear bounds of the actual radiosity solution.

The important factor now is that we can compute the bounds only if solutions to the substituted integral equations exist. As $\underline{K}^{(i\leftarrow j)}(s,t,u,v) \leq K^{(i\leftarrow j)}(s,t,u,v)$, if there is a solution to the integral equation with K as kernel then there will be a solution to the integral equation with $\underline{K}$ as kernel. But same cannot be true for the integral equation with $\overline{K}$ as kernel. In the later section we shall bring in some transformation to the projected linear system of this integral equation to arrive at a solution.

4 Computation of $\underline{K}$ and $\overline{K}$

We shall now proceed to compute the kernel functions given in equation 6. The method presented here has a similar flavor of the kernel approximation principle used in the *oracle* process of [7]. The *oracle* approximates the kernel between an interacting surface element pair and used the magnitude of error in this approximation to decide on whether the link can be established or not. In a similar fashion, we first find out a linear approximation of the kernel. Instead of trying to find out the error in the approximation, we go on to determine the maximum and minimum deviation of the actual kernel function from this approximation and use these deviations and the approximation to derive the kernel bounds.

The various steps of computation of kernel bounds are follows:

Step I Computation of linear approximation of the kernel:
The emphasis in this step is to find a linear approximation with minimal effort. Finding a linear approximation of a 1D function requires at least the evaluation of the function at an arbitrary pair of non-coincident points. Extending this to 4D kernel will involve the evaluation of the kernel at 2^4 points. So we evaluate the kernel at 2^4 points and set up a linear system of the following form

$$\hat{K}(s_i, t_i, u_i, v_i) = \sum_k \sum_l K_{k,l} \mathbf{N}_k(s_i, t_i) \mathbf{N}_l(u_i, v_i) \qquad \text{for } i = 1 \ldots 2^4 \quad (8)$$

Solution of this system will give the values of $K_{k,l}$.
Choosing 2^4 points amounts to choosing 2 noncoincident points each in the parametric domains of s, u, v and t. We have chosen the extremes of their parametric domain *i.e.* -1 and +1 as the required points. Thus we have the necessary 16 kernel evaluation points as:
{(-1,-1,-1,-1),...,(1,1,1,1)}. Gortler *et al* [7] used Gauss quadrature points for the polynomial approximation in their oracle. This choice facilitated the subsequent Gauss quadrature for the evaluation of kernel coefficients. As we are making sure that the kernel functions are linear, we do not have to perform the Gauss quadrature for the evaluation of those coefficients. Further, choice of the extreme points of the parametric domain may prove to be better because the linear kernel function passing though them seems to be an extrema (*i.e.* either a minima or a maxima). We compute the coefficients $K_{k,l}$ by solving the linear system in equation 8.
If the kernel at any of the evaluated points is singular then its reevaluated by shifting the position of that point.

Step II Compute the maximum and minimum kernel deviation:
Using a searching technique [9] we compute the maximum and minimum deviation, respectively d_{max} and d_{min}, of the original kernel function from $\hat{K}(s, t, u, v)$.

Step III Compute the bounds of the kernel:
We define our minimal and maximal kernel functions as

$$\underline{K}(s, t, u, v) = \hat{K}(s, t, u, v) - d_{min} \;,\; \overline{K}(s, t, u, v) = \hat{K}(s, t, u, v) + d_{max}.$$

Using this definition we can derive for each of them an expansion form similar to equation 5

$$\begin{aligned}\underline{K}(s,t,u,v) &= \sum_k \sum_l \underline{K}_{k,l}\mathbf{N}_k(s,t)\mathbf{N}_l(u,v)\\ \text{and } \overline{K}(s,t,u,v) &= \sum_k \sum_l \overline{K}_{k,l}\mathbf{N}_k(s,t)\mathbf{N}_l(u,v)\end{aligned}$$

where

$$\underline{K}_{k,l} = \begin{cases} K_{1,1} - \Delta K_{min} & \text{iff } (k = l = 1), \\ K_{k,l} & \text{otherwise.} \end{cases} \qquad \overline{K}_{k,l} = \begin{cases} K_{1,1} + \Delta K_{max} & \text{iff } (k = l = 1), \\ K_{k,l} & \text{otherwise.} \end{cases} \tag{9}$$

and $\Delta K_{min} = \frac{d_{min}}{\mathbf{N}_1(s,t)\mathbf{N}_1(u,v)} = 4 * d_{min}$ (from definition of basis function in equation 3) and similarly $\Delta K_{max} = 4 * d_{max}$.

