Solving the Classic Radiosity Equation Using Multigrid Techniques

Robert R. Lewis
bobl@cs.ubc.ca

University of British Columbia
Department of Computer Science

4 February 1992

## Abstract

We investigate the application of multigrid techniques to the solution of the "classic" radiosity equation. After overviews of the global illumination problem and of radiosity, we describe the latter's solution via multigrid methods.

An implementation of the multigrid algorithm presented here is able to solve the classic radiosity equation in about 50% of the time required by the more commonly-used Gauss-Seidel approach. Although few researchers currently use classic radiosity, we discuss possibilities for the adaption of multigrid methods to more recent radiosity solution techniques.

## 1 Introduction

In this paper, we'll investigate the application of multigrid techniques to the solution of the radiosity equation, one of the two major approaches to the solution of the global illumination problem. (The other is raytracing.)

After a brief overview of the problem itself, we'll review the radiosity method and then describe how we can apply multigrid methods to its solution.

We'll confine ourselves to what [Heck91] refers to as *classic radiosity*, the approach first disseminated to the computer graphics community in [Coh85]. A number of substantial improvements have been made to radiosity since then, but as our goal here is to investigate the usefulness of multigrid, the fact that we're not really on the forefront of radiosity research is of secondary relevance.

## 2 Global Illumination

The problem of global illumination is a longstanding one with aspects in fields as diverse as computer graphics, architecture, and mechanical engineering. The fundamental question of global illumination is this: given the set of physical characteristics describing an illuminated environment, how does the light distribute itself? The brief discussion of it we present here follows after [Heck91].

In order to make the problem tractable, we make a series of assumptions. The first is that we have a non-participatory environment. This restricts us to dealing with surfaces rather than volumes. The second assumption is that the surfaces are all opaque, so we don't worry about transmission.

From energy conservation, the outgoing intensity $I_o(\mathbf{x}, \mathbf{\Theta}_o)$ (units of {energy} $\{\text{area}\}^{-1}$ $\{\text{time}\}^{-1}$ $\{\text{solid angle}\}^{-1}$) at a point $\mathbf{x}$ in direction $\mathbf{\Theta}_o$ is given by

$$I_o(\mathbf{x}, \mathbf{\Theta}_o) = \frac{e(\mathbf{x}, \mathbf{\Theta}_o)}{\pi} + \int_{\mathbf{\Omega}} I_i(\mathbf{x}, \mathbf{\Theta}_i)\, \rho(\mathbf{x}, \mathbf{\Theta}_i, \mathbf{\Theta}_o) \cos\theta_i \, d\mathbf{\Theta}_i$$

where $\frac{e(\mathbf{x}, \mathbf{\Theta}_o)}{\pi}$ is the (outgoing) emissivity, $\mathbf{\Theta}_i = (\theta_i, \phi_i)$ is a direction on the hemisphere $\mathbf{\Omega}$, $I_i(\mathbf{x}, \mathbf{\Theta}_i)$

is the incoming intensity, and $\rho(\mathbf{x}, \mathbf{\Theta}_i, \mathbf{\Theta}_o)$ is the reflectivity of the surface.

From out assumption of a non-participating medium, the incoming intensity of a ray at $\mathbf{x}$ in direction $\mathbf{\Omega}$ is equal to the outgoing intensity of a ray that starts at $\mathbf{x}'$ in direction $\mathbf{\Omega}'$, so

$$I_i(\mathbf{x}, \mathbf{\Theta}_i) = v(\mathbf{x}, \mathbf{x}')\, I_o(\mathbf{x}', \mathbf{\Theta}'_o)$$

where $v(\mathbf{x}, \mathbf{x}')$, the visibility function, is

$$v(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \text{if there are no obstructions} \\ & \text{between } \mathbf{x} \text{ and } \mathbf{x}' \\ & \\ 0 & \text{otherwise} \end{cases}$$

