Volume 21, (*2002*), *Number yyy pp.*

Wavelet Radiative Transfer and Surface Interaction

Robert R. Lewis
School of Electrical Engineering and Computer Science
Washington State University at Tri-Cities
bobl@tricity.wsu.edu

Alain Fournier
Imager Computer Graphics Laboratory
Department of Computer Science
University of British Columbia
fournier@cs.ubc.ca

Abstract

Recently, there has been considerable interest in the representation of radiance in terms of wavelet basis functions. We will present a coordinate system called Nusselt coordinates *which, when combined with wavelets, considerably simplifies computation of radiative transport and surface interaction. It also provides straightforward computation of the physical quantities involved.*

We show how to construct a discrete representation of the radiative transport operator $\mathcal{T}$ involving inner products of smoothing functions, discuss the possible numerical integration techniques, and present an application. We also show how surface interaction can be represented as a kind of matrix product of the wavelet projections of an incident radiance and a bidirectional reflectance distribution function (BRDF).

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 is typically decomposed into two sub-areas: local and global.

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 nomenclature for local illumination studies.

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. Global illumination solutions are built on top of local illumination solutions. Fournier[7] describes their interrelation further.

In this paper, we will advance a new approach to an illumination solution that is intermediate between local and global illumination. Using wavelets, we are able to treat the interaction between two surfaces and the interaction of a surface with a radiation field in a source-to-destination model that applies to whole surfaces, not just small elements. We have used this work as the basis of a fully global solution (see Lewis[16]).

Wavelets are relatively recent additions to the rendering toolkit. They were first used by Gortler *et al.*[11] to solve the radiosity equations. Schröder et al.[19] and Christensen et al.[2] applied them to non-diffuse situations. Lalonde and Fournier[14] applied them to the representation of reflectance data. What we present here may be considered a further development of work done by these researchers.

2. Radiative Transfer and Surface Interaction Theory

Let us first discuss some of the basics of how light is represented. The fundamental quantity is *radiance*, the amount of power passing in a given direction through a given surface per unit area (perpendicular to the direction of travel) per unit solid angle. In this report, we will take radiance

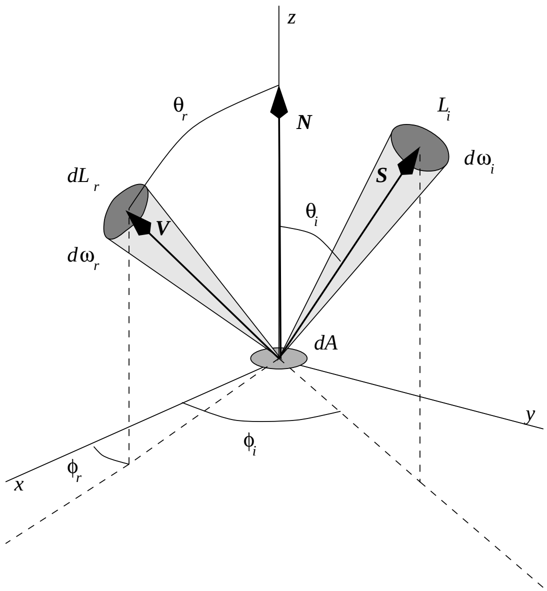

Figure 1: *Local Illumination Geometry*

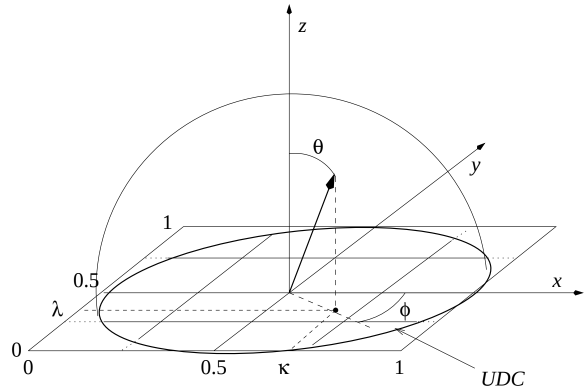

Figure 2: *Nusselt Geometry. "UDC" indicates the unit directional circle.*

to be monochromatic and assume we can construct a polychromatic images by combining independently computed monochromatic images. We also ignore polarization.

Radiance at a point **P** in a direction **S** is usually represented by a function $L(\mathbf{P},\mathbf{S})$.

