An Empirical Comparison of Progressive and Wavelet Radiosity

Andrew J. Willmott and Paul S. Heckbert[1]

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract. This paper presents a comparison of basic progressive and wavelet radiosity algorithms. Several variants of each algorithm were run on a set of scenes at several parameter settings, and results were examined in terms of their error, speed, and memory consumption. We did not compare more advanced variations such as clustering or discontinuity meshing. Our results show that progressive radiosity with substructuring works fairly well for all scenes. Among wavelet methods, the Haar basis works best, while higher order methods suffer because of extreme memory consumption and because poor visibility handling causes discontinuous, blocky shadows.

1 Introduction

A number of variations on the radiosity method have been published in the last decade [4]. The algorithmic options are quite numerous. They include: matrix radiosity, progressive radiosity, wavelet radiosity, hemicube form factors, ray-traced form factors, discontinuity meshing, importance-driven radiosity, and clustering. Given all these options, a developer or user might ask:

- Which method is fastest? Which is most accurate?
- What wavelet basis function is best?
- Are discontinuities a problem for wavelet methods?
- Which method is best for shadows? For complex scenes?

The radiosity literature does not currently answer these questions in a very satisfying manner. While the papers introducing these techniques have demonstrated the strengths of each radiosity algorithm, and some have offered theoretical complexity analyses, radiosity algorithms have not been compared extensively in terms of actual speed and accuracy. The existing comparisons focus on matrix and progressive radiosity, and their variations [7, 19, 1]. To our knowledge, wavelet and progressive radiosity have not been thoroughly compared.

There seem to be several causes of the lack of algorithm comparison: implementing a radiosity algorithm properly is difficult; there is little free, public radiosity software available; comparing results is not as simple as measuring CPU time; and there are few incentives for performing large-scale empirical software tests in computer graphics.

In order to provide some hard data on the relative performance of various radiosity methods, to learn more about the strengths and weaknesses of these algorithms, and to develop a deeper understanding of the maths and physics of diffuse interreflection, we designed and carried out a

1. {ajw|ph}@cs.cmu.edu, http://www.cs.cmu.edu/~{ajw|ph}.

large array of empirical experiments.

Rather than piece together existing code or experiments, we re-implemented and re-ran the algorithms ourselves, since differences in programming style, compiler, computer, and test scene can cause large variations in running times. Re-implementing in a common code base allowed us to compare speeds of various algorithms directly. Such empirical comparison is valuable, since the theoretical performance of an algorithm is not always wholly representative. In the real world, constant factors and expected case behaviour are often more important than asymptotic, worst case behaviour.

To keep the project manageable, we did not attempt to test all of the algorithmic options in the literature. Discontinuity meshing, importance-driven radiosity, clustering, and distribution raytracing [17], for example, were neglected. Specular radiosity and participating media were also not examined. The accuracy of simulations was measured using a simple, view-independent error metric, rather than attempt to account for perception [10]. We feel that error metrics that model colour sensitivity and nonlinearity alone would not change our results significantly, but that metrics that model sensitivity to spatial frequencies would emphasize shadow problems and bring our objective error measure closer to a subjective one.

In our broader study, we selected matrix, progressive, and wavelet radiosity as the algorithms to compare, and designed a series of experiments to test the algorithms’ ability to handle interreflection, occlusion, complexity, and several other attributes. In the tests, around 10 methods were run on 7 different experimental scenes, each scene varied according to some parameter at, on average, 6 different settings. The parameters varied included geometric separation, reflectance, and the number of objects. In all, 500 different experimental runs were executed, generating a total of 0.5 gigabytes of data and requiring a week of CPU time. The radiosity and testing software comprises 40,000 lines of C++ code. A more complete report on the full study is available [18]; in this paper we summarize the most important results for progressive and wavelet radiosity.

2 Experimental Method

Designing the tests involved a choice of which algorithms to implement, what scenes to run them on, and how to collect and compare results. We discuss each of these issues in turn below.

2.1 Implementation

The two algorithms examined in this paper:

Progressive radiosity iteratively “shoots” light from the brightest light sources and reflective surfaces, effectively computing one column of the form factor matrix at a time [3]. This method can be made adaptive by the introduction of a two-level mesh; the coarser patches shoot light to a finer set of adaptively refined elements [2]. This process is known as substructuring.

Wavelet Radiosity employs multilevel meshes to represent the radiosity function, and allows inter-patch interactions to take place between arbitrary levels of the mesh hierarchy [9,8]. Theoretically, wavelet algorithms have cost $O(k^2 + n)$, where k is the number of input surfaces, and n is the number of elements.

Within the wavelet framework, radiosity functions can be represented with a variety of different basis functions. We implemented the methods most widely discussed in the literature: the Haar basis [15], flatlets of order 2 and 3 (F2, F3) [12], and multiwavelets, also of order 2 and 3 (M2, M3) [8]. The lower-order methods are easier to integrate and require fewer coefficients per link, while the higher order methods create sparser approximations to the kernel. We use multigridding with brightness-weighted refinement.