So if these kernel bounds are used in the radiosity equation then the projection of the resulting equation now becomes trivial. Furthermore, another important consequence of the above derivation is that the difference between these two bounds of the kernel, $\Delta K(s,t,u,v) = (\Delta K_{min} + \Delta K_{max})/4 = \Delta K$, is independent of (s,t) and (u,v).

We bring to attention that such computation of bounds may give rise to a certain problem. The kernel of the radiosity equation is always nonnegative. However, the lower bound computed in the above fashion can lead to a function which is negative in some part of its domain and hence is not acceptable. Though we cannot avoid such happening we must detect it and take corrective measures. Because of the linear nature of the function one can detect this by checking the value of $K_{min}(s,t,u,v)$ at its parametric corner points. We do not at this moment know what best corrective measure must be taken. However, we take a very simple measure. It is: we redo the bound computation by switching over the degree of the kernel approximation from linear to constant at the *step I* and then we proceed as if we are dealing with linear function. We must emphasize that this event happens very infrequently, so should not cast any doubt on the usefulness of the linear approximation.

5 Radiosity Bounds

We have shown in the previous section that if we use the kernel bounds, $\underline{K}$ and $\overline{K}$, in the radiosity equation then the resulting radiosity will be the upper and lower bounds of the actual radiosity function. The form of these functions has been chosen in such a way that the projection follows without much effort. That means we can right away proceed, without any computation effort, to set up the linear systems

$$\overline{B}_k^{(i)} = E_k^{(i)} + \sum_{j=1}^{n} \sum_l \overline{B}_l^{(j)} \overline{K}_{k,l}^{(i\leftarrow j)} \text{ and } \underline{B}_k^{(i)} = E_k^{(i)} + \sum_{j=1}^{n} \sum_l \underline{B}_l^{(j)} \underline{K}_{k,l}^{(i\leftarrow j)} \tag{10}$$

The $\overline{B}_k^{(i)}$'s and $\underline{B}_k^{(i)}$'s in the above expression are the unknown expansion coefficients of $\underline{B}(s,t)$, the lower bound and $\overline{B}(s,t)$, the upper bound of the radiosity function, *i.e.*

$$\overline{B}^{(i)}(s,t) \quad = \quad \sum_k \overline{B}_k^{(i)} \mathbf{N}_k^{(i)}(s,t) \quad \text{and} \quad \underline{B}^{(i)}(s,t) = \sum_k \underline{B}_k^{(i)} \mathbf{N}_k^{(i)}(s,t)$$

All we have to do now is to write an algorithm to compute the unknown $\overline{B}_k^{(i)}$s and $\underline{B}_k^{(i)}$s.

5.1 Algorithm for Computing Radiosity Bounds

In [1] we have seen the adaptation of both the standard full matrix [4] and hierarchical radiosity [10] methods to the computation of the constant radiosity bounds. Here we shall extend the method of [1] to support the computation of linear radiosity bounds.

Before discussing the extension, we must see what fundamental changes are brought in by the use of *piecewise linear* basis functions in place of *piecewise constant* ones.

- Over each surface element we have 2^2 basis functions instead of 1 basis function in *constant* case and one of these 2^2, *i.e.* $\mathbf{N}_1()$, is exactly same as the basis function used in *constant* case. Consequently there are 2^2 radiosity coefficients in the *linear* case. If we set to zero all except one ($i.e. B_1$) of the radiosity coefficients of each surface element then we shall get the piecewise constant approximation of the environment radiosity. Thus these other coefficients actually represent the linear variation of the radiosity function from this constant radiosity over an element.