$d\mathbf{\Theta}_i$, the solid angle subtended by an area $d\mathbf{x}'$ whose surface normal is inclined at an angle $\theta'_o$ away from $\mathbf{x} - \mathbf{x}'$ is given by

$$d\mathbf{\Theta}_i = \frac{d\mathbf{x}' \cos \theta'_o}{|\mathbf{x} - \mathbf{x}'|^2}$$

The next assumption we make is one of isotropicity: that all surfaces are diffuse emitters and reflectors. This leads to the following:

$$\begin{aligned} e(\mathbf{x}, \mathbf{\Theta}_o) &= e(\mathbf{x}) \\ I_o(\mathbf{x}, \mathbf{\Theta}_o) &= I_o(\mathbf{x}) \equiv \frac{u(\mathbf{x})}{\pi} \qquad (1) \\ \rho(\mathbf{x}, \mathbf{\Theta}_o) &= \frac{\rho_h(\mathbf{x})}{\pi} \end{aligned}$$

where (1) defines the *radiosity* $u(\mathbf{x})$. $\rho_h$, the hemispherical reflectivity, varies between 0 (perfect absorption) and 1 (perfect diffuse reflection).

Combining all of this, we get the fundamental integral equation for radiosity:

$$\begin{aligned} u(\mathbf{x}) \;\; = \;\; & e(\mathbf{x}) + \qquad (2) \\ & \rho_h(\mathbf{x}) \int_{\mathbf{\Gamma}} \frac{\cos \theta_i \, \cos \theta'_o}{\pi \, |\mathbf{x} - \mathbf{x}'|^2} v(\mathbf{x}, \mathbf{x}') u(\mathbf{x}') d\mathbf{x}' \end{aligned}$$

where the integration now takes place over the set of all surfaces $\mathbf{\Gamma}$.

With the obvious definitions, we can rewrite (2) in the form

$$u = f + \mathcal{K} u$$

where $\mathcal{K}$ is a linear integral operator. This identifies (2) as a Fredholm integral of the second kind.

# 3 Classic Radiosity

The classic ([Coh85]) way of solving (2) uses collocation. $\mathbf{\Gamma}$ is treated as a collection of $N$ patches, with radiosity $u_i$ constant on each patch $i$. If we then integrate over each patch $i$ and divide by the patch's area $A_i$, (2) becomes:

$$u_i = e_i + \rho_i \sum_{j=1}^{N} u_j F_{ij} \qquad (3)$$

where $e_i$ is the integral of $e(\mathbf{x})$ over patch $i$ and $F_{ij}$, the *form factor*, is the fraction of radiation given off by patch $j$ that reaches patch $i$. We have

$$F_{ij} = \frac{1}{A_i} \int_{\mathbf{\Gamma}_i} \int_{\mathbf{\Gamma}_j} \frac{\cos \theta_i \, \cos \theta'_j}{\pi \, |\mathbf{x} - \mathbf{x}'|^2} v(\mathbf{x}, \mathbf{x}')\, d\mathbf{x}' d\mathbf{x} \qquad (4)$$

where $\mathbf{\Gamma}_i$ is the area of patch $i$ and $\mathbf{\Gamma}_j$ is the area of patch $j$.

We can rearrange (3) in the form

$$\mathbf{A}\mathbf{u} = \mathbf{e}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 - \rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1 - \rho_2 F_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ -\rho_N F_{N1} & \cdots & \cdots & 1 - \rho_N F_{NN} \end{bmatrix}$$

and then solve it using any of the standard techniques for solving systems of linear equations. Like [Coh85], most classic radiosity researchers use(d) Gauss-Seidel iteration, which we'll refer to hereafter as "G-S".

Although this equation permits $F_{ii} \neq 0$, as long as our patches are flat or convex, they cannot illuminate themselves, so $diag(\mathbf{A}) = \mathbf{I}$, which is usually the case.

$F_{ij}$ is dimensionless, and it depends neither on the illumination nor on the position of an observer, only on the scene geometry. This is a big advantage of classic radiosity over other global illumination solution techniques such as raytracing: once we have $\mathbf{F}$, we can change the illumination or observer positions, and all we need do is re-solve (3).