In the remainder of this section we discuss the most salient of our implementation choices. Further details can be found in [18].

Control Algorithms. For progressive radiosity with substructuring, elements are subdivided until their radiosity gradient drops below a preset *epsilon*, and the algorithm is terminated when the residual error drops below a preset fraction of the original residual error. In wavelet radiosity, refinement is controlled by the oracles described in the original papers, which also depend on a single epsilon parameter. In regions of partial visibility this epsilon is reduced in order to encourage subdivision around shadows [4]. The algorithm is terminated when no more links need refinement and the residual error has dropped below a fraction of the original.

Visibility. Ray-casting is used for all inter-patch visibility tests, in order to standardize visibility testing between the algorithms. For the progressive methods, visibility is tested by casting a single ray from a patch to each other element in the scene [16]. For the wavelet methods, a 4×4 jittered sampling scheme is used [14], along with triage [9]: refined links have their visibility recalculated only in areas of partial occlusion.

Meshing. For progressive radiosity, each polygon in the scene is subdivided into a mesh such that the edge-lengths of all mesh elements are smaller than a specified maximum.

For methods using piecewise-constant basis functions, it is common to perform post-process smoothing to eliminate step discontinuities. We use this approach for the progressive and Haar wavelet methods. For multiwavelets, we display directly from the basis functions, as did Gortler et al. [8]. The resulting functions will thus be piecewise linear or piecewise quadratic, with possible discontinuities between elements; we attempt to minimise these by ensuring the mesh is balanced. Flatlets are first converted to the same order multiwavelet basis, and then displayed in a similar manner, and thus may also exhibit discontinuities.

Form-Factors. While a closed-form solution for the double integral that defines the area-to-area form factor exists [12], it is too expensive to use in most situations. We follow the standard practice of approximating the form factor by its point-to-point counterpart, which works well when two patches are sufficiently far apart. There are potential difficulties; when two patches are not sufficiently far apart for the approximation to hold, and in particular as r gets small, the term goes to infinity. In either of these cases, we switch to the polygon-to-point form factor, which is more accurate and contains no singularity, but is more expensive [4, 18].

Multiwavelets require quadrature rules to calculate the coupling coefficients between basis functions. The presence of a singularity in the form-factor kernel where surfaces touch can cause problems, because the function diverges from the lower-order-polynomial assumption of these rules [20]. In such situations we swap to the C-C transfer functions [11].

2.2 Test Scenes

We chose test scenes that explore among others the following properties**:**

- **Interreflection.** High reflectance makes some algorithms quite slow, and exposes flaws in others.
- **Occlusion.** Hard shadows are a difficult test for most radiosity algorithms.
- **Complexity.** Some algorithms have a cost linear in scene complexity, while others are quadratic.

To test these properties, we constructed a number of experiments, each of which consisted of a scene, and some parameter of that scene which could be varied. The radiosity algorithms were then run against each of the scene variants in turn. Representative images of the various scenes used can be seen in Figure 1, and the experiments are summarised below.

Experiment	Parameter Varied	Property Tested
Blocker	Blocker height	Occlusion
Tube	Reflectance	Multi-bounce interreflection
Complex	Number of chairs	Geometrical complexity

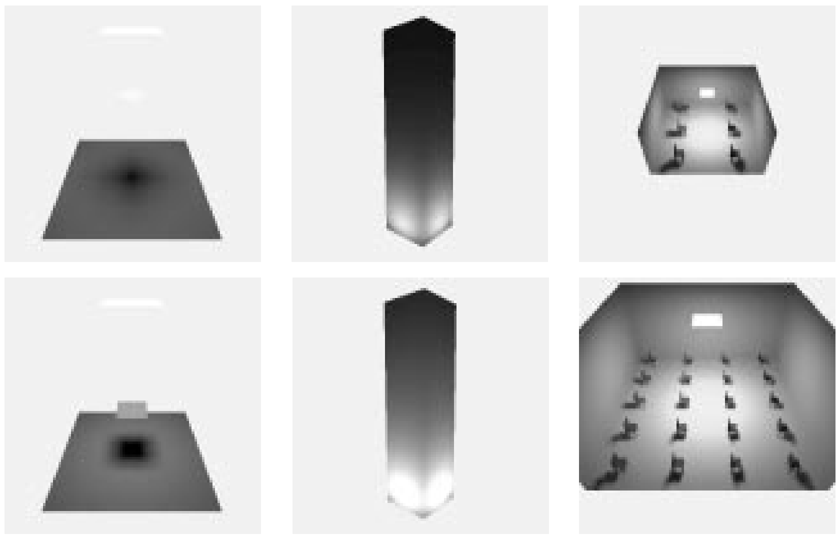

Fig. 1. From left to right: blocker experiment, separations of 1.4 (top) and 0.7 (bottom), tube experiment, reflectance of 0.3 (top) and 0.9 (bottom), complex experiment, 6 chairs (top) and 20 chairs (bottom).

