Light-Driven Global Illumination with a Wavelet Representation of Light Transport

Robert R. LEWIS bobl@cs.ubc.ca
Alain FOURNIER fournier@cs.ubc.ca

Imager Computer Graphics Laboratory
Department of Computer Science
University of British Columbia

29 May, 1996

Abstract

We describe the basis of the work he have currently under way to implement a new rendering algorithm called *light-driven global illumination*. This algorithm is a departure from conventional raytracing and radiosity renderers which addresses a number of deficiencies intrinsic to those approaches.

1 Introduction

In computer graphics, we use illumination – the study of how light interacts with matter to produce visible scenes – to produce realistic images. Illumination encompasses both local and global phenomena. Local illumination describes the interaction of light with a single, small volume or surface element with given incident and viewing directions.

We take the fundamental equation describing local illumination to be

$$L = L_e + \int_{\Omega^R_{\mathbf{N}}} f_r(\mathbf{S}', \mathbf{V})\, L_i\, |\mathbf{N} \cdot \mathbf{S}'|\, d\omega'_i + \int_{\Omega^T_{\mathbf{N}}} f_t(\mathbf{S}', \mathbf{V})\, L_i\, |\mathbf{N} \cdot \mathbf{S}'|\, d\omega'_i \qquad (1)$$

where $\mathbf{N}$ is the surface normal, L is the total radiance given off (either L_r, reflected, or L_t, transmitted) in direction $\mathbf{V}$, L_e is the surface emissivity, L_i is the radiance from direction $\mathbf{S}$, $\Omega^R_{\mathbf{N}}$ is the reflection hemisphere (contains $\mathbf{V}$), $\Omega^T_{\mathbf{N}}$ is the transmission hemisphere (opposite $\Omega^R_{\mathbf{N}}$), f_r is the bidirectional reflectance distribution function (BRDF), f_t is the bidirectional transmittance distribution function (BTDF), $d\omega_i$ is $\sin\theta_i d\theta_i d\phi_i$, θ_i is the incident polar angle, and ϕ_i is the incident azimuthal angle. We use the $'$ to indicate bound variables of integration. A local illumination solution is entirely characterized by its BRDF and BTDF functions.

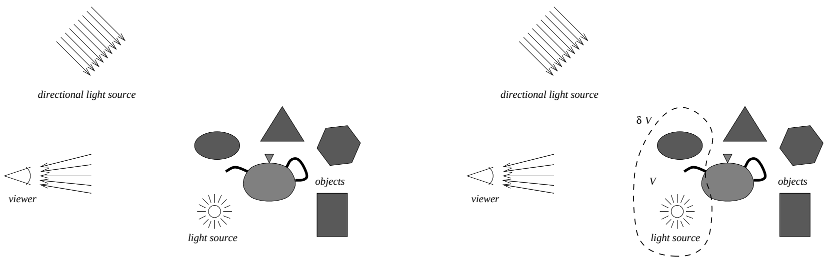

Figure 1: A Typical Global Illumination Problem (left) and an Application of Isolation (right)

Global illumination describes how light is distributed in a scene. Global illumination solutions are built on top of local illumination solutions. Figure 1 (left) illustrates a typical global illumination problem, in this case a viewer-dependent one.

Before we move on, it is interesting to consider that both ray tracing and radiosity rendering techniques simplify the integral in (1) in quite different ways. Classical ray tracing simplifies it either by setting f_r and f_t to be Dirac δ-functions, which permits (mirrorlike) reflection and refraction, or by setting L_i to include a δ-function, which permits point and directional light sources. Classical radiosity takes the radiance L to be isotropic and constant over a patch and therefore equal to $\frac{B}{\pi}$, where B is the radiosity of the patch.

We will summarize here the work we've done so far in developing a "light-driven" global illumination algorithm we refer to as *Lucifer*[1]. This is a new global illumination solution that allows the incorporation of a larger range of local phenomena than other global solutions. The work presented here is a continuation of work done by Fournier, Fiume, Ouellette, and Chee [3] (hereafter referred to as "FFOC").