# 4 A Radiosity Model Problem

Before going any further, let's define a simple radiosity problem:

> A small room has been built inside a unit cube. Light of uniform intensity enters isotropically through a $\frac{1}{2}$ by $\frac{1}{2}$ hole in the center of the ceiling. The walls have varying reflectivities:

| Wall | $\rho$ |
|---|---|
| $x = 0$ | 0.2 |
| $x = 1$ | 0.6 |
| $y = 0$ | 0.4 |
| $y = 1$ | 0.8 |
| $z = 0$ | 1.0 |
| $z = 1$ | 1.0 |

For reasons that will soon be apparent, we'll choose the number of patches $N$ to be $6 \cdot 4^l$ for some non-negative integer l. Table 4 shows some typical values, along with $N^2$, the resulting number of elements required in $\mathbf{A}$ and $\mathbf{F}$.

| $l$ | $N$ | $N^2$ |
|---|---|---|
| 0 | 6 | 36 |
| 1 | 24 | 576 |
| 2 | 96 | 9,216 |
| 3 | 384 | 147,456 |
| 4 | 1,536 | 2,359,296 |
| 5 | 6,144 | 37,748,736 |
| 6 | 24,576 | 603,979,776 |

Table 1: Matrix Sizes of the Scaled Radiosity Model Problem

So radiosity in its classic form places large demands upon memory and time. Indeed, computing $F_{ij}$ accurately and efficiently is a problem with its own literature (see [Baum89], for example).

Things wouldn't be so bad if $\mathbf{A}$ were sparse, but in the case of this model problem, it's not, as we can see from Figure 1, a density diagram of the $\mathbf{A}$ from our model problem with $N = 96$. The black regions correspond to places where $|a_{ij}| = 1$ (the diagonal) and the white regions correspond to places where $|a_{ij}| = 0$ ($i \neq j$, $i$ and $j$ on the same wall). The gray regions indicate $0 < |a_{ij}| < 1$.

The lack of sparsity is because our geometry dictates $v(\mathbf{x}, \mathbf{x}') \equiv 1$. Paradoxically, if we add more obstructing objects such as a table and chairs to the room, the matrix becomes sparser, although the number of surfaces increases. [Coh88] cites a 10% occupancy in a typical scene.

Figure 1 also shows that $\mathbf{A}$ is non-symmetric. This asymmetry is caused by varying reflectivity $\rho$.

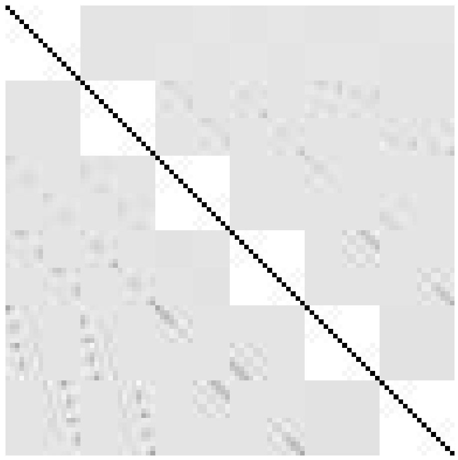

Figure 1: Density of $\mathbf{A}$ for Model Problem, Normalized to Largest (Diagonal) Element

# 5 Applying Multigrid to Radiosity

[Hack85] describes how multigrid methods (hereafter, "MG"), which were originally developed to solve elliptic PDE's, may be adapted to solve integral equations as well. In particular, Figure 2 shows his algorithm for solving Fredholm integrals of the second kind ($u = f + \mathcal{K}u$).

There are three notable differences between Figure 2 and the usual PDE MG algorithm:

- The smoothing step, which is often taken to be ($\nu_1$ =)1 or 2 G-S (typically) relaxations, is replaced by the very equation we're trying to solve, and only applied once (at line 15).
- $\gamma$, the number of recursive calls to itself that

```
 1: procedure mg(l, u, f);
 2: integer l;
 3: array u, f;
 4: begin
 5:     integer i;
 6:     array d, v;
 7:
 8:     if l = 0 then
 9:         u := (I - K)^{-1} f;
10:     else
11:     begin
12:         u := f + K u;
13:         d := I_h^H (u - K u - f);
14:         for i = 1,2 do
15:             mg(l - 1, v, d);
16:         u := u - I_H^h v;
17:     end
18: end
```

Figure 2: Pseudocode of Multigrid Algorithm for Radiosity (after [Hack85])

mg() makes, is fixed at 2 (at line 14).