The first experiment tests occlusion from sharp-edged shadows (obscuring polygon close to the receiving polygon) to very soft shadows (where the occluder is close enough to the source for the umbra to vanish on the receiver). The tube experiment consists of a long tube with a light at one end. It tests interreflection between long chains of patches. The complex experiment tests geometrical complexity by varying *m*, the number of chairs. There are 5 + 27*m* polygons in this scene, and the room's surface area scales in proportion to *m*. We scale the geometry as the complexity grows because we feel that this is typically the way detail is added to a scene. Elsewhere we examined an unscaled scene [18].

2.3 Testing

The testing process involved applying a subset of the radiosity methods available to us to each scene variant in each experiment. A batch renderer was used to produce radiosity simulations and statistics for each of the experiments' scenes. The renderer takes a scene description, as well as a number of parameters describing which method and settings to use. Rather than an image, the renderer outputs a scene file containing a description of the mesh used, and the radiosity distribution over that mesh. Once computed and stored, this output mesh can later be retrieved and viewed in a scene viewer, compared with another mesh file, or rendered to an image file.

Checkpointing. For each radiosity simulation, the algorithm is checkpointed a number of times over the life of its run, in order to see how the algorithms behave over time. At each of the checkpoint times, we write out the current state of the solution along with other pertinent statistics. Progressive and wavelet radiosity are inherently iterative, so for these algorithms we get a picture of how quickly they converge.

Reference Solutions. To measure the error in the current solution at each of the checkpoints (a post-process), we compare it to a reference solution which is either “exact”, or much more accurate than any method we are testing. Validation against analytic solutions was done where possible (on scenes with no interreflection), including the blocker experiment. To generate a reference solution for the tube experiment, we used matrix radiosity, with analytical patch-to-point form factors, and a mesh with at least four times as many elements as any of the solution meshes that were generated. Matrix radiosity was used because it produces close to an exact solution for a given mesh. We regard this as an improvement over previous comparisons, which have typically used a "converged" solution on the same mesh as a reference.

While we could afford to run the reference solutions for much longer than we ran the experimental solutions, for the complex experiment, the matrix radiosity method proved too memory intensive for the available hardware. Instead, we generated our reference solutions using the progressive radiosity algorithm; the algorithm was run long enough to generate a solution of accuracy equal to the matrix algorithm, and again, a denser mesh was used.

Error Measure. The radiosity algorithms we consider here are all view-independent global illumination techniques, so we use a view-independent error measure, as opposed to an image-based one. To this end we compute the relative error between the experimental solution and the reference solution. Both share the same set of base polygons, so we first define the absolute error to be the area-weighted RMS radiosity difference over all pairs of base polygons. To compute this we used Monte Carlo sampling over the base polygons. As the total emitted radiosity in the scene scales any error, we then define the relative error as the absolute error divided by the average reflected radiosity over the reference solution [18]. Parameters were typically set to drive the relative error below 5% or so. We do not use the residual as our error measure, as some previous studies have done, because it does not support comparison of solutions from different meshes well.

Equalised Plots. For each experiment we run, we can produce a plot of the solution error over time for every combination of scene and method (Figure 2, left). Each curve is a run, and each data-point is a snapshot of that run at some point in time. The right-most point shows the error at convergence (when the algorithm terminates). To fairly compare methods with varying time and accuracy, we produce a time-equalised error plot. For a plot of a given set of methods, we find the minimum convergence time of any method, and then find the error of each method at that time by interpolation. This is illustrated generically in Figure 2: in the graph on the left for parameter value 12, the substructuring method terminates first at 144 seconds, so all the other methods are 'rolled back' to this point, and the resulting interpolated error values used to produce the highlighted data points at parameter value 12 in the graph on the right. In order to clarify equalised plots, we label each set of data points on them with their corresponding time at the top.

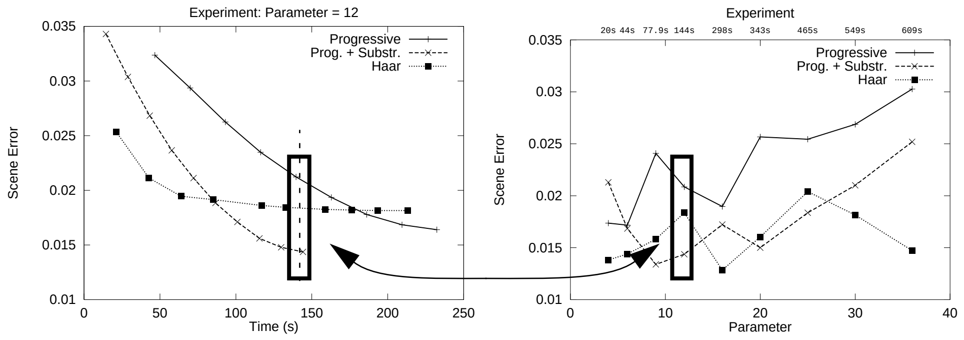

Fig. 2. Points interpolated from left graph are used to create equalised plot at right.

3 Results

Our final results comprised a large database of statistics and stored radiosity representations. In this section we present the most interesting of the graphs and images produced, and discuss their implications.