Radiance's most useful property is its invariance: In a non-participating medium, the radiance given off at a point $\mathbf{P_o}$ on a surface in a direction **S** is constant until reaching another surface. We can express this as the action of a transport operator $\mathcal{T}$:

$$L(\mathbf{P},\mathbf{S}) = \mathcal{T}L = L(\mathbf{P}_o(\mathbf{P},\mathbf{S}),\mathbf{S})$$

This principle underlies raytracing.

We take the fundamental equation describing surface interaction to be (cf. Glassner[9], p. 879 or Cohen and Wallace[4], p. 39):

$$\begin{aligned} L(\mathbf{V}) &= L_e(\mathbf{V}) \\ &+ \int_{\Omega^R_{\mathbf{N}}} f_r(\mathbf{S}^+,\mathbf{V})\, L_i(\mathbf{S}^+) \left|\mathbf{N}\cdot\mathbf{S}^+\right| d\omega_i \\ &+ \int_{\Omega^T_{\mathbf{N}}} f_t(\mathbf{S}^-,\mathbf{V})\, L_i(\mathbf{S}^-) \left|\mathbf{N}\cdot\mathbf{S}^-\right| d\omega_i \end{aligned} \tag{1}$$

where L is the total radiance given off of a surface with normal **N**, L_e is the surface emissivity, L_i is the incident radiance, $\Omega^R_{\mathbf{N}}$ is the reflection hemisphere (contains N, the "viewing" direction **V**, and the "positive source" direction $\mathbf{S}^+$), $\Omega^T_{\mathbf{N}}$ is the transmission hemisphere (opposite $\Omega^R_{\mathbf{N}}$, containing the "negative source" direction $\mathbf{S}^-$), f_r is the bidirectional reflectance distribution function (BRDF), and f_t is the bidirectional transmittance distribution function (BTDF). Note that, since it symbolically factors out of (1), we ignore the spatial variation of the radiances and distribution functions.

We represent this combined reflection and transmission surface interaction by the integral operator $\mathcal{S}$.

$$L_r = L_e + \mathcal{S}L_i$$

Note that we treat the emissivity term separately.

3. Radiance in Nusselt Coordinates

If we confine our discussion to surfaces, we can assume a planar (possibly local) parameterization for **P** of (u,v). **S** is then typically represented in polar and azimuthal coordinates (θ,ϕ) according to the local frame of reference.

Consider the x, y, and z direction cosines corresponding to a direction (θ,ϕ):

$$\mu_x = \sin\theta\cos\phi \quad \mu_y = \sin\theta\sin\phi \quad \mu_z = \cos\theta \tag{2}$$

We take (μ_x,μ_y) to be an alternative parameterization of direction.

It is convenient in what follows for all variables to vary between the extrema of 0 and 1, so let us make a further change of the directional variables from (μ_x,μ_y) to (κ,λ):

$$\kappa = \frac{\mu_x+1}{2} \qquad \lambda = \frac{\mu_y+1}{2} \tag{3}$$

Figure 2 shows the relation between (θ,ϕ) and (κ,λ) graphically. It also shows the *unit directional circle* (hereafter, UDC), defined by $\mu_x^2+\mu_y^2 = 1$.

To convert integration over (θ,ϕ) to integration over (κ,λ) we account for the change of integration variables by multiplying by the determinant of the Jacobian, which is:

$$\left|\frac{\partial(\theta,\phi)}{\partial(\kappa,\lambda)}\right| = \frac{4}{\cos\theta\sin\theta} \tag{4}$$

so, assuming L_i is zero for directions outside the UDC, (1) becomes

$$L = L_e + 4\int_0^1\int_0^1 \Big[f_r(\mathbf{S}^+,\mathbf{V})\,L_i(\mathbf{S}^+) + f_t(\mathbf{S}^-,\mathbf{V})\,L_i(\mathbf{S}^-)\Big]\,d\kappa_i\,d\lambda_i \qquad (5)$$

That the integral no longer contains trigonometric functions should come as no surprise. We have simply used a differential form of the "Nusselt analog" (Nusselt[18], but see Cohen and Wallace[4], p. 80 for a description in English): the amount of power per unit area transferred from a differential solid angle $d\omega_i$ is proportional to $d\kappa_i\,d\lambda_i$, the area of the surface that the projection of $d\omega_i$ on a unit sphere subtends. For this reason, we refer to κ and λ as "Nusselt coordinates".