- Over each interacting surface element pair, there are 2^4 basis functions instead of 1 in *constant* case. Again, as in the radiosity coefficients, one of the coefficients *i.e.* $K_{1,1}$, is exactly same as the coefficient in *constant* case. Thus all the coefficients define the various deviations of the kernel function from a constant kernel determined by the first coefficients.

From this above, we understand intuitively that as in the *constant* case the convergence of the iterative solution of the radiosity system (equation 4) depends on the fact that $\sum_j K_{1,1}^{(i\leftarrow j)} \leq 1$. As the condition $\sum_{j=1}^n \underline{K}_{1,1}^{(i\leftarrow j)} \leq 1$ is trivially satisfied, the method for full matrix constant radiosity bound computation can be used for *linear* bound computation without any modification. For the upper bound computation, a little change is required to take care of the additional kernel elements. As in [1], for each node element i we zero out all the $\overline{K}_{k,l}^{(i\leftarrow j)}$ coefficients corresponding to the dimmer elements till the condition $\left(\sum_j \overline{K}_{1,1}^{(i\leftarrow j)} \leq 1\right)$ is satisfied. If we permute the columns such that the non-zero

```
GatherLowerBounds(node,B̲)
{
foreach link ∈ node.links do
  for k = 1...2^2 do
    B̲_k += node.ρ * Σ_l K̲_{k,l} * link.source.B̲_l
if IsLeaf(node) then
  for k = 1...2^2 do node.B̲_k = B̲_k + node.E_k
else
  foreach child ∈ node.child do
    ConstructChildBoundFromParentBound(child, B̲, newB̲)
    GatherLowerBounds(child, newB̲)
  ConstructLowerBoundFromChildrenBounds(node.B̲, child_1.B̲, child_2.B̲, ...)
}
```

Figure 1: Gathering Lower Bounds.

$\overline{K}_{k,l}^{(i\leftarrow j)}$s are in the beginning of the row then we can write the expression for the radiosity coefficients of the i-th element during any iteration step as:

$$\overline{B}_k^{(i)} = E_k^{(i)} + \rho^{(i)} \left[\sum_{j=1}^{m} \sum_{l} \overline{K}_{k,l}^{(i\leftarrow j)} B_l^{(j)} + \frac{\left(1 - \sum_{j=1}^{m} \overline{K}_{1,1}^{(i\leftarrow j)}\right)}{\overline{K}_{1,1}^{(i\leftarrow (m+1))}} \sum_{l} \overline{K}_{k,l}^{(i\leftarrow (m+1))} B_l^{(m+1)} \right] \quad (11)$$

where m is the largest item in the permuted row of i such that $\sum_{j=1}^{m} \overline{K}_{1,1}^{(i\leftarrow j)} \leq 1$.

We shall now consider the computation of linear radiosity bounds in the hierarchical framework. The two main operations in the hierarchical solution method are: gathering at nodes and *push/pull* operation. Lower bound gathering at the nodes of the hierarchy are same as that in [1]. The upper bound gathering and the associated *push/pull* operations require special attention.

We saw above, in the full matrix computation of upper bound, the convergence of the iterative solution requires ordering of contributors before any gathering is done. One must do the similar ordering while gathering at the nodes of the hierarchy. For any node element in a hierarchy, the contributors are not only the ones specified in its links but also all those specified in the links of its parent and its ancestor nodes. That is why the algorithm in [1] collects all such links in a list called *contribList*. We have to do the same thing. But it is not enough. We must take note that the kernel function between the

```
GatherUpperBounds(node,contribList)
{
foreach link ∈ node.links do
  add node and link to contribList
if IsLeaf(node) then
  KSum = 0
  for k = 1...2^2 do node.B̄_k = node.emission_k
  CreateNewSortedList(contribList, NewContribList)
  foreach pair (p_node, p_link) ∈ NewContribList do
    ConstructKernelFromParentKernel(p_node.K̄, p_link.source, node, newK̄)
    if KSum + newK̄_{1,1} ≤ 1 then
      KSum += newK̄_{1,1}
      for k = 1...2^2 do
        node.B̄_k += node.ρ * Σ_l newK̄_{k,l} * p_link.source.B̄_l
    else
      factor = (1 − KSum) / newK̄_{1,1}
      for k = 1...2^2 do
        node.B̄_k += node.ρ * factor * Σ_l newK̄_{k,l} * p_link.source.B̄_l
      break
else
  foreach child ∈ node.child do
    GatherUpperBounds(child, contribList)
  ConstructUpperBoundFromChildrenBounds(node.B̄, child[1].B̄, child[2].B̄, ...)
}
```