3.1 Time and Error

The time-equalised plot of Figure 3, left summarises the performance of progressive and wavelet radiosity in the complex experiment.The substructuring and Haar wavelet methods have much the same error until the last two scenes, where the Haar method gains the edge. Noticeably, the trend for the wavelet methods is towards lower errors at this point, whereas the trend for the progressive methods is towards higher errors. (We stop at 36 chairs, or 1000 polygons, because here the higher-order methods run out of memory.) The M2 basis does not perform as well as Haar in this plot because of poor visibility handling, as we will see later, and the M3 basis is hampered by its extra time and space overhead.

The reason that hierarchical radiosity starts to do better than substructuring radiosity is apparent from the time-to-convergence graph shown in Figure 3, right. Whereas the convergence times of the progressive methods increase quadratically, the convergence time of wavelet radiosity with the Haar basis is linear with scene complexity, as is the M2 basis. This qualitative difference in speeds of the two methods is one of the most interesting results from the complex experiment.

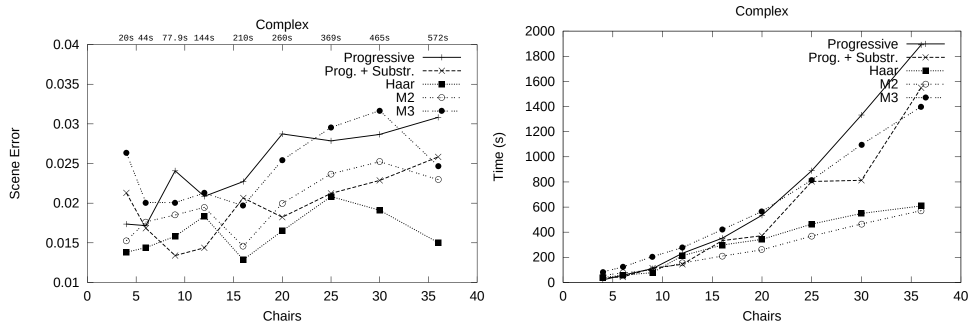

Fig. 3. Complex experiment, progressive and hierarchical radiosity, left: time-equalised error, right: time to convergence.

For this experiment, as complexity increases, the progressive methods spend more and more time shooting light from the walls of the scene, which are increasing in area, whereas the wavelet methods use roughly a constant number of links between a wall and other parts of the scene. We have observed that for an unscaled complex scene, where room size does not increase with the number of chairs, progressive radiosity is not penalised so heavily in this way, and outperforms hierarchical radiosity.

3.2 Wavelet Memory Use

We found that higher-order wavelet radiosity techniques consume excessive memory. To some extent this is true of Haar as well. As Figure 4 shows, the memory consumption of the wavelet algorithms is large, especially compared to the 1Mb at most used by the progressive radiosity algorithm. In this particular run, the M3 basis reached 550MB of storage. An obvious argument here is that for some of these runs we have chosen too high an accuracy, and created too many links. We tried to choose epsilon to achieve a reasonable time/error trade-off. The higher order methods produced a scene with higher error than the Haar basis, so they were not trying to do too much work compared to Haar.

Clustering has been shown to help this problem; in particular Gibson et al. [6] have impressive results for high-complexity scenes, and have managed to run wavelet radiosity with the Haar basis against a 225,000 polygon scene in 180Mb of memory. Unfortunately, clustering techniques have not yet been applied to higher-order methods, only the Haar basis. Even assuming linear storage complexity is achieved with such clustering methods, our results lead us to predict that memory consumption would be over 2Gb if the M3 basis were used on a similar scene.

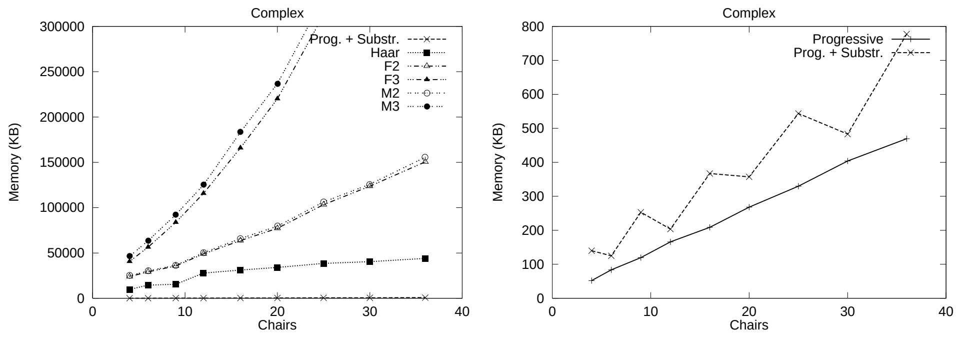

Fig. 4. Complex experiment, left: memory use, right: progressive memory use.