- There is no post-smoothing ($\nu_2 = 0$).

In order for this algorithm to work, we need several things in addition to $\mathbf{K}$ and $f$. In particular, we need a multilevel gridding scheme, a way to compute $\mathbf{K}$, an initial $u$, a restriction operator $\mathbf{I}_h^H$, a prolongation operator $\mathbf{I}_H^h$, and a coarse grid solver for $(\mathbf{I}-\mathbf{K})^{-1}f$.

The choices we've made for each of these follow.

## 5.1 Gridding Scheme

For our model problem, we can assume each wall to be covered with a rectangular grid. At level $l$, there are $2^l \times 2^l$ patches along each wall, and with 6 walls. That's why we took $N = 6 \cdot 4^l$ in Table 4.

In PDE problems, we often deal with $u$ evaluated at various points on a hierarchy of grids. In radiosity, though, we must keep in mind that we're dealing with values integrated over patches, not points. At any level $l$, we need to evaluate $F_{ij}^l$ not just between two points, but between two patches.

## 5.2 Computing K

Since $K_{ij} = \rho_i F_{ij}$, computing $\mathbf{K}$ is equivalent to computing the form factor. Doing this accurately is not a trivial task, since $F_{ij}$ is a quadruple integral with a potentially vanishing kernel. Various schemes such as analytical ([Baum89]), hemicubes ([Coh85]), MG-like hierarchy ([Han91]), and raytracing ([Wall89]) have all been proposed.

For our purposes, we'll take a very simple and extremely approximate scheme and replace (4) by

$$F_{ij} = \frac{1}{\alpha} A_j \frac{\cos\theta_i \, \cos\theta_j'}{\pi \, |\mathbf{x}-\mathbf{x}'|^2} \, v(\mathbf{x},\mathbf{x}') \qquad (5)$$

where $\cos\theta_i$, $\cos\theta_j'$, $\mathbf{x}$, and $\mathbf{x}'$ are all measured at the center of their respective patches, $v(\mathbf{x},\mathbf{x}') = 1$ in our geometry, and $\alpha$ is chosen so that $\sum_{j=1}^N F_{ij} = 1$, which is also demanded by our geometry.

## 5.3 Initial $u$

MG convergence is usually pretty insensitive to the initial choice for $u$. Often, $u$ is initially set to all 0's. In this case, though, we can go one better and set $u = f$, which in fact would be the solution if the walls were non-reflective. Also, line 12 in Figure 2 shows that this is what we'd get if we started with $u = 0$ in the first place, so we're slightly speeding up the iterative process.

## 5.4 Restriction

When solving PDE's with multigrid, the choice of restriction and prolongation operators is often critical to the convergence of the solution.

For all $l > 0$, each patch on level $l-1$ corresponds to four patches on level $l$, so the first type of restriction that comes to mind is to take the mean of those four patches as shown in Figure 3.

## 5.5 Prolongation

The first kind of prolongation that suggests itself is injection, but our experience with injection in the

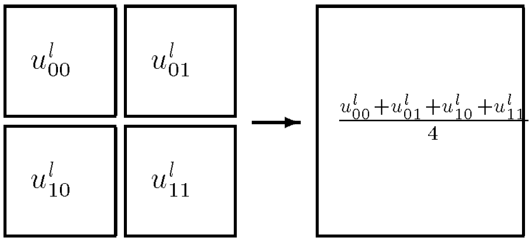


Figure 3: Multigrid Restriction

PDE case indicates that injection often gives rise to convergence problems.

We can do a more preferred kind of prolongation, bilinear interpolation, as shown in Figure 4. In this case, each of the four subpatches gets a value interpolated between that of its parent patch and three of the parent's neighbors.

