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

31 March, 1995

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

Illumination is the study of how light interacts with matter to produce visible scenes. In computer graphics, we use illumination to produce "realistic" images. Illumination studies 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. Figure 1 shows the typical geometry and and nomenclature for local illumination studies. The symbols are defined in Table 1. We have attempted to be compatible with the ANSI/IES standard [ans86] wherever possible.

The fundamental equation describing local illumination is

$$\begin{aligned} L &= L_e + \int_{\Omega_{\mathbf{N}}^R} f_r(\mathbf{S}', \mathbf{V})\, L_i\, |\mathbf{N}\cdot\mathbf{S}'|\, d\omega_i' \\ &+ \int_{\Omega_{\mathbf{N}}^T} f_t(\mathbf{S}', \mathbf{V})\, L_i\, |\mathbf{N}\cdot\mathbf{S}'|\, d\omega_i' \qquad (1) \end{aligned}$$

where L is the total radiance given off (either L_r or L_t), L_e is the surface emissivity, L_i is the incident radiance, $\Omega_{\mathbf{N}}^R$ is the reflection hemisphere (contains $\mathbf{V}$), $\Omega_{\mathbf{N}}^T$ is the transmission hemisphere (opposite $\Omega_{\mathbf{N}}^R$), f_r is the bidirectional reflectance distribution function (BRDF), f_t is the bidirectional transmittance distribution function (BTDF), and $d\omega_i$ is $\sin\theta_i d\theta_i d\phi_i$. We use the $'$ to indicate bound variables of integration. A local illumination solution is entirely characterized by its BRDF and BTDF functions.

Global illumination describes how light is distributed in a *scene*: a collection of objects, including light sources, immersed in a given medium. Global illumination solutions must consider multiple reflections. The solution scope of global illumination may vary. We may be interested in determining illumination at a few points (e.g. the amount of light reaching reading surfaces in the design of an office), on visible surfaces (the "viewer-dependent" solution), on all surfaces (the "viewer-independent" solution), or

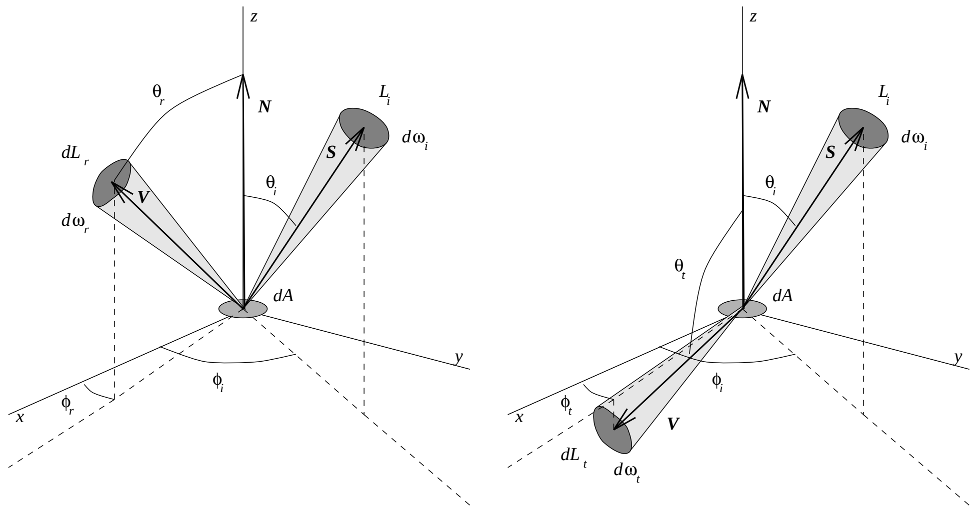

Figure 1: Geometry and Nomenclature for Reflection and Refraction in Local Illumination

throughout a volume (the "participating media" solution).

Global illumination solutions are built on top of local illumination solutions. Figure 2 illustrates a typical global illumination problem, in this case a viewer-dependent one.

We will summarize here the work we've done so far in developing a light-driven global illumination (LDGI), 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 [four89] (hereafter referred to as "FFOC").

1.1 Anatomy of a Renderer

Renderers, which are systems to create images, include:

- a *source model*: how light sources are represented
- an *object model*: how objects in the scene are represented
- a *surface model*: how small-scale surface structure is represented (e. g. texturing)

symbol	definition
dA	element of surface area
ϕ_i	incident azimuthal angle
ϕ_r	reflected azimuthal angle
ϕ_t	transmitted azimuthal angle
L_i	incident radiance
L_r	reflected radiance
L_t	transmitted radiance
$\mathbf{N}$	the surface normal
$\mathbf{S}$	incident direction
θ_i	incident polar angle
θ_r	reflected polar angle
θ_t	transmitted polar angle
$\mathbf{V}$	direction of viewer

Table 1: Key to Figure 1

- an *illumination model*: how light interacts with the surface of an object
- a *propagation model*: how light travels from source to surface, surface to surface, and surface to viewer
- a *shading model*: how light passing to the viewer is converted into an image
- a *rendering technique*: how all the models that make up the package interact to produce an image

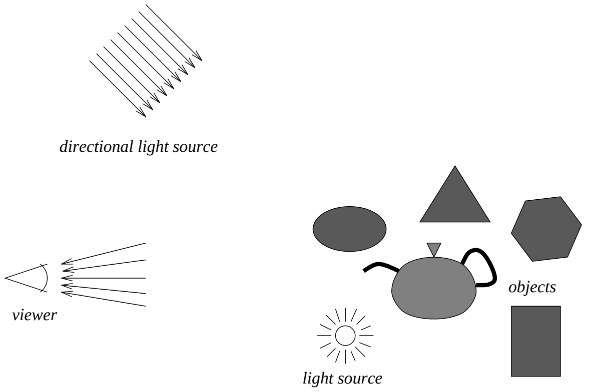

Figure 2: A Typical Global Illumination Problem

All of these are necessary for a rendering package, although for some packages they can be trivial.

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 by setting either f_r and, where necessary, f_t to be Dirac δ-functions, which permits (mirrorlike) reflection and refraction, or by setting L_i to be a Dirac 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 $\frac{1}{\pi}B$, where B is the radiosity of the patch.

Table 2 lists a number of illumination effects and whether or not various rendering techniques are capable of them. Of course, if we apply it to specific renderers, this table is an oversimplification. Some renderers which refer to themselves as raytracers, for instance, are capable of diffuse effects. This is because those renderers have incorporated radiosity or Monte Carlo techniques in hybrid form and are not "pure" raytracers in that sense. For this reason, Table 2 should not be taken to refer to specific renderers, only to rendering techniques.

Note that Monte Carlo is capable of producing all of the effects listed in Table 2. As in many other areas in which it is applied, however, Monte Carlo techniques exhibit slow convergence.

Can we devise a technique that will do produce all these effects without the slow convergence of Monte Carlo? One way to do this would be to have a global illumination scheme that allowed a wider range of local illumination models.

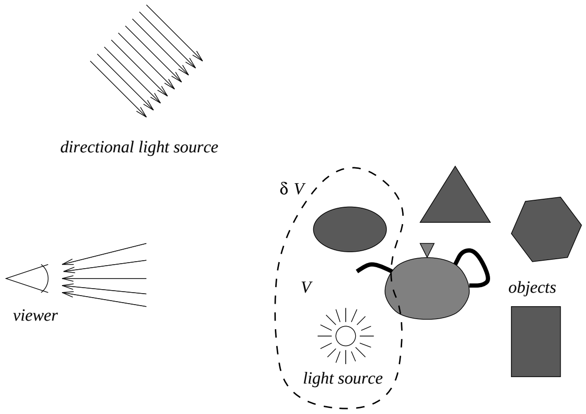

Figure 3: Example of Isolation

2 Light-Driven Global Illumination

Let us consider a computational analogue of how energy distributes itself in a scene.

2.1 Isolation

For possibly complex scenes, we can take a divide-and-conquer approach based on the principle of *isolation*, illustrated in Figure 3.

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.

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

Table 2: Effects Available with Current Rendering Techniques

2.2 Power Computation for Isolated Volumes

For any volume 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

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

is the flux entering V_i,

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

is the flux leaving V_i,

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

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

If, during the distribution process, we ensure that no more energy leaves 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 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 [lewi94] and have considered several illumination models commonly in use in computer graphics from the aspects of both energy conservation and Helmholtz reciprocity[1].

2.4 Spatial Partitioning

[1]Note that LDGI as presented here does not require reciprocity.

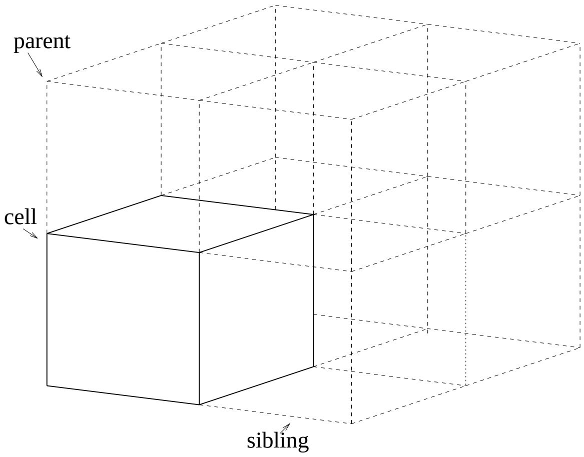

Figure 4: Octree Nomenclature

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, as shown in Figure 4.

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 Light-Driven Global Illumination – The Algorithm

Figure 5 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, `sumUnshotPower(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

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

    q = createPriorityQueue(rootCell(s));
    while (sumUnshotPower(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 5: The Light-Driven Global Illumination Algorithm

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 deal with[2], 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.

[2]This could mean, for instance, that it contained a single primitive object.

2.6 Light Through a Window

The radiance on the cell wall is a potentially discontinuous, generally nonanalytic function of four variables (2 positional, 2 directional).

Before proceeding further, let us transform our definition of direction, however, by defining

$$\begin{aligned} \kappa &= \sin\theta\,\cos\phi \\ \lambda &= \sin\theta\,\sin\phi \end{aligned} \qquad (6)$$

This avoids discrepancies close to $\theta = 0$ and makes for a more uniform grid.

Let us consider an example of what L on a cell wall looks like. Figure 6 shows an instance of the "balls" model from Haines's Standard Procedural Database of test models for raytracers (described in [hain87]) rendered by a typical raytracing renderer.

We have adapted this renderer to perform 4-dimensional ray tracing after a "light through a window" model. In this, we imagine light from a scene going through a window and sampling the radiance over the incoming hemisphere of directions on a uniformly spaced grid of positions. (In LDGI, the "window" corresponds to a cell wall, but we'll discuss the more general case here.) The modified renderer permits us to create an array of images, each image corresponding to a single direction (Figure 7) or a single position on the window (Figure 8). In the former case, the individual images are parallel projections and in the latter case they are "fisheye" views. Note that, as in a perspective view, a parallel projection of a sphere is in general not round.

From these examples, we can see that a 4-dimensional representation of radiance exhibits the same combination of discontinuities and relatively smooth areas we find in 2-dimensional images.

2.7 Radiance Representation

We can represent radiance L at a point $\mathbf{P}$ and in a direction (κ, λ) with a finite element expansion with N_f degrees of freedom:

$$L(\mathbf{P}, \kappa, \lambda) = \sum_{i=1}^{N_f} a_i B_i(\mathbf{P}, \kappa, \lambda) \qquad (7)$$

Even though the basis functions B_i may take on values for $\kappa^2 + \lambda^2 \leq 1$, 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, and wavelets.

2.8 Box Discretization

Let us consider the particular case of box discretization, as was done by FFOC. For box discretization, the B_i are constant within a quantized direction for each quantized position.

Figure 9 shows the result of LDGI using a box discretization on a simple model consisting of three squares forming one corner of a cell with a sphere in the middle. The square on the upper right is a diffuse white emitter and the other two squares are diffuse red (bottom) and gray (upper left) reflectors. The sphere is a mirror.

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 9. They have a common shortcoming, however: by quantizing angles and positions, box discretization causes various rendering artifacts. These can be reduced 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 9, for example, required about 35MB of memory and 2.5 IBM RS/6000 CPU hours.

2.9 Wavelets

Wavelets are a promising alternative to other basis functions. We include a brief summary of their properties in Appendix A. There are two 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. As an example of this, we apply wavelet compression to the 4-dimensional data shown in Figure 7. On the left of Figure 10 is a magnified view of one image from Figure 7 in a particular direction, on the right is a view in the same direction that was reconstructed from a 4-dimensional wavelet transform of the original data compressed by 96% – only the top 4% (in magnitude) of the original coefficients were retained. It is important to note that this compression was in all four dimensions of the data *prior* to reconstruction.

There are an infinite number of base scaling functions $\psi(x)$ and therefore an infinite number of possible wavelets. For the example above, we used the $L = 2$ "Coiflet" wavelet described in [daub92], for these, but other wavelets gave qualitatively similar results. 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 (see Appendix A) and minimizing operation count by reducing the size of $\{h_m\}$.

Other considerations in selecting a wavelet include:

- Is $\phi(x)$ symmetric about some value of x?
- Does $\phi(x)$ need to have an analytical form? (This leads to biorthogonal wavelets.)
- Does refinement of the wavelet interpolate the coarser values?
- What is the tradeoff between the size ("support") of $\{h_m\}$ ($\equiv W_h$) and representing discontinuities compactly

The second property of wavelets relevant to LDGI is their dyadic nature, which corresponds well with our octree spatial subdivision scheme. Splitting a cell whose walls represent radiance with with wavelets

3 Ongoing Research

Can we use wavelets to enhance the efficiency of LDGI? The answer to this question is the focus of our current research. We believe wavelets may be useful in two capacities.

The first 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. In this capacity, we would be able to address the "clustering" problem in global illumination: rendering a scene with N_{obj} objects requires $O(N_{\text{obj}}^2)$ object-object interactions, even if some of the paired objects are very far away from each other[3]. In LDGI, interactions always take place between objects and cell walls, which, with LDGI's spatial partitioning scheme, is $O(N_{\text{obj}})$. Simplification of the light transport takes place automatically during propagation.

The second capacity is direct evaluation. In this, we do away with sampled radiances completely and propagate, reflect, and transmit wavelet coefficients directly from incoming cell wall to outgoing cell wall. A similar application of wavelets representing radiance, although in a radiosity-like renderer has been done by Christensen, et al. in [chri94].

3.1 Work Already Completed

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

We have implemented the energy-conserving illumination models we described in [lewi94].

We have written a public domain wavelet package called `wvlt`, which was published along with the notes for the "Wavelets and Their Applications in Computer Graphics" course on the SIGGRAPH '94 Course Notes CD-ROM. An updated version is available online via the "Wavelets at Imager" World Wide Web page (URL: `http://www.cs.ubc.ca/nest/imager/contributions/bobl/wvlt/top.html`) or by anonymous FTP (node: `ftp.cs.ubc.ca`, file: `/pub/local/bobl/wvlt/wvlt_r1_3.shar`).

[3] A typical example of a clustering problem is representing the spines of a set of multicoloured books on a bookshelf at large distances with much fewer than one patch per book.

3.2 Proposed Experiments

In this section, we will discuss the experiments we intend to perform. The purpose of these experiments is to contrast LDGI with other global and local rendering techniques.

We will create test scenes in four classes and assign a renderer to each class. For each test scene, we will generate one image with the class renderer and one with *lucifer*. For Classes 1 to 3, our goal is to compare favorably with the class renderer. Class 4 includes effects that no known renderer does well on, so in that case the goal is just to produce an acceptable image.

One criterion we will deliberately downplay is speed. Our goal in LDGI is to show its functionality rather than its efficiency. Once we have established what we hope to be the wide range of effects available, this can be an added incentive to improve the speed of implementation of the algorithm.

The four classes we are considering are:

- Class 1 (Local Effects): We will choose scenes in Class 1 to show effects that local illumination techniques do well: diffuse surfaces, specular highlights, reflection, refraction, and sharp shadows. The Class 1 renderer will be Kolb's *Rayshade* [kolb91], a strictly local renderer. A likely source of Class 1 models is Haines's [hain87] Standard Procedural Database.
- Class 2 (Global Effects): Scenes in this class will be chosen to show effects that radiosity schemes do well: soft shadows, diffuse lighting, and diffuse interreflection (color bleeding). The Class 2 renderer will be Pattanaik's [patt91] *rad*, a conventional radiosity-based global renderer. We will use the standard "Cornell Red-Blue Box" both empty (as used by Goral [gora84]) and with the two boxes and diffuse light source used in (for example) Plate 1 of Cohen and Wallace [cohe93]. There are also test cases suggested informally by Rushmeier [rush92].
- Class 3 (Mixed Local and Global Effects): Scenes in this class combine effects used in Classes 1 and 2 in the same scene. We will construct these scenes ourselves. The Class 3 renderer will be Ward's *Radiance* [ward94], a local renderer with global enhancements that has achieved popularity in the illumination engineering community. *Radiance* combines local and global rendering techniques.
- Class 4 (Difficult Effects): Scenes in this class will be chosen to show effects that neither exclusively local nor exclusively global renderers do well: caustics, specular interreflection, projective imaging[4] (a special case of caustics), and clustering. The Class 4 renderer will again be *Radiance*. In these cases, we will develop our own models.

As mentioned above, unlike the other three classes, our criterion here is simply to create a credible image, since we anticipate that *Radiance* will have difficulty with these. (We will verify this.)

A Wavelets

Let us briefly discuss some of the properties of the multiresolution basis functions known as *wavelets*. Much of this is taken from Reissell [reis94], although a similar and more easily obtainable treatment is contained in Daubechies [daub92].

A.1 Fundamental Wavelet Properties

Wavelets are built from *scaling functions*, which we define by dilations and translations of a base scaling function $\phi(x)$ of the form:

$$\phi_{lm}(x) = 2^{-l/2}\phi(2^{-l}x - m) \tag{8}$$

each level l corresponds to a function space V_l, which is part of a nested sequence of subspaces $\ldots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \ldots$ with these properties:

- the union of all V_l spans L^2
- $f(x) \in V_l \rightarrow f(x+k) \in V_l$
- $f(x) \in V_l \leftrightarrow f(2^l x) \in V_0$
- any $f(x) \in V_l$ has a unique representation as a linear combination of $\phi_{lm}(x)$'s

[4] For example, a image of a set of luminous polygons projected through a lens onto a diffuse polygon.

We define a wavelet function space W_l as composed of those functions that need to be added to a given space V_l to span the next finer space V_{l+1}: $V_{l+1} = V_l \oplus W_l$. The basis functions for W_l are also dilations and translations of a mother ("parent"?) wavelet $\psi(x)$:

$$\psi_{lm}(x) = 2^{-l/2}\psi(2^{-l}x - m) \tag{9}$$

Since $\phi(x) \in V_0$ and $V_0 \subset V_1$, we can write $\phi(x)$ as a linear combination of the basis functions $\phi(2x - m)$ for V_1:

$$\phi(x) = \sqrt{2}\sum_m h_m \phi(2x - m) \tag{10}$$

This also holds for ψ:

$$\psi(x) = \sqrt{2}\sum_m g_m \phi(2x - m) \tag{11}$$

These are the *dilation* or *refinement equations*. They are the essence of multiresolution analysis. Wavelet bases differ principally in their choice of $\{h_m\}$ (which turns out to determine $\{g_m\}$).

Let $P_l f$ be the projection of a function $f \in L^2$ into the subspace V_l:

$$P_l f(x) = \sum_m < f, \phi_{lm} > \phi_{lm}(x) \tag{12}$$

It can be shown

$$\| f - P_l f \| \leq C 2^{-lN_v} \sqrt{\sum_n \| f^{(n)} \|^2} \tag{13}$$

where N_v is the number of vanishing moments of the wavelets, i. e. for $n = 0, \ldots, N_v - 1$

$$\int x^n \psi(x) dx = 0 \tag{14}$$

A.2 Wavelet Compression

Given a set of data $s_{lm}, m = 0 \ldots 2^l - 1$, we treat these as coefficients of ϕ_{lm} and can compute a fast wavelet transform in $O(W_h 2^l)$ operations. The fast wavelet transform can be thought of as a series of matrix multiplications by sparse coefficient matrices $H_{l'}$ and $G_{l'}$, $1 \leq l' \leq l$. After we do this, we've still got 2^l coefficients.

Major benefits of the wavelet transform arise from the fact that for piecewise-smooth data, many of the transformed coefficients are relatively small and can, under certain circumstances, be ignored, allowing a sparse representation of data which is initially non-sparse.

It can be shown that if f is of the form

$$f(x) = \sum_{l,m} w_{lm}\psi_{lm}(x) \tag{15}$$

and we approximate it by

$$\tilde{f}(x) = \sum_{w_{lm} \in \tilde{W}} w_{lm}\psi_{lm}(x) \tag{16}$$

where the $\tilde{W}$ is a subset of the set of coefficents $\{w_{lm}\}$, that

$$\| f - \tilde{f} \|^2 = \sum_{w_{lm} \notin \tilde{W}} |w_{lm}|^2 \tag{17}$$

so the sum of the squared magnitudes of the coefficients we discard for compression is an error metric.

References

[ans86] American National Standards Institute/Illuminating Engineering Society of North America. *Nomenclature and Definitions for Illuminating Engineering*, ansi/ies rp-16-1986 edition, June 1986.

[chri94] Per Christensen, Eric J. Stollnitz, David H. Salesin, and Tony DeRose. "Global Illumination of Glossy Environments using Wavelets and Importance". Technical Report UW-CSE-94-10-01, University of Washington Department of Computer Science and Engineering, November 1994.

[cohe93] Michael F. Cohen and John R. Wallace. *Radiosity and realistic image synthesis.* Academic Press, 1993.

[daub92] Ingrid Daubechies. *Ten Lectures on Wavelets*, Vol. 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM, Philadelphia, PA, 1992.

[four89] 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.

[gora84] Cindy M. Goral, Kenneth K. Torrance, Donald P. Greenberg, and Bennett Battaile. "Modelling the Interaction of Light Between Diffuse Surfaces". *Computer Graphics (SIGGRAPH '84 Proceedings)*, Vol. 18, No. 3, pp. 213–222, July 1984.

[hain87] Eric Haines. "A Proposal for Standard Graphics Environments". *IEEE Computer Graphics and Applications*, Vol. 7, No. 11, pp. 3–5, November 1987.

[kolb91] Craig E. Kolb. *Rayshade User's Guide and Reference Manual*, release 4.0 edition, January 1991.

[lewi94] Robert R. Lewis. "Making Shaders More Physically Plausible". *Computer Graphics Forum*, Vol. 13, No. 2, pp. 109 – 120, June 1994.

[patt91] S. N. Pattanaik. "Rad: The Radiosity Package". public domain software, available via anononymous FTP from site "wuarchive.wustl.edu", 1991.

[reis94] Leena-Maija Reissell. "Multiresolution and Wavelets". SIGGRAPH '94 Course: "Wavelets and Their Applications in Computer Graphics" (available on SIGGRAPH '94 Course Notes CD-ROM), July 1994.

[rush92] Holly Rushmeier. "An Experiment". posting to "globillum" Internet mailing list, October 1992.

[ward94] Gregory J. Ward. "The RADIANCE Lighting Simulation and Rendering System". *SIGGRAPH '94 Conference Proceedings*, pp. 459 – 472, July 1994.

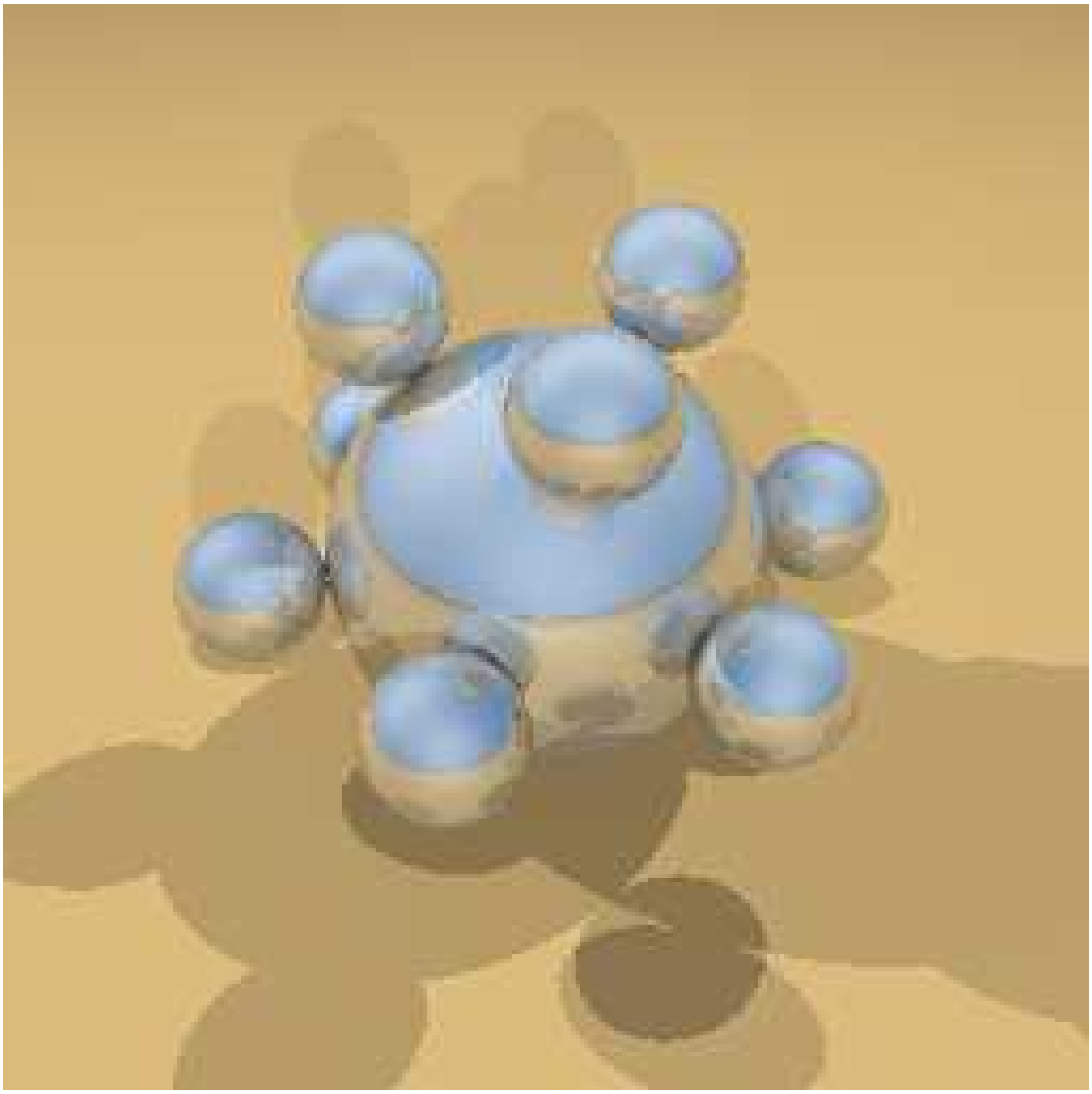

Figure 6: "Balls" Model

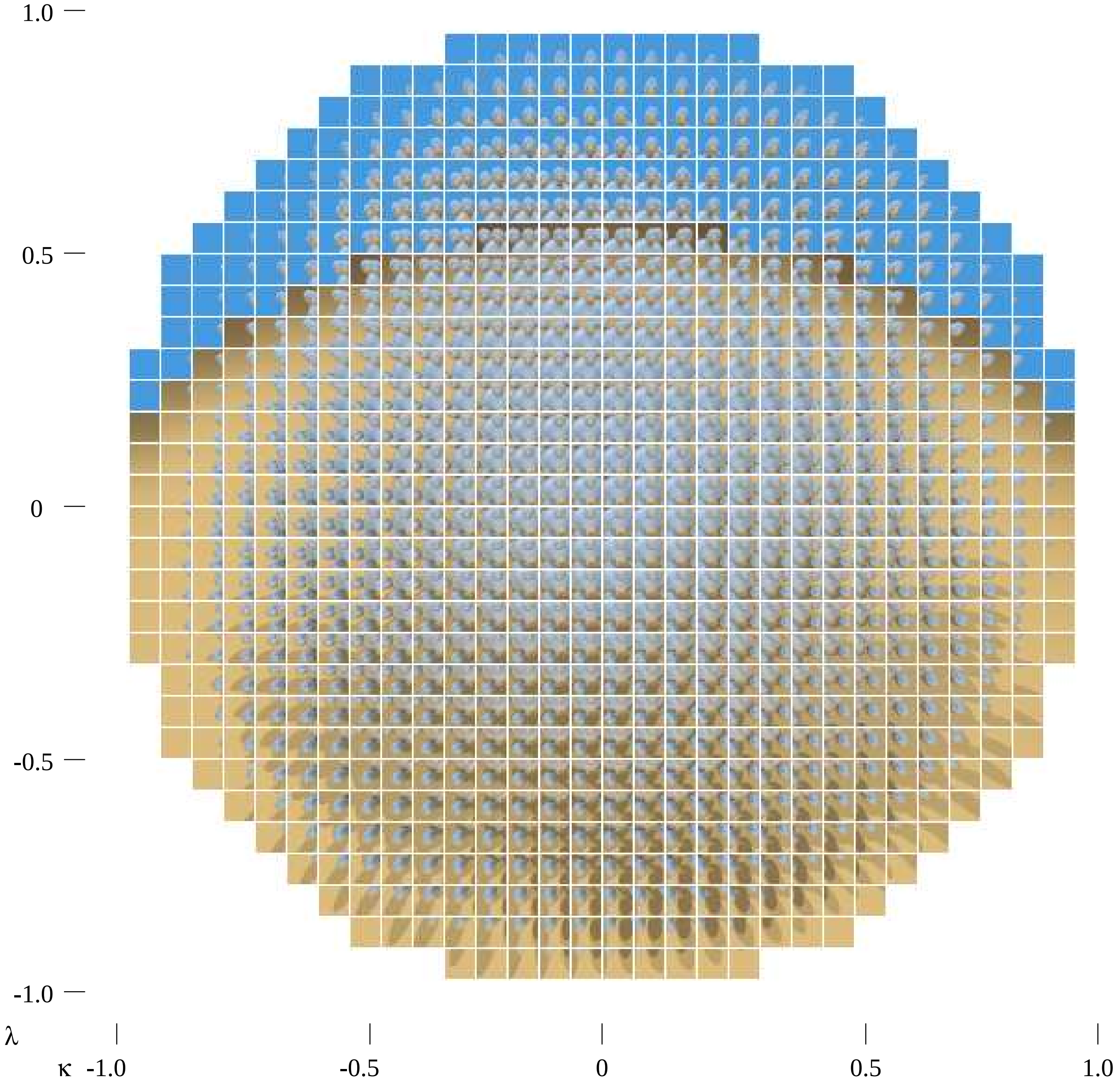

Figure 7: Array of Fixed Direction Views

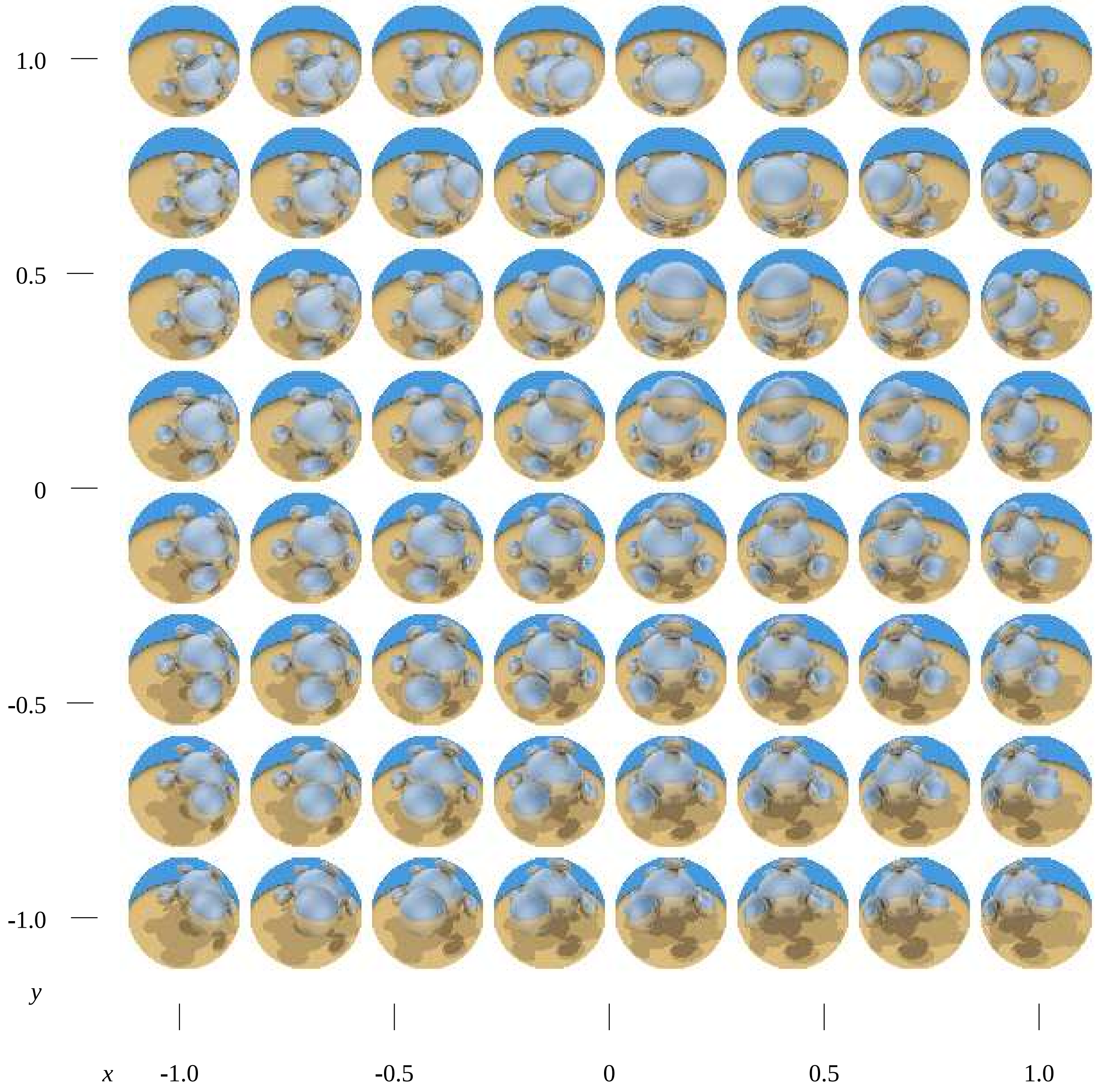

Figure 8: Array of Fixed Position Views

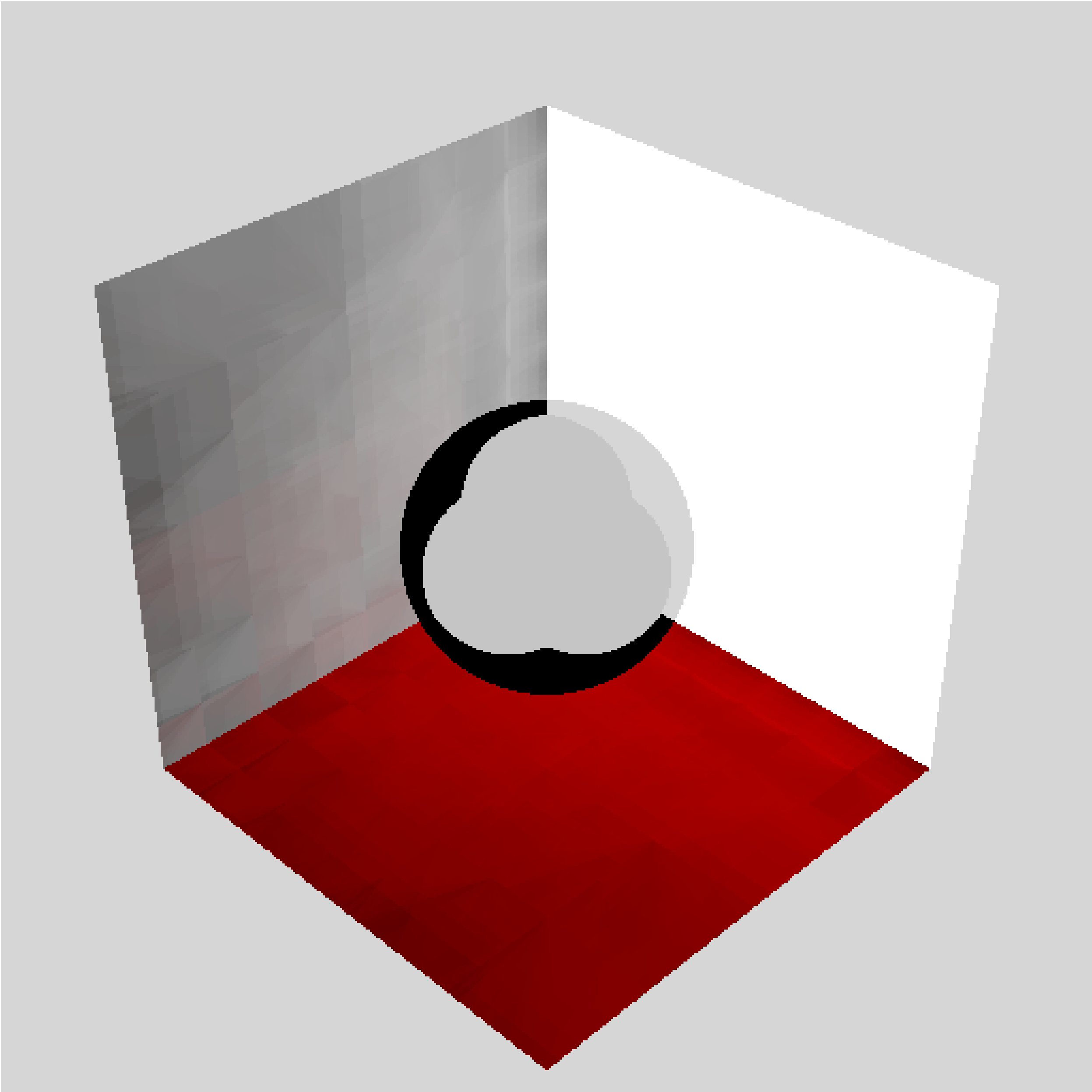

Figure 9: Box Discretization Example.

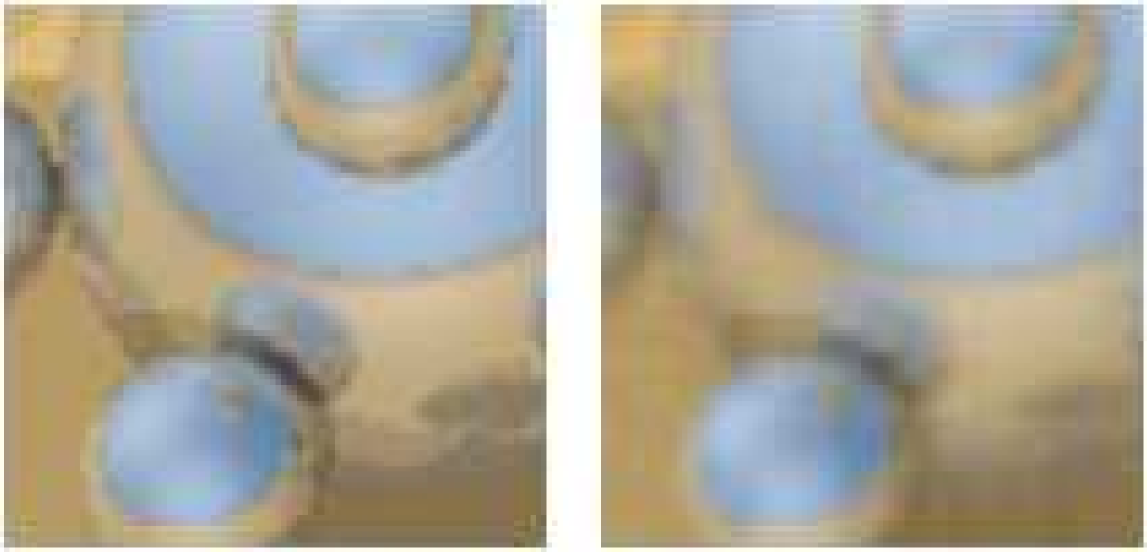

Figure 10: Original (left) and Reconstruction from 4-Dimensional Wavelet Coefficients (right)