3.3 Empirical Complexity of Wavelet Radiosity

We observed strong effects of complexity on time and memory use in our experiments. The most interesting of these relations were:

- **#links = σ*k*^{2} + β*k***. The number of initial links in the wavelet radiosity methods is σ*k*^{2}, where *k* is the number of polygons, and σ is the visual density of the scene (the average fraction of scene polygons visible from a particular scene polygon). The number of extra links created during refinement is Θ(*k*); evidence of this can be seen in Figure 5 for the higher order wavelet methods. Time and memory use are both linear in the number of links as epsilon is varied. Interestingly, for the scenes tested, the extra link term dominated for Haar (hence its linear time cost in Figure 3) and the initial links dominated for the higher order methods, yielding near-quadratic time cost.These results extend the existing theoretical analysis [9].
- **#links = Θ(1/epsilon)**. For small enough epsilon, the number of links generated by the wavelet methods for a fixed scene is proportional to 1/epsilon, as we see in Figure 5. As epsilon is increased, and the total number of links approaches the minimum, σ*k*^{2}, the relationship becomes sublinear, approaching *O*(1) as epsilon approaches 1. This relationship is useful when setting epsilon for a wavelet radiosity solution: as a rule of thumb, halving epsilon will double the time and memory cost of the algorithm.

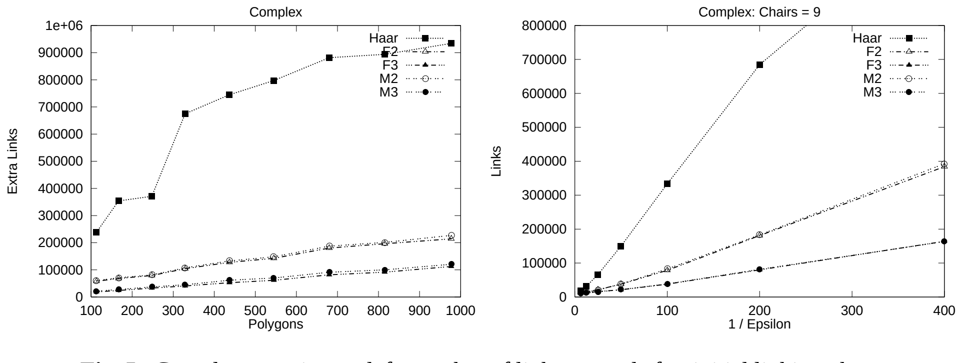

Fig. 5. Complex experiment, left: number of links created after initial linking phase vs. number of polygons (*k*), right: effect of varying epsilon on total links created.

3.4 Progressive and Hierarchical Shadow Handling

The blocker scene was used as a simple test of shadowing. From the error plot in Figure 6 we see that whereas substructuring achieves the lowest error when the blocker is closer to the receiver, and hierarchical radiosity the worst, this order is reversed as the blocker approaches the source. Significantly, the cross-over point occurs almost exactly when the umbra disappears, at a separation of $1\frac{1}{3}$. Although hierarchical radiosity achieves the best results when the shadows are softer, it takes much longer to do so, as can be seen from the time graph in Figure 6, because it is trying to solve a harder problem. Whereas substructuring will only subdivide the receiver, hierarchical radiosity can also subdivide the source, and in this case it must subdivide heavily; the source is only a little larger than the blocker, so source elements must be quite small for them to be close to completely visible from the elements of the receiver.

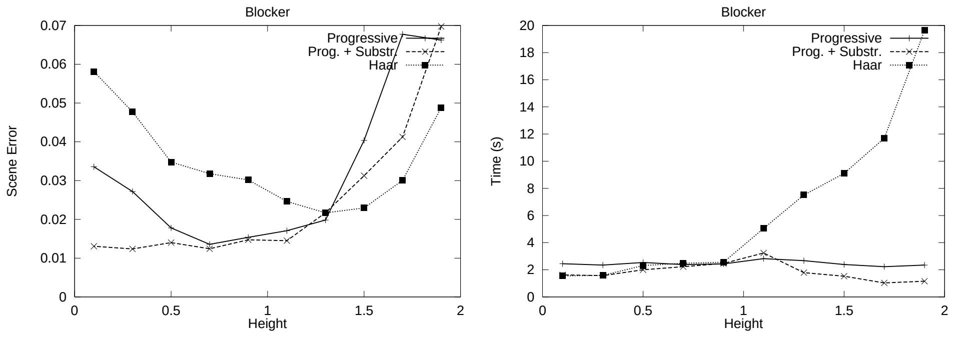

Fig. 6. Blocker experiment for progressive and hierarchical radiosity, left: error at convergence, right: time to convergence.

3.5 Wavelet Shadow Handling

We tried two quadrature methods for the higher order wavelets, *fractional visibility*, which scales the results of quadrature with the estimated visibility between two patches [9,8], and *visibility-in-quadrature*, which leaves visibility in the integrand [5]. Neither of these methods performed particularly well in the complex experiment. The bottom-left image in Figure 7 shows a solution using the M2 basis and the fractional-visibility method, where visibility is factored out of the quadrature: it is overly blocky. The bottom-right image shows the same basis with the visibility-in-quadrature approach, which looks smoother, but still has large discontinuities between elements. We found that the objective error of both these solutions (and those for other complex scenes) were much the same. Both methods are clearly subjectively inferior to the reference solution (upper left) and the Haar basis (upper-right).