1.1 Illumination Effects

A renderer, which is a system to create synthetic images, is based on a rendering technique. Table 1 lists a number of illumination effects and whether or not various established rendering techniques are capable of them. One word of caution: If we evaluate this table for specific renderers, it becomes clear that it is an oversimplification. Some renderers which refer to themselves as raytracers, for instance, are capable of soft shadows. This is because those renderers have incorporated radiosity or other methods in hybrid form and are not "pure" raytracers in that sense. For this reason, Table 1 should not be taken to refer to specific renderers, only to rendering techniques.

Indeed, ray tracing and radiosity techniques can be enhanced (often by Monte Carlo methods) to produce a wider range of effects than Table 1 indicates. As in many

[1] From the Latin for "light bearer".

Effect	Decreasing Efficiency → Scan Conversion w/Z-Buffer	Ray Tracing	Radiosity
diffuse surfaces	yes	yes	yes
specular highlights	yes	yes	no
transparency	yes	yes	no
mirror reflection	no	yes	no
refraction	no	yes	no
sharp shadows	yes	yes	no
soft shadows	no	no	yes
diffuse lighting	yes	no	yes
color bleeding	no	no	yes
caustics	no	no	no

Table 1: Effects Available with Established Rendering Techniques

other areas in which they are applied, however, Monte Carlo techniques exhibit slow convergence.

To be able to produce these and other effects, we will elaborate on a global illumination technique that allows a wider range of local illumination models.

2 Light-Driven Global Illumination

The algorithm we present is a sequential computational analogue of how energy distributes itself in a scene. It combines the physical concepts of isolation and energy conservation with a refineable spatial partitioning scheme.

2.1 Isolation

For possibly complex scenes, we can take a divide-and-conquer approach based on the principle of *isolation*, illustrated in Figure 1 (right). Isolation is a conceptual tool: we place any part of our scene (including the viewer) within a volume V. Suppose then that somehow we can determine the distribution of radiance on the surface ∂V of V. Isolation says that for the purposes of solving for global illumination anywhere outside V, we can effectively replace the contents of V with ∂V's radiance distribution.

Isolation gives us a way to deal with complex scenes. If we cannot deal with the whole scene, break it into volumes we *can* deal with as local illumination problems and transfer radiance distributions along the boundaries between the volumes.

2.2 Power Computation for Isolated Volumes

For any volume $\mathbf{V}_i$, physics demands energy conservation. In the steady state, this is equivalent to power conservation:

$$\Phi_{\text{in},i} + \Phi_{\text{em},i} = \Phi_{\text{out},i} + \Phi_{\text{abs},i} \tag{2}$$

where

$$\Phi_{\text{in},i} = \int_{\partial \mathbf{V}_i} \int_{\Omega^+_{\mathbf{N}}} L_{\text{in}} \, |\mathbf{N}' \cdot \mathbf{S}'| \, d\omega' dA' \tag{3}$$

is the flux entering $\mathbf{V}_i$,

$$\Phi_{\text{out},i} = \int_{\partial \mathbf{V}_i} \int_{\Omega^-_{\mathbf{N}}} L_{\text{out}} \, |\mathbf{N}' \cdot \mathbf{S}'| \, d\omega' dA' \tag{4}$$

is the flux leaving $\mathbf{V}_i$,

$$\Phi_{\text{em},i} = \int_{\mathbf{V}_i} \int_{\Omega} j \, d\omega' d\mathbf{V}'_i + \int_{S_{\text{em}}(\mathbf{V}_i)} \int_{\Omega^+_{\mathbf{N}}} L_e \, |\mathbf{N}' \cdot \mathbf{S}'| \, d\omega' dA' \tag{5}$$

is the flux generated within $\mathbf{V}_i$, $\Omega^+_{\mathbf{N}}$ and $\Omega^-_{\mathbf{N}}$ at any point on ∂V are the unit hemispheres entirely outside and entirely inside $\mathbf{V}$, respectively, L_{in} is the radiance coming into $\mathbf{V}_i$ from $\Omega^+_{\mathbf{N}}$, L_{out} is the radiance passing out of $\mathbf{V}_i$ from $\Omega^-_{\mathbf{N}}$, Ω is the unit directional sphere, j is the volume emissivity within $\mathbf{V}$, L_e is the surface emissivity on all emissive surfaces $S_{\text{em}}(\mathbf{V}_i)$ within $\mathbf{V}$, and $\Phi_{\text{abs},i}$ is the amount of radiant power absorbed and not converted into radiant energy again (at least in $\mathbf{V}_i$).