Figure 2: Gathering Upper Bounds.

parent/ancestor element and a source is not the same as that between the child element and the source, because there is now a change in the domain of the kernel function. We are using a parametric expression for the kernel function. A parametric expression changes with the change of the parametric domain. So we must carry out the re-parameterisation of this kernel function. We take care of this by introducing a function called *ConstructKernelFromParentKernel()*.

The *push* operation, that we know of in hierarchical radiosity, is only done in the case of lower bound computation. The pushing operation also amounts to a re-parameterisation of the radiosity function defined in parent domain, to get the function defined in child's domain. This re-parameterisation is same as applying a push filter as done in the wavelet radiosity method[7].

The *pull* operation for lower/upper bound radiosity can be viewed as constructing the minimum/maximum bound of the radiosity function resulting from the combination of the child functions. So this can be performed exactly as was done for the kernel bound computation. As the child functions are also linear, in this case finding the minimum/maximum deviation is much simpler. Here again, we check for the possibility of lower radiosity bound becoming negative at the extremities and on its detection switch over to constant bounds as the corrective measure. The routines *ConstructLowerBoundFromChildrenBounds()* and *ConstructUpperBoundFromChildrenBounds()* carry out this pulling operation.

Now we have all the modification necessary for writing a gathering algorithm for the lower and upper radiosity bound in a hierarchical framework. We have given these algorithms in figures 1 and 2.

6 Error Norms

One of the main contributions of [1] is relating the error in the computed radiosity to the upper and lower bounds of the radiosity. Assuming that we are taking out computed radiosity as the average of the lower and upper bound radiosity functions then we can use the same relationship as [1] and derive below a quantitative error estimate over each surface element.

$$\epsilon^{(i)} \leq \frac{1}{2}\left\|\overline{B}^{(i)}(s,t) - \underline{B}^{(i)}(s,t)\right\| \leq \frac{1}{2}\sum_k \left|\overline{B}_k^{(i)} - \underline{B}_k^{(i)}\right| \left\|\mathbf{N}_k^{(i)}(s,t)\right\|$$

where $\|.\|$ is a functional norm. Once we decide which norm to use, for that norm we can precalculate the $\|\mathbf{N}_k^{(i)}(s,t)\|$ values. Using them we can compute the upper bound of the error.

7 Error-Driven Refinement

The error norm computed above gives us an estimate of the error in the computed radiosity function and thus indicates if the solution is acceptable or requires further improvement. If we find that the solution is not acceptable then we must find the strategy to carry out the refinement so as to get the optimal effect. In this section we discuss such a strategy.

The radiosity over an element is the result of the gathering over the various links which connect it to other elements. So any improvement in the solution can only be carried out by refining the links. So here we must address two questions. First, which of the links need refinement ? Second, how do we refine the link ? Refining a link means subdividing one of the elements of the link. So, in this context the second question becomes: which of the two elements of a link should be subdivided?

In order to be able to decide which of the links to refine, now we must turn our attention to the error associated with each individual link. If $\epsilon^{(i \leftarrow j)}(s,t)$ is the error function over element i due to the gathering link connecting it to element j then its expression will be:

$$\begin{aligned}
\epsilon^{(i \leftarrow j)}(s,t) &\leq \frac{1}{2}\left(\overline{B}^{(i \leftarrow j)}(s,t) - \underline{B}_l^{(i \leftarrow j)}(s,t)\right) \\
&= \frac{1}{2}\left[\int \overline{K}^{(i \leftarrow j)}(s,t,u,v)\overline{B}^{(j)}(u,v)dudv - \right. \\
&\qquad\qquad \left. \int \underline{K}^{(i \leftarrow j)}(s,t,u,v)\underline{B}^{(j)}(u,v)dudv\right]
\end{aligned}$$