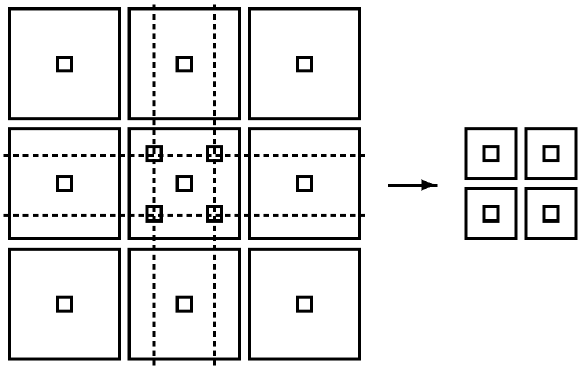

Figure 4: Multigrid Prolongation

The only problems with this scheme happen at edges and corners of walls. We don't want to use neighboring patches of adjoining walls, if they exist: too many interesting effects such as interreflection and changes in reflectivity happen there. That is: we *expect* discontinuities there in our solution, so we shouldn't smooth them over.

We have two alternatives:

- Inject the parent $u$ on the corners and linearly (one dimensionally) interpolate the parent $u$ with those of its two edge-neighbors on the edges.
- Extrapolate the value of the missing patch.

Given the general unreliability of extrapolation and the need for accuracy near edges and corners, we'll use the former [1].

### 5.6 The Coarse Grid Solver

In our case, this means a solver for $(\mathbf{I}-\mathbf{K})u = f$ where $(\mathbf{I}-\mathbf{K})$ is $6\times 6$. Just about any solver will do for this, even Gaussian elimination. In our case, though, we'll choose 20 G-S iterations, because a routine to do that is available, for reasons as we'll explain below.

## 6 Implementation

We have implemented this algorithm and an equivalent G-S solver in C on a Sun SPARCstation-2. Figure 5 shows the result for $l = 4$. The $x = 0$ (lowest reflectivity) wall is on the left. The $y = 0$ wall has been folded down so we can look inside the cube. As we would hope, both algorithms yield approximately identical solutions.

### 6.1 Performance

Table 2 shows performance figures of the MG and G-S solutions, and Figure 6 graphs these results. All times are in seconds and include both user and system time, the latter of which is always $< 2\%$ of the total.

Both MG and G-S stop when the maximum residual drops below $10^{-4}$, scaled to account for decreasing surface area (since, like $u$, the residual has dimensions of $\{\text{area}\}^{-1}$, we need to do this to obtain the same fractional accuracy at varying grid spacings).

The uncertainties arise because the number of iterations, $n_{iter}$ should be taken to be $\pm 1$ in all cases.

As often advertised, the number of iterations required for MG to converge is independent of the size of the problem. What's surprising, though, is that the same is pretty much true for G-S.

The reason why G-S does so well in this case is evident from Figure 1. $\mathbf{A}$ is strongly diagonally dominant

[1] We've actually tried using injection everywhere without interpolation here and it seems to work fine.

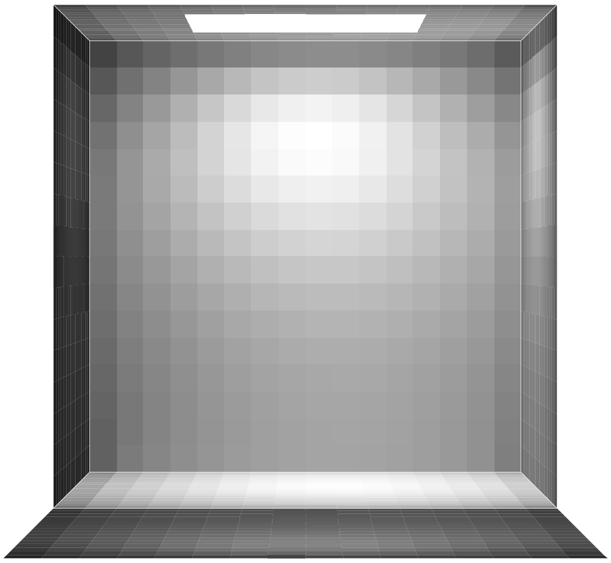

Figure 5: Solution of Model Radiosity Problem for $l = 4$ ($N = 1536$)

since $0 \leq F_{ij} \leq 1$, $\sum_{j=1}^{N} F_{ij} \leq 1$ and $0 \leq \rho_i \leq 1$ for all $i : 1 \leq i \leq N$.