The cause of the problems with the fractional-visibility approach is that it assumes visibility is constant across the element, even though with the higher-order methods the radiosity is no longer constant. The visibility-in-quadrature approach we used handles the variation of visibility across individual elements better, but because it samples visibility on a 2×2 grid, rather than a 4×4 grid, it suffers more from aliasing problems. This is exacerbated by triage, as aliasing can occur at higher levels in the hierarchy; to produce the image in Figure 7, we had to disable triage to prevent large ‘dropouts’ in the shadow.

A solution to these problems is to subdivide even more heavily in areas of partial visibility. However, this wastes transfer and basis coefficients, especially for higher order wavelets. Slusallek et al. [13] have addressed this problem by applying the shadow masks proposed by Zatz [20]; this is a generalisation of fractional visibility in which visibility is evaluated at a finer resolution, smoothed by interpolation, and multiplied into the radiosity function. But visibility and irradiance are not independent, so a more correct approach would be to use visibility-in-quadrature with dis-

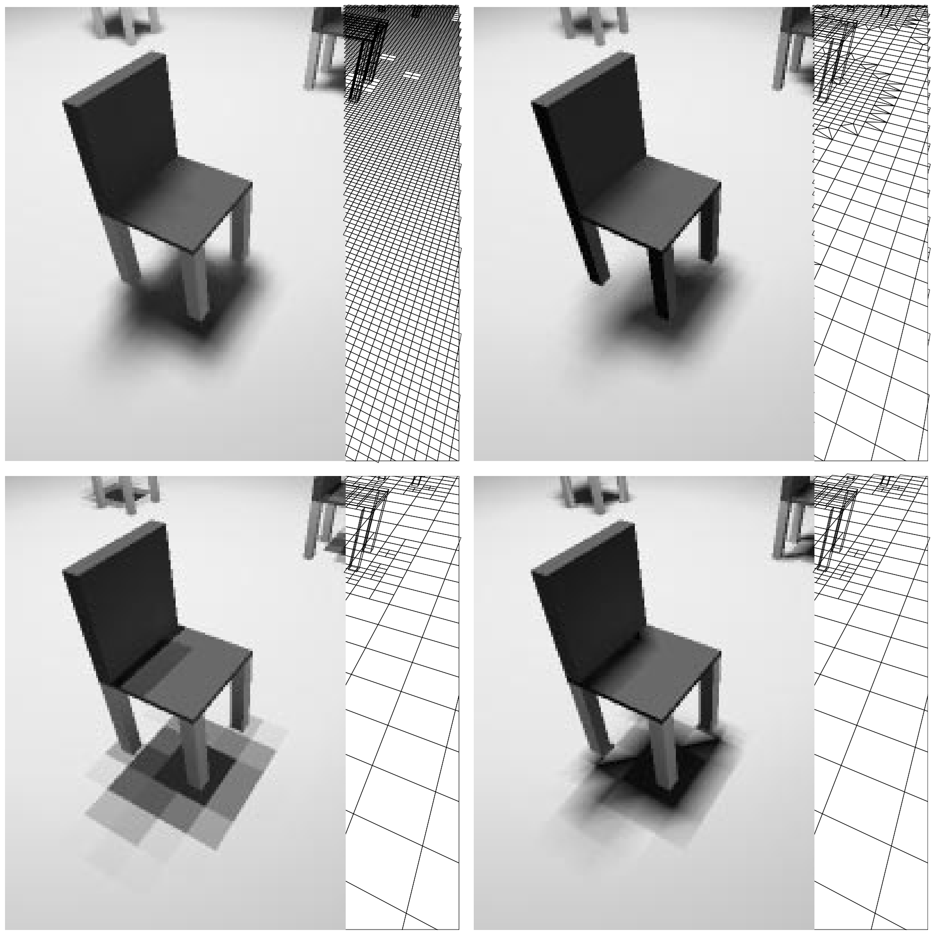

Fig. 7. Upper-left: reference solution, upper-right: the Haar basis, lower-left: the M2 basis (fractional visibility), lower-right: the M2 basis (visibility incorporated into the quadrature). The split views show the solution on the left, and the mesh on the right.

continuity meshing and continuous wavelet basis functions, to avoid the problems seen above. However, this may well have too large an overhead to be practical.

3.6 Reflectance

In the tube experiment, high reflectance coupled with the large number of abutting patches in the scene acted to retard the adaptive methods so that they performed worse than vanilla progressive radiosity. Substructuring is slow, and Haar has large error, for high reflectance values (Figure 8). The tube experiment highlights how touching or close-proximity surfaces are a problem for wavelet methods; many links are created in these cases, slowing them down markedly.