adding and subtracting $\int \underline{K}^{(i \leftarrow j)}(s,t,u,v)\overline{B}^{(j)}(u,v)dudv$ on the right we get

$$\begin{aligned}
&= \frac{1}{2}\int \left[\overline{K}^{(i \leftarrow j)}(s,t,u,v) - \underline{K}^{(i \leftarrow j)}(s,t,u,v)\right]\overline{B}^{(j)}(u,v)dudv \\
&\quad + \frac{1}{2}\int \left[\overline{B}^{(j)}(u,v) - \underline{B}^{(j)}(u,v)\right]\underline{K}^{(i \leftarrow j)}(s,t,u,v)dudv
\end{aligned}$$

In the above we have an expression of the error due to the link $i \leftarrow j$ as a sum of two terms: the 1st term is due to the error in the kernel approximation between i and j, and the 2nd term is due to the error in the radiosity of the element j. Written in this fashion makes the following points clear:

- The link error is nonzero even when the kernel approximation is exact *i.e.* even when
$$\left(\overline{K}^{(i \leftarrow j)}(s,t,u,v) - \underline{K}^{(i \leftarrow j)}(s,t,u,v)\right) = 0.$$
- If we decide on a refinment based on the magnitude of $\epsilon^{(i \leftarrow j)}$ the subsequent solution may not at all give any reduction in the magnitude of $\epsilon^{(i \leftarrow j)}$. It is because, refinement of a link can at best improve the kernel

approximation. If the kernel approximation is already correct the refinement cannot do anything better.

From these statements we infer that though the full expression of $\epsilon^{(i\leftarrow j)}$ is not a good indicator for refinement, the 1st term of its expression can serve the purpose. So here we derive a quantity $\alpha^{(i\leftarrow j)}$, which we shall call *refinement indicator* of a given link, by integrating the above 1st term over the element i, *i.e*

$$
\begin{aligned}
\alpha^{(i\leftarrow j)} &= \frac{1}{2}\int_x \int_{u,v} \left[\overline{K}^{(i\leftarrow j)}(s,t,u,v) - \underline{K}^{(i\leftarrow j)}(s,t,u,v)\right] \overline{B}^{(j)}(u,v)dudvdx \\
&= \frac{1}{2}\text{Area}_i \Delta K^{(i\leftarrow j)} \sum_k \overline{B}_k^{(j)} \int_{u,v} N_k^{(j)}(u,v)dudv = \text{Area}_i \Delta K^{(i\leftarrow j)} \overline{B}_1^{(j)}.
\end{aligned}
$$

Thus we can refine all those links for which $\alpha^{(i\leftarrow j)}$ is more than a threshold. The solution obtained after this refinement is most likely to be an improvement over the current solution.

Now we shall address here the second question of the refinement, *i.e.* which element of the link should be subdivided so that the resulting refinement would give the maximum improvement? Intuitively, the subdivision which would reduce the magnitude of overall ΔK should be the choice. To find this we have taken a simple approach. We find the maximum variation $d^{(j)}$ of the kernel as a function of position on the element j and the variation $d^{(i)}$ as a function of position on the element i. If $d^{(i)} > d^{(j)}$ we subdivide i, otherwise we subdivide j.

8 Results

We have implemented the algorithm discussed above to compute the linear radiosity bounds. For efficiency comparison we have also implemented the error bound computation with constant basis functions. We have created various plots of the computed radiosity values along the dotted line drawn on the surfaces of the simple environments given in figure 3. We have used a very simple method of computing maximum and minimum deviation of the kernel *i.e.* d_{min} and d_{max}, by finding the minimum and maximum over the difference between the actual and approximated kernel at a finite number of random points.