In order to achieve these results, we made several modifications to the original algorithm in Figure 2:

- Contrary to [Hack85], we changed

  12: $u := f + \mathbf{K}u$;

  back into a G-S iteration for smoothing:

  12: $u :=$`gs(`$\mathbf{K}$`,` $u$`,` $f$`);`

  The reason for this is evident by noting that if, as in our case, $F_{ii} = 0$, the original form of line 12 is identical to Jacobi iteration, and the general improvement of G-S over Jacobi is pretty well established. (Even if $F_{ii} \neq 0$, G-S would probably still be a good choice.)

- Expanding the defect correction

  13: $d := \mathbf{I}_h^H (u - \mathbf{K}u - f)$;

  into

  13: $d := \mathbf{I}_h^H u - \mathbf{I}_h^H \mathbf{K}u - \mathbf{I}_h^H f$;

  and noting that, just as we precompute $\mathbf{K}$, we can also precompute $\mathbf{I}_h^H \mathbf{K}$ at the same time. This reduces the number of multiplications required for $\mathbf{I}_h^H \mathbf{K}u$ during defect computation by a factor of 4 at an increase of 25% in our memory requirements.

- We added the enhancement described in [Hack85] that foregoes smoothing on the first ($i = 1$ on line 15) of the two recursive `mg()` calls.

### 6.2 In-Memory vs. On-The-Fly Form Factor Evaluations

The largest $N$ we've dealt with so far is 1536 ($l = 4$). That's because $N^2$, 2359296, was the size of the

| $l$ | $N$ | Gauss-Seidel $n_{iter}$ | Time | Multigrid $n_{iter}$ | Time | $\frac{Time_{MG}}{Time_{G-S}}$ |
|---|---|---|---|---|---|---|
| 1 | 24 | 11 | $< 0.017 \pm 0.001$ | 4 | $< 0.017 \pm 0.004$ | $\sim 1$ |
| 2 | 96 | 11 | $0.1300 \pm 0.01$ | 4 | $0.10 \pm 0.03$ | 0.769 |
| 3 | 384 | 11 | $2.0 \pm 0.2$ | 4 | $1.0 \pm 0.3$ | 0.500 |
| 4 | 1536 | 12 | $35 \pm 3$ | 4 | $15 \pm 4$ | 0.429 |

Table 2: Performance of Multigrid vs. Gauss-Seidel on the Model Problem

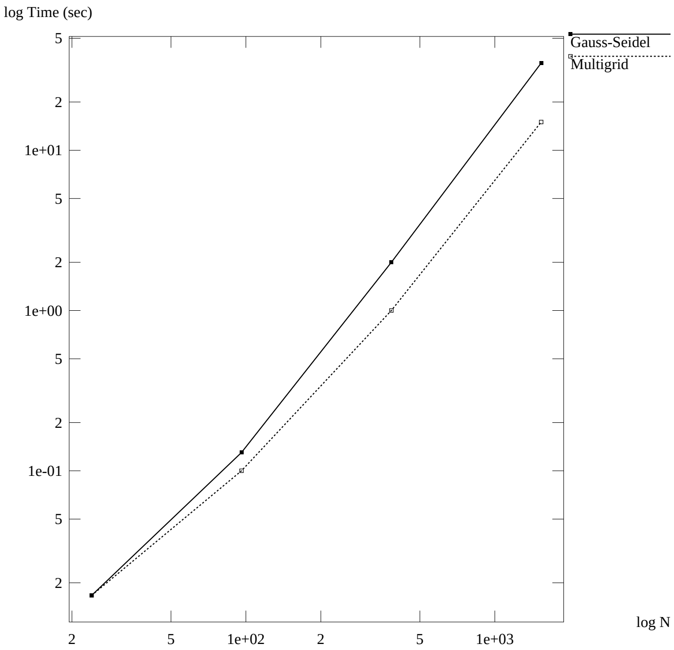


Figure 6: CPU Time Required by Gauss-Seidel and Multigrid vs. $N$

largest (single-precision) $\mathbf{K}$ we could fit in memory.