The substructuring method’s problems were caused by a phenomenon we refer to as *overrefinement*, which occurs with high reflectance scenes and lights that touch another surface at a reflex corner. During early shooting steps, the illumination gradient is high near the bottom of the tube, causing that area to be heavily refined, even though later shooting steps will “fill in” the tube’s illumination, smoothing the gradient and eliminating the need for such a fine mesh. This effect is demonstrated in Figure 9; in the first picture, only the right hand part of the light source has been shot, and the algorithm has meshed finely around this initial bright spot. In the second picture we see that after the algorithm has run further, the entire area in front of the light source has been smoothly illuminated, and in fact this area could have been meshed relatively coarsely.

The problem could be solved using a multigrid approach. If the scene is first illuminated with a coarse mesh and with substructuring turned off, then, when the substructuring method is run using that mesh as a starting point, patches that are shot early on will not overrefine areas where the final radiosity gradient is smooth.

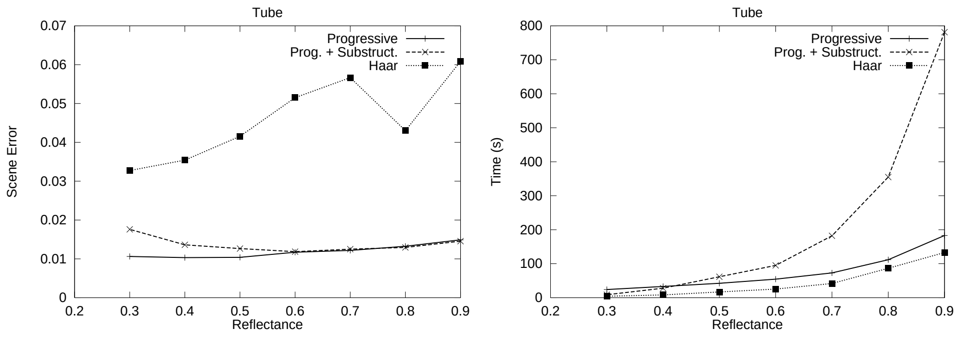

Fig. 8. Tube experiment, left: time to convergence, right: error at convergence.

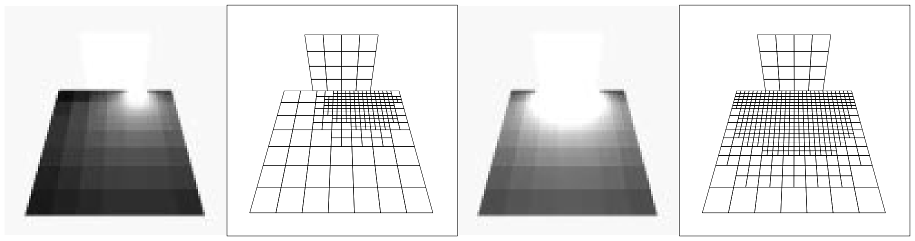

Fig. 9. Over-refinement in Progressive Radiosity. On the left, the first shots from emitter patches cause heavy subdivison, because most of the receiver is still unilluminated. On the right, shots from the rest of the emitter have illuminated the rest of the receiver, making some of the earlier refinement redundant.

We have observed significant *accumulated projection error* when wavelet radiosity with constant basis functions (Haar, F2 or F3) is applied to medium to high-reflectance scenes. This leads to bright strips of light where two patches meet along an edge. The radiosity function should go to zero along such an edge if the source element itself does not touch it, but when the function is projected into a constant basis, this no longer occurs. These edge errors accumulate as the link hierarchy deepens, so that the more heavily the mesh is subdivided, the worse the problem gets. This phenomenon has also been observed by Slusallek et al. [13], who propose as a solution splitting the form-factor into constant and cos(θ) dependent parts, and re-evaluating the latter at lower levels of the hierarchy. An alternative is to ensure that links to elements adjacent to the source patch are always within a few levels of the leaf nodes, thus limiting the depth of the hierarchy. It is not clear to us which is the most efficient approach.

4 Conclusions

Our main conclusions are:

- **For moderately complex scenes, wavelet radiosity using the Haar basis is often best**. For our ‘complex’ experiment, Haar did the best or near best of the methods we tested (Figure 3). Also, for the complex scene, its time cost appeared to grow linearly, while the cost of progressive radiosity with substructuring appeared to be quadratic. For a given amount of time, Haar often gave the most accurate results. However ...
- **Wavelet methods use an immense amount of memory**, so for very complex scenes they become impractical. Empirically, the memory requirements for wavelet methods grow quadratically with scene complexity (Figure 4). Also, the higher-order methods require many times more memory than Haar. Recall that wavelet methods of order M use $O(M^2)$ memory per element and $O(M^4)$ memory per link. Furthermore, in contrast to time, memory is a finite resource. **When the scene is too complex for wavelet methods, substructuring becomes a better choice**.