We also note that, since $\mu_x^2+\mu_y^2+\mu_z^2 = 1$ and since each vector is defined only over a hemisphere, not the whole directional sphere, we can express $\mathbf{S}^+$, $\mathbf{S}^-$, and $\mathbf{V}$ all unambiguously in terms of their respective incident and reflected κ's and λ's. Simply put, it is always clear which sign to attach to the square root.

Other ways to parameterize the directional component of a radiance distribution are possible. Light fields (as in Levoy and Hanrahan[15]) and lumigraphs (as in Gortler *et al.*[10]), are very promising approaches for display purposes. Christensen *et al.*[3] use a combination of a gnomonic projection and "stretch" to map directions to the unit square. None of these approaches, however, leads to the simplification of surface interaction that (5) demonstrates.

4. Radiance Representation

What are the characteristics of a four-dimensional radiance distribution $L(x,y,\kappa,\lambda)$? The easiest way to visualize this is as "light through a window" where the position of an observer on a window is (x,y) and he or she casts a ray in direction (κ,λ). For a fixed direction, the resulting two-dimensional projection is a parallel projection. (The special case $(\kappa,\lambda) = (\frac{1}{2},\frac{1}{2})$ is an orthographic projection.) For a fixed position, the distribution in (κ,λ) would be a "fisheye" view. In both cases, the result is an image, so we can deal with those radiance distributions as we deal with images.

Radiance at a point on a surface is a potentially discontinuous, generally nonanalytic function. We can approximate it with a finite element expansion with N_f degrees of freedom:

$$L(x,y,\kappa,\lambda) = \sum_{j=1}^{N_f} b_j B_j(x,y,\kappa,\lambda)$$

Choices for the basis functions B_j include box discretization (*à la* Fournier *et al.*'s[8] FIAT), Fourier, discrete cosine, orthogonal polynomials, and wavelets. We are particularly interested in wavelets because, unlike the other bases listed, their basis functions are of limited support and they can represent discontinuities compactly. They are also capable of considerable compression.

5. Multidimensional Wavelets

In this section, we will describe multidimensional wavelets with the intention of applying them to radiative transport and surface interaction.

For D-dimensional coordinates

$$\mathbf{q} = (q_1, q_2, \ldots, q_D),$$

we can define a set of multidimensional wavelet basis functions indexed by a *standard multiresolution index*

$$\mathbf{j} = (\nu^j, l_1^j, m_1^j, l_2^j, m_2^j, \ldots, l_D^j, m_D^j) \qquad (6)$$

where ν^j, which we call the "basis selector", determines the combination of one dimensional smoothing and wavelet functions:

$$B_{\mathbf{j}}(\mathbf{q}) = \begin{cases} \phi_{l_1^j m_1^j}(q_1)\phi_{l_2^j m_2^j}(q_2)\ldots\phi_{l_D^j m_D^j}(q_D) & \nu^j = 0 \\ \psi_{l_1^j m_1^j}(q_1)\phi_{l_2^j m_2^j}(q_2)\ldots\phi_{l_D^j m_D^j}(q_D) & \nu^j = 1 \\ \phi_{l_1^j m_1^j}(q_1)\psi_{l_2^j m_2^j}(q_2)\ldots\phi_{l_D^j m_D^j}(q_D) & \nu^j = 2 \\ \qquad\vdots & \\ \psi_{l_1^j m_1^j}(q_1)\psi_{l_2^j m_2^j}(q_2)\ldots\psi_{l_D^j m_D^j}(q_D) & \nu^j = 2^D-1 \end{cases} \qquad (7)$$

We refer to the special case of $\nu^j = 0$ as the "pure smoothing" component, as the corresponding basis function is made up of only smoothing functions.

We define the inner product of two functions f and g with respect to x. (Using x as a subscript is non-traditional, but will become useful when we speak of multidimensional wavelets.)

$$\langle f \mid g\rangle_x \equiv \int_{-\infty}^{+\infty} f(x)g(x)dx$$

If we have $\mathbf{j}$ as in (6) and define $\mathbf{k}$ as

$$\mathbf{k} = (\nu^k, l_1^k, m_1^k, l_2^k, m_2^k, \ldots, l_D^k, m_D^k) \qquad (8)$$

then

$$\left\langle \tilde{B}_{\mathbf{j}} \mid B_{\mathbf{k}}\right\rangle_{\mathbf{q}} = \delta_{\nu^j\nu^k}\prod_{d=1}^{D}\delta_{l_d^j l_d^k}\delta_{m_d^j m_d^k}$$

$\tilde{B}_{\mathbf{j}}$ is identical to $B_{\mathbf{j}}$ as defined in (7), but with the primal wavelets and smoothing functions replaced by their duals.