2.3 Solution Order and Convergence

As with raytracing and radiosity, we are constructing a sequential analogue of what nature does in parallel. We need a sequential ordering. A reasonable way to proceed is to maintain the volumes in a queue sorted in order of decreasing *undistributed power* $\Phi_{\text{in},i} + \Phi_{\text{em},i}$ and to always concentrate our efforts on the volume $\mathbf{V}_i$ at the front of the queue. Once the undistributed power in this volume is distributed, the volume moves to the end of the queue. Note that, from (4) and (5), undistributed power is computed from radiances, emissivities, and geometrical information about $\mathbf{V}_i$.

If, during the distribution process, we ensure that no more energy leaves $\mathbf{V}_i$ than was either incident upon or emitted within it, we can guarantee that the total undistributed power in the scene (i.e., in the queue) is monotonically nonincreasing, since all the power distributed by $\mathbf{V}_i$ that does not get absorbed becomes incident on another volume or leaves the scene entirely, possibly reaching the observer.

One way to ensure that no excess energy is created is to require an energy-conserving local illumination model. We have studied some necessary constraints for such a model in [8] and have considered several illumination models commonly in use in computer graphics from the aspects of both energy conservation and Helmholtz reciprocity[2].

[2]Note that *Lucifer* as presented here does not require reciprocity.

```
render(scene s)
{
    queue q;
    cell c;

    q = createPriorityQueue(rootCell(s));
    while (sumUndistPower(q) > powerTolerance) {
        c = dequeue(q);
        if (isEmptyCell(c))
            propagate(c);
        else if (canDealWith(c))
            balance(c);
        else if (size(c) > minimumSize)
            subdivide(c, q);
        else
            trivialize(c);
    }
}
```

Figure 2: The *Lucifer* Algorithm

2.4 Spatial Partitioning

We require a tesselation of the scene that can be easily refined as needed as the solution progresses. Several data structures permit this, but the simplest one for our purposes is the octree.

We take octrees to be composed of cubic *cells*. The *root cell* contains the entire scene. Each cell is either a *parent cell* or a *leaf cell*. A parent cell has eight *child cells*. A leaf cell has no child cells. Only leaf cells are "volumes" in the sense of our previous discussion. Non-leaf cells are purely structural. Each cell except the root cell has seven *sibling cells*.

We also choose a minimum size for an octree cell. Cells that we cannot deal with that are this size will be solved trivially, but still conserving energy.

2.5 The *Lucifer* Algorithm

Figure 2 shows our rendering algorithm pseudocode. We start off with a scene `s` with a single cubic cell, `rootCell(s)`. We then create a priority queue `q` with, initially, a single element, `rootCell(s)`. `q` is always maintained in decreasing order of undistributed power.

If the queue is empty or, `sumUndistPower(q)`, the sum of the undistributed powers of all cells in the queue, is lower than some prespecified `powerTolerance`, we consider the scene rendered.

Otherwise, we remove the cell with the largest amount of undistributed power, `c`, from the front of `q`. All further computation in the main loop is concerned only with `c`. This is why we refer to our algorithm as "light-driven": it is always concerned with the cell with the largest undistributed power.

We test `c` against the first of these cases that it matches:

- If it is empty, we call `propagate()` to transfer the radiance coming into `c` to its neighbors.
- If it is a cell that we know how to pass power through, we call `balance()` to perform the redistribution of the reflected and refracted radiance to `c`'s neighbors.
- If it is above a specified minimum size, we call `subdivide()` to split `c` into eight child cells and add these back to `q` in order.

If it meets none of these criteria, we call `trivialize()` to perform an ad hoc solution that may be some combination of the first two cases, but which in any case guarantees that power balance is preserved.

`propagate()` is conceptually straightforward. It makes use of the invariance of radiance in the direction of propagation. Radiance at a given point in a given direction on an incoming cell wall defines a ray which may intersect an outgoing cell wall at a new point and direction. (The direction is, of course, the same, but may be expressed in a different coordinate frame.) We may alternatively project backwards from the outgoing cell wall to the incoming one.