An earlier version of the program computed $\mathbf{K}u$ (that is, the form factors $F_{ij}$) on-the-fly. With it, we were able to reach $N$ as large as 24576 ($l = 6$), which would otherwise have required 2.4GB of memory!

We chose to go with an in-memory computation for several reasons:

- Even with the simplest possible form factor calculations, it took CPU-*hours* to get results for large $N$.
- Most other classic radiosity work computes $\mathbf{K}$ in-memory, which would indicate that other researchers consider the time to do on-the-fly $F_{ij}$ computation prohibitive. This is especially true as most of them have more involved and accurate form factor calculation techniques than (5).
- On-the-fly $F_{ij}$ computation prevented us from using the $\mathbf{I}_h^H\mathbf{K}$ speedup for MG. (Without it, MG had only a negligible advantage over G-S.)

## 7 Conclusions and Future Work

MG has a definite advantage over G-S in solving classic radiosity problems. These days, however, very few people do radiosity classically. In the past few years, a number of techniques have been developed that show large improvements over classic radiosity in other and more substantial ways than we've done here.

As [Hack85] mentions and as Figure 6 shows, both MG and G-S solutions are dominated by the $O(N^2)$ operations required to compute the $\mathbf{K}u$ product. It's intriguing to think about using MG to evaluate $\mathbf{K}u$ faster. [Brandt91] sketches some ways that this might be done, which time does not permit us to explore. The general idea is that given some tolerance $\epsilon$, find $i_k$, $j_k$, and $l_k$ for $0 < k < N_k$ such that

$$\left| \mathbf{K}u - \sum_{k=1}^{N_k} K_{i_k j_k}^{l_k} u_{j_k}^{l_k} \right| < \epsilon$$

for $N_k \ll N$.

In fact, we already have code that does something like this using on-the-fly $F_{ij}$ computation (as this method demands). Results are negative so far (about twice as slow as G-S), the probable reason being that we have not yet found suitable criteria for $l_k$ given $\epsilon$. We also need a better form factor calculation than (5).

Other researchers in computer graphics are already using MG-like techniques. [Han91] describe a technique they call "BF refinement" which is an adaptive MG solution.

## References


[Baum89] Baum, D. R., Rushmeier, H. E., and Winget, J. M. *Computer Graphics (SIGGRAPH '89 Proceedings)*, vol. 23, no. 3, p. 325, (July, 1989).

[Brandt91] Brandt, A. *Computer Physics Communications*, vol. 65, p. 24, (1991).

[Coh85] Cohen, M. F., and Greenberg, D. P. *Computer Graphics (SIGGRAPH '85 Proceedings)*, vol. 19, no. 4, p. 75, (August, 1985).

[Coh88] Cohen, M. F., Chen, S. E., Wallace, J. R., and Greenburg, D. P. *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol. 22, no. 4, p. 75, August, 1988.

[Hack85] Hackbusch, W. *Multi-Grid Methods and Applications*, (Springer-Verlag, Berlin, 1985).

[Han91] Hanrahan, P., Salzman, D., and Aupperle, L. *Computer Graphics (SIGGRAPH '91 Proceedings)* vol. 25, no. 4, p. 197, (July, 1991).

[Heck91] Heckbert, P. "Radiosity in Two and Three Dimensions", (Ph. D. Thesis, Computer Science Department, UC Berkeley, 1991).

[Wall89] Wallace, J. R., Elmquist, K. A., and Haines, E. A. *Computer Graphics (SIGGRAPH '89)*, vol. 23, no. 3, p. 315, (July, 1989).