This is the "standard" multidimensional Cartesian product basis. It is also possible to constrain $l_1^j = l_2^j = \ldots = l_D^j \equiv l^j$, resulting in the so-called "nonstandard" basis. In general multidimensional (especially image-oriented) applications, as cited in Daubechies[5] and in Schröder *et al.*[19], the nonstandard bases are preferred because of their "square" support.

In this paper, we are considering 4-dimensional, nonstandard basis functions, so let us enumerate the coordinates with the 4-vector

$$\mathbf{q} = (u, v, \kappa, \lambda)$$

and the basis functions with a *nonstandard multiresolution index*

$$\mathbf{j} = (\nu, l, m_u, m_v, m_\kappa, m_\lambda). \tag{9}$$

Using the standard wavelet recurrence relations (see Daubechies[5]) and (7), all the basis functions at level l of the pyramid can be written in terms of the $\nu = 0$ basis functions at level $l+1$:

$$B_{\nu l \mathbf{m}}(\mathbf{q}) = \tag{10}$$

$$4 \begin{cases} \sum_{\mathbf{m}'} h_{m'_u} h_{m'_v} h_{m'_\kappa} h_{m'_\lambda} B_{0(l+1)(2\mathbf{m}+\mathbf{m}')}(\mathbf{q}) & \nu = 0 \\ \sum_{\mathbf{m}'} g_{m'_u} h_{m'_v} h_{m'_\kappa} h_{m'_\lambda} B_{0(l+1)(2\mathbf{m}+\mathbf{m}')}(\mathbf{q}) & \nu = 1 \\ \sum_{\mathbf{m}'} h_{m'_u} g_{m'_v} h_{m'_\kappa} h_{m'_\lambda} B_{0(l+1)(2\mathbf{m}+\mathbf{m}')}(\mathbf{q}) & \nu = 2 \\ \vdots & \\ \sum_{\mathbf{m}'} g_{m'_u} g_{m'_v} g_{m'_\kappa} g_{m'_\lambda} B_{0(l+1)(2\mathbf{m}+\mathbf{m}')}(\mathbf{q}) & \nu = 15 \end{cases}$$

where $\mathbf{m}' \equiv (m'_u, m'_v, m'_\kappa, m'_\lambda)$.

So for any function f,

$$\langle f \mid B_{\nu l \mathbf{m}} \rangle_q = \sum_{\mathbf{m}'} W_{\nu \mathbf{m}'} \left\langle f \mid B_{0(l+1)(2\mathbf{m}+\mathbf{m}')} \right\rangle_q$$

where $W_{\nu\mathbf{m}'}$ is (4 times) a product of smoothing and wavelet coefficients.

6. Wavelet Radiance Properties

Apart from compression, representing radiance in terms of a wavelet basis with direction expressed in Nusselt coordinates makes several calculations of relevance to illumination computation easier. Notice that these all act directly on wavelet coefficients themselves and do not require an inverse wavelet transform.

6.1. Irradiance

Irradiance is computed as

$$E(x,y) = \int_{\Omega_{\mathbf{N}}^R} L_i(x,y,\theta,\phi)\, |\mathbf{N}\cdot\mathbf{S}|\, d\omega_i.$$

Again making use of (4), we have

$$E(x,y) = 4 \int_0^1 \int_0^1 L_i(x,y,\kappa,\lambda)\, d\kappa_i\, d\lambda_i.$$

The limits of both integrations are 0 and 1, but if L_i is zero outside the UDC (see Figure 2), we can safely extend the integration limits to $-\infty$ and $+\infty$.

This allows us to say that if

$$L_i(x,y,\kappa,\lambda) = \sum_{\mathbf{j}} b_{\mathbf{j}} B_{\mathbf{j}}(x,y,\kappa,\lambda) \tag{11}$$

then

$$E(x,y) = 4 \sum_{\mathbf{j}} b_{\mathbf{j}} \langle B_{\mathbf{j}} \mid 1 \rangle_{\kappa,\lambda} \tag{12}$$

is the wavelet representation of the irradiance.

The inner products on the right hand side are usually easy to compute in tabular form, if not analytically, making particular use of the fact that $\int_{-\infty}^{+\infty} \psi(x) dx = 0$ to eliminate many coefficients.