`balance()` is somewhat more complicated. The ambiguity of the phrase "know how to pass power through" at this level of description is intentional. Our current interpretation of it is that the cell contains at most a single non-concave object. This allows us to decompose the procedure into four subtasks:

1. Transport radiance from the incoming walls to the object (or that part of the object which lies within the cell).
2. Interact the incoming radiance with the object's BRDF (and BTDF) to produce a reflected radiance being given off by the object.
3. Transport reflected and emitted radiance from the surface of the object to the outgoing walls.
4. Transport the unobstructed part of the incoming radiance to the outgoing walls.

Subtasks 1 and 3 are variations on `propagate()`, allowing transport to and from surfaces as well as cell walls. Subtask 2 requires surface interaction, as dictated by (1). Subtask 4 is another variation on `propagate()`, including a geometric blocking function that inhibits propagation if the ray intersects the object within the cell.

2.6 Complexity Issues

Let us compare the complexity of *Lucifer* and two rendering techniques that can produce comparable effects: raytracing and radiosity. We assume the scene has N_{obj} objects.

In classical raytracing, finding each intersection point requires $O(N_{\text{obj}})$ time per ray, and an additional $O(N_{\text{obj}})$ amount of time is required to characterize the illumination in the vicinity of the point. So the cost is $O(N_{\text{ray}} D N_{\text{obj}})$ where N_{ray} is the number of primary rays and D is the average depth of the raytrace per ray. As summarized by Arvo and Kirk in [1], many techniques have been devised to improve on this. With suitable restrictions on such renderer components as illumination model, luminaire model, and the spatial distribution of objects, considerable speedups are possible. Fujimoto et al. [4] cite rendering times that are practically independent of the number of objects in the scene (i.e., $O(N_{\text{ray}} D)$) but this is quite optimistic.

In classical radiosity, the computation of form factors is $O(N_{\text{obj}}^2)$ and the (iterative) inversion of the transfer matrix is as high as $O(N_{\text{obj}}^3)$ but easily can be made $O(N_{\text{obj}}^2)$. Considerable speedups of radiosity are possible with hierarchical radiosity, as presented by Hanrahan et al. in [6]. Smits et al. in [12] extended this work by introducing clustering. As reported there, hierarchical radiosity has a complexity of $O(N_{\text{surf}}^2 + N_{\text{patch}})$, where N_{surf} is the number of initial surfaces (proportional to N_{obj}) and N_{patch} is the final number of patches (which is at least a linear function of N_{surf}). Clustered radiosity, on the other hand, has a complexity of $O(N_{\text{surf}} \log N_{\text{surf}} + N_{\text{patch}})$. The linear or superlinear dependence of N_{patch} on N_{surf} has given rise to a considerable amount of interest in optimal meshing schemes. Obviously, a clever adversary can defeat these speed-up schemes, but in practice the expected values apply.

In *Lucifer*, interactions are never between objects, but between a cell's walls and either the walls of neighboring cells or the object contained within the cell. Even though a particular cell may need to be balanced more than once, the number of times that needs to happen is dependent on the scene geometry, not on N_{cell}. If the proportion of power that comes back to the cell is fixed, the number of times a cell has to be balanced is proportional to the log of the tolerance for undistributed power. Hence, as may be inferred from Figure 2 the asymptotic efficiency of *Lucifer* is $O(T_{\text{cell}} N_{\text{cell}})$ where T_{cell} is the cost of operation within a cell (a function of the maximum resolution used to express the radiance). Since our spatial partitioning scheme guarantees the number of objects, N_{cell} is $O(N_{\text{obj}})$, we claim that *Lucifer* is intrinsically $O(N_{\text{obj}})$ – asymptotically more efficient than other global illumination schemes.

It is important to note that in these techniques visibility is an important part (often the most important part) of the cost, and what makes it superlinear. *Lucifer* in a real sense "clusters" visibility, since it is carried along with the radiance representation from cell to cell. As an example, consider a light source blocked by a large object. Before the blocker the light flux is represented normally, and small shadows will be carried along as well. After the blocker a large portion of the light flux will be zero, the small shadows included in it will be absorbed, and any scheme capable of compressing the light flux data will take advantage of this, and objects and cells in the shadow will not even interact with this light front.