In figures 4,5 we have shown the error bound computation results for the environment in figure 3(a) as a function of surface discretization. For this illustration the source dimensions have been kept very small so as to remove the source size dependent error in kernel approximation computation. In the legend of the figures, the levels indicate the discretization. Level 0 corresponds to the full surface as the element and in each subsequent level each element has been equally subdivided into 4 child elements. No attempt has been made to carry out adaptive refinement.

file=env.psfix (a) (b)

Figure 3: Simple test environments.

Figures 6 show the adaptive refinement of a similar environment but this time with a larger source, so that source also is subdivided if required. The resulting number of elements along the indicated line of the surface are: 106 for constant radiosity and 16 for linear radiosity.

Finally in figure 7 we show the results of the enclosure shown in figure 3(b). The outer most curves belong to the constant bounds, the inner most curve belongs to the actual radiosity and the curves in between belongs to linear bounds. This actual radiosity value has been obtained by carrying out a full matrix solution of the finely subdivided environment. This plot contains 5 sets of results, each set belonging to the radiosity over the dotted line on a surface, starting at the left the emitting source in the ceiling and moving along the dotted line in counterclockwise fashion. For comparison we give the execution time here. They are: 104 seconds for the constant bounds and 32 seconds for the linear bounds. The plots and the execution time respectively show that bounds computed in the linear case are tighter and the computation is faster compared to the constant case.

9 Discussion and Conclusion

We have described a hierarchical radiosity method for computing linear radiosity with tight upper and lower bounds for the actual radiosity function. We compute the radiosity bounds by creating linear upper and lower bounds to the radiosity kernel function. This computation is simple and fast. Further, we derived an *error indicator* to refine the inter-element links to reduce the error in the computed radiosity functions.

In this paper we have not brought occlusion into consideration. At the occlusion point kernel evaluates to zero. An immediate strategy to accommodate occlusion will be: for a link with occlusion (i) to switch over to constant kernel approximation, (ii) to compute the upper bound by evaluating kernel without

Figure 4: Upper and lower bounds of constant radiosity for environment in figure 3(a) at various levels of subdivision.

the occlusion, and (iii) to set the lower bounds to zero. This strategy will increase the ΔK values for such links and hence force their refinement in subsequent iterations. We have to get a feel for the performance of the method based on this strategy. We believe that, for environments with sparse occlusion or for the ones prediscretised along the shadow boundaries, our method will work without any problem and give superior performance compared to the method based strictly on constant basis. However, for highly occluded environments we may have to find out some other strategy to bound the kernels.

We have only made use of non-overlaping linear basis functions. Thus the computed radiosity functions over a surface element is not likely to have any continuity with that over the neighbouring elements even if the elements belonged to the same surface. Thus prior to rendering an image, this discontinuity must be resolved by a reprojection. Because of their conflict with the hierarchical advantages, overlapping basis functions are generally not used for the projection of the radiosity equation. It may be worthwhile to reexplore with them in the error-bounding setup.

References

[1] Dani Lischinski, Brian Smits, and Don Greenberg. Bounds and error es-

Figure 5: Upper and Lower bounds of linear radiosity for environment in figure 3(a) at various levels of subdivision.

Figure 6: Upper and Lower bounds with adaptive subdivision for figure 3(a).

Figure 7: Bounds in the enclosure in figure 3(b). The plot shows the actual radiosity and the bounds for constant and linear radiosity, along the dotted line of the scene.

timates for radiosity. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):67–74, July 1994.

[2] Harold R. Zatz. Galerkin radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):213–220, 1993.

[3] L. M. Delves and J. L. Mohamed. *Computational Methods for Integral Equations*. Cambridge University Press, 1985.

[4] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the interaction of light between diffuse surfaces. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):212–222, July 1984.

[5] Nelson L. Max and Michael J. Allison. Linear radiosity approximation using vertex to vertex form-factors. In David Kirk, editor, *Graphics Gems III*. Academic Press, Inc, Boston, 1992.

[6] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–216, May 1992.

[7] Steven Gortler, Peter Schroder, Michel F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):221–230, 1993.

[8] Roy Troutman and Nelson L. Max. Radiosity algorithms using higher order finite element methods. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):209–212, 1993.

[9] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

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