	Substruct.	**Haar**	**F2 & M2**	**F3 & M3**	**Matrix**
Memory for ‘complex’, 36 chairs	0.8 MB	44 MB	150 MB	550 MB	1000 MB (estimate)

- **Higher-order wavelet methods as implemented look poor on scenes with occlusion.** With either the fractional visibility method or the visibility-in-quadrature method, shadows appear extremely blocky (Figure 7) unless heavy subdivision occurs. Higher-order wavelets currently handle visibility poorly, preventing them from achieving their potential accuracy.
- **For hard shadows, substructuring is recommended.** Objectively, on the blocker scene we observed that substructuring is the most accurate method when a shadow contains an umbra (Figure 6). Because wavelet methods typically refine less than substructuring near shadow boundaries, their shadows appear blurrier or blockier.

It should be noted that our conclusions are based on our experimental scenes, and the basic radiosity methods as we have implemented them. While we have attempted to devise the tests, to choose algorithms, and to implement them in an unbiased way, one should use caution in generalizing our conclusions. We would expect the use of clustering and other advanced techniques to alter these conclusions.

Future Work

Now that we have our testbed in place, and have completed our initial investigation of radiosity methods, there are a number of avenues for future work. These include:

- Implementing and investigating clustering for all wavelet methods [14].
- Testing more sophisticated meshing schemes.
- Evaluating the effect a final-gather pass has on the accuracy of the simulation.
- Investigating performance on a range of complex real-world scenes, such as architectural models.
- Comparing the radiosity approaches investigated here to Ward’s RADIANCE system [17].
- Producing results for the standard library of scene files due to Rushmeier et. al. [10].
- Considering perceptually-based error measures.

To assist others in the research community with the investigation of these areas, and for evaluation of our experiments, we plan to place our radiosity renderer and experimental system on the internet at the following URL: `http://www.cs.cmu.edu/~radiosity/`.

References

[1] Gladimir V. Baranoski, Randall Bramley, and Peter Shirley. Fast Radiosity Solutions for High Average Reflectance Environments. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 345–356. Springer-Verlag/Wien, 1995. (Technical report: http://swarm.cs.wustl.edu/~hart/mirror/indiana-biblo.html.)

[2] Michael Cohen, Donald P. Greenberg, Dave S. Immel, and Philip J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(3):26–35, March 1986.

[3] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):75–84, Aug. 1988.

[4] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press, Boston, 1993.

[5] Reid Gershbein, Peter Schröder, and Pat Hanrahan. Textures and radiosity: Controlling emission and reflection with texture maps. In *Computer Graphics (Proceedings of SIGGRAPH '94)*, pages 51–58, July 1994. http://www-graphics.stanford.edu/papers/texture.

[6] Simon Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, December 1996.

[7] Steven J. Gortler, Michael F. Cohen, and Phillipp Slusallek. Radiosity and Relaxation Methods. *IEEE Computer Graphics and Applications*, 14(6):48–58, November 1994. http://www.cs.princeton.edu/gfx/papers/relax.

[8] Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, August 1993. http://www-graphics.stanford.edu/papers/wavrad.

[9] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991. http://www-graphics.stanford.edu/papers/rad.

[10] Holly Rushmeier, Greg Ward, Christine Piatko, Phil Sanders, and Bert Rust. Comparing real and synthetic images: Some ideas about metrics. In *Rendering Techniques '95*, pages 82–91. Springer-Verlag/Wien, 1995. http://radsite.lbl.gov/mgf/compare.html.

[11] Peter Schröder. Numerical integration for radiosity in the presence of singularities. In *Fourth Eurographics Workshop on Rendering*, Paris, June 1993. http://www.cs.princeton.edu/gfx/papers/integration/.

[12] Peter Schröder and Pat Hanrahan. On the form factor between two polygons. In *Computer Graphics (Proceedings of SIGGRAPH '93)*, pages 163–164, 1993. http://csvax.cs.caltech.edu/~ps/formfactor/ffpaper.ps.gz.

[13] Philipp Slusallek, Michael Schroder, Marc Stamminger, and Hans-Peter Seidel. Smart Links and Efficient Reconstruction for Wavelet Radiosity. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 240–251. Springer-Verlag/Wien, 1995. http://www9.informatik.uni-erlangen.de/eng/research/pub95/.

[14] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. http://csvax.cs.caltech.edu/~arvo/papers.html.

[15] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, San Francisco, CA, 1996. http://www.amath.washington.edu/~stoll/pub.html.

[16] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 315–324, July 1989.

[17] Gregory J. Ward. The RADIANCE lighting simulation and rendering system. In *SIGGRAPH 94 Proc.*, pages 459–472, July 1994. http://radsite.lbl.gov/radiance/papers/sg94.1/paper.html.

[18] Andrew J. Willmott and Paul S. Heckbert. An empirical comparison of radiosity algorithms. Technical Report CMU-CS-97-115, School of Computer Science, Carnegie Mellon University, April 1997. http://www.cs.cmu.edu/~radiosity/emprad-tr.html.

[19] Wei Xu and Donald S. Fussell. Constructing Solvers for Radiosity Equation Systems. *Fifth Eurographics Workshop on Rendering*, pages 207–217, June 1994.

[20] Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. *Computer Graphics (SIGGRAPH '93 Proceedings)*, August 1993. http://www.rhythm.com/~hzatz/.