To illustrate the trade-off consider the following scene. We have $N_{\mathrm{obj}} \sim 10^7$ objects[3]. Any approach that has to consider pairs (either for explicit light transfer or for visibility) will have to deal with an order of 10^{14} such interactions. To represent light fluxes at an acceptable level of detail, assume that the space is 10m by 10m, and we want about 1 cm accuracy. Assuming equivalent accuracy for directions, that means 10^{12} *elements* to represent the light flux on a virtual wall across the room. If compression of some kind can achieve a 100:1 ratio (which is not unrealistic, as we shall see below), then we have 10^{10} elements. It is harder to estimate how many interactions (reflection, transmission) each element will be involved in, but in a hierarchical scheme (which we will use) it is only a constant number (a function of the width of support of the basis functions used). This has to be multiplied by the number of expected iterations, but there is still enough margin between 10^{10} and 10^{14} to be able to win. The most important point here is that the advantage is expected to increase as N_{obj} increases.

2.7 Radiance Representation

Radiance on a cell wall is a potentially discontinuous, generally nonanalytic function of four variables (2 positional, 2 directional). We can represent radiance L at a point $\mathbf{P}$ on a cell wall or a surface and in a direction (μ_x, μ_y) with a finite element expansion with N_f degrees of freedom:

$$L(\mathbf{P}, \mu_x, \mu_y) = \sum_{i=1}^{N_f} a_i B_i(\mathbf{P}, \mu_x, \mu_y) \tag{6}$$

Even though the basis functions B_i may take on values for $\mu_x^2 + \mu_y^2 \geq 1$, we can disregard their behaviour there, since we never evaluate them in that region.

Choices for B_i include: box discretization, Fourier, discrete cosine, orthogonal polynomials, "light fields" (as described by Levoy and Hanrahan in [7]) and wavelets.

2.8 Box Discretization

For box discretization, as done by FFOC, the B_i are constant within a quantized direction for each quantized position.

Figure 3 shows the result of *Lucifer* using box discretization on a simple model. This image illustrates some of the effects possible with *Lucifer* – diffuse surfaces, color bleeding, soft shadows, and mirror reflection – all produced with the same rendering technique.

There are various ways to perform propagation and object interaction of the discretized radiances. FFOC describe one possible way, equivalent to the one we used for Figure 3. They have a common shortcoming, however: by quantizing angles and positions, box discretization causes various rendering artifacts. These can be reduced

[3] We should note here that in the *Lucifer* context, the definition of the term "object" might be more powerful than in some others.

by increasing the number of quantized positions or directions, but this causes memory requirements to increase dramatically, even with dynamic allocation of memory.

In practice, the CPU time and memory required for box discretization is limiting. Figure 3, for example, required about 9.3MB of memory and 3.2 CPU-hours on an SGI Indigo 2 workstation.

2.9 Wavelets

Wavelets are a promising alternative to other basis functions. They were successfully applied to radiosity solutions by Gortler, et al. [5] and Schröder, et al. [11]. Schröder and Hanrahan [10] and Christensen, et al. [2] extended this work from radiosity to the representation of radiance itself. In this section, we will consider the general applicability of such a wavelet representation of radiance to *Lucifer*.

There are two properties of particular interest for radiance representation. First is their ability to approximate L^2 functions, even those with discontinuities, with a relatively sparse set of coefficients.

There is an infinite number of base scaling functions $\phi(x)$ and therefore an infinite number of possible wavelets. Part of our work will be to find which wavelet basis works best for radiance representation. Most importantly, we need to evalutate the tradeoff between improving approximation by increasing N_v, the number of vanishing moments, and minimizing operation count by reducing W_h, the size ("support") of $\{h_m\}$, the set of wavelet coefficients.

The second property of wavelets relevant to *Lucifer* is their dyadic nature, which corresponds well with our octree spatial subdivision scheme. Splitting a cell whose walls represent radiance with with wavelets amounts to a partitioning of the wavelet coefficients.