6.2. Power Flux

The power flux passing through an area A is defined as

$$\begin{aligned} \Phi &= \int_A \int_{\Omega_{\mathbf{N}}^R} L_i(x,y,\theta,\phi)\, |\mathbf{N}\cdot\mathbf{S}|\, d\omega_i dA \\ &= \int_A E(x,y)\, dA \end{aligned} \tag{13}$$

If we have a spatial parameterization $(u,v) \leftrightarrow (x,y)$ that maps the unit square to A and back, then we have

$$\Phi = \int_0^1 \int_0^1 E(x(u,v), y(u,v)) \left| \frac{\partial(x,y)}{\partial(u,v)} \right| du dv.$$

If, as above, we take $E(x,y) = 0$ for (x,y) outside of A and if we extend (11) to include this parameterization:

$$L_i(x,y,\kappa,\lambda) = \sum_{\mathbf{j}} b_{\mathbf{j}} B_{\mathbf{j}}(u(x,y), v(x,y), \kappa, \lambda),$$

then we can incorporate (12) to get the wavelet representation of flux:

$$\Phi = 4 \sum_{\mathbf{j}} b_{\mathbf{j}} \left\langle B_{\mathbf{j}} \mid \left| \frac{\partial(x,y)}{\partial(u,v)} \right| \right\rangle_q. \tag{14}$$

6.3. Transport

We represent radiance as

$$L(\mathbf{q}) = \sum_{\mathbf{k}} b_{\mathbf{k}} B_{\mathbf{k}}(\mathbf{q}) \tag{15}$$

where

$$b_{\mathbf{k}} = \langle L \mid \tilde{B}_{\mathbf{k}} \rangle_{\mathbf{q}} \tag{16}$$

and $\mathbf{k}$ is defined as in (8).

Radiance travels from a source point $\mathbf{q}_s$ to a destination point $\mathbf{q}_d$. If we have a mapping of $\mathbf{q}_s \to \mathbf{q}_d$, we can compute

$$L_d(\mathbf{q}_d) = \sum_{\mathbf{k}} b^d_{\mathbf{k}} B_{\mathbf{k}}(\mathbf{q}_d)$$

where

$$\begin{aligned} b^d_{\mathbf{k}} &= \langle L_s(\mathbf{q}_s(\cdot)) \mid \tilde{B}_{\mathbf{k}} \rangle_{\mathbf{q}_d} \\ &= \sum_{\mathbf{j}} b^s_{\mathbf{j}} T_{\mathbf{jk}}, \end{aligned} \tag{17}$$

$\mathbf{j}$ is defined as in (6), and we define geometry-dependent "transport coefficients"

$$T_{\mathbf{jk}} \equiv \left\langle B_{\mathbf{j}}(q_s(\cdot)) \mid \tilde{B}_{\mathbf{k}} \right\rangle_{\mathbf{q}_d}. \tag{18}$$

Using the multidimensional refinement shown in (10), given $T_{(0 l_j \mathbf{m}_j)\mathbf{k}}$ on level l_j, we can compute all coefficients on the coarser level above it in the pyramid:

$$T_{(\nu_j (l_j-1)\mathbf{m}_j)\mathbf{k}} = \sum_{\mathbf{m}'} W_{\nu_j \mathbf{m}'} T_{0 l_j (2\mathbf{m}_j+\mathbf{m}')\mathbf{k}} \tag{19}$$

and given $T_{\mathbf{j}(0 l_k \mathbf{m}_k)}$ on level l_k, we can compute

$$T_{\mathbf{j}(\nu_k (l_k-1)\mathbf{m}_k)} = \sum_{\mathbf{m}''} W_{\nu_k \mathbf{m}''} T_{\mathbf{j}(0 l_k (2\mathbf{m}_k+\mathbf{m}''))} \tag{20}$$

where $\mathbf{m}_j = (m^j_u, m^j_v, m^j_\kappa, m^j_\lambda)$ and $\mathbf{m}_k = (m^k_u, m^k_v, m^k_\kappa, m^k_\lambda)$. This means that we can compute all transport coefficients strictly in terms of pure smoothing components:

$$T_{\mathbf{jk}} = \sum_{\mathbf{m}'} \sum_{\mathbf{m}''} W_{\nu_j \mathbf{m}'} W_{\nu_k \mathbf{m}''} T_{(0 l_j (2\mathbf{m