3 Ongoing Research

Can we use wavelets to enhance the efficiency of *Lucifer*? The answer to this question is the focus of our current research. We believe wavelets may be useful at two levels.

The first level is data compression and filtering. We maintain radiance coefficients in compressed form on cell walls and only expand them for the cell we are currently balancing and its neighboring walls. Balancing and propagation are otherwise unchanged from the box discretization case.

The second level moves beyond the first to direct evaluation. In this, we do away with sampled radiances completely and propagate, reflect, and transport wavelet coefficients directly from incoming cell wall to outgoing cell wall, including surface interaction. A description of this work is in [9] (an updated and more widely-available version is forthcoming). A similar application of wavelets representing radiance has been done by Christensen, et al. in [2].

Space does not permit us to go into more detail, but we can illustrate the current state of our research with the image shown in Figure 4. In it, light diffusing through a stained glass window creates a "projective image" on a floor. This particular image is reconstructed from a wavelet representation of the radiance falling on the floor. This representation may be treated as a four-dimensional texture and sampled by an otherwise conventional raytracer to produce the reflected radiance seen by an observer at any point.

We have completed the renderer *Lucifer 1.0*, a reimplementation of FFOC's *FIAT* renderer. We will next enhance it to support a wavelet radiance representation.

References

[1] James Arvo and David Kirk. A survey of ray tracing acceleration techniques. In Andrew S. Glassner, editor, *An Introduction to Ray Tracing*, pages 201–262. Academic Press, 1989.

[2] Per H. Christensen, Eric J. Stollnitz, David H. Salesin, and Tony D. DeRose. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics*, 15(1):37–71, January 1996.

[3] A. Fournier, E. Fiume, M. Ouellette, and C. K. Chee. Fiat lux: Light driven global illumination. Technical Memo DGP89–1, Dynamic Graphics Project, Department of Computer Science, University of Toronto, 1989.

[4] Akira Fujimoto, Takayuki Tanaka, and Kansei Iwata. Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications*, pages 16–26, April 1986.

[5] Steven J. Gortler, Peter Schroder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 221–230, 1993.

[6] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.

[7] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH '96 (to appear)*, 1996.

[8] Robert R. Lewis. Making shaders more physically plausible. *Computer Graphics Forum*, 13(2):109 – 120, June 1994.

[9] Robert R. Lewis. Wavelet radiative transport and surface interaction. In John C. Hart, editor, *Proceedings of the Seventh Western Computer Graphics Symposium*, pages 73–83, March 1996.

[10] Peter Schroder and Pat Hanrahan. Wavelet methods for radiance computations. In *Fifth Eurographics Workshop on Rendering*, pages 303–311, Darmstadt, Germany, June 1994.

[11] Peter Schroeder, Steven Gortler, Michael Cohen, and Pat Hanrahan. Wavelet projections for radiosity. In Michael F. Cohen, Claude Puech, and Francois Sillion, editors, *Fourth Eurographics Workshop on Rendering*, pages 105–114. Eurographics, June 1993. held in Paris, France, 14–16 June 1993.

[12] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

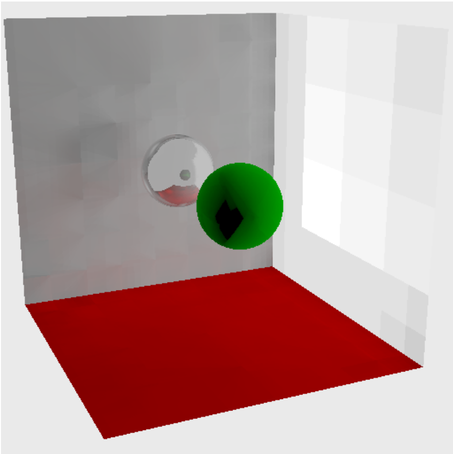

Figure 3: Example of Box Discretization. Two spheres (one a diffuse green reflector, one a mirror) are suspended in a corner. The right wall is a diffuse light source. Diffuse surfaces, color bleeding, soft shadows, and mirror reflection are all visible. (Lewis and Fournier)

Figure 4: Example of Wavelet Propagation. A wavelet decomposition of light shining through the stained glass window propagates to the floor. (Lewis and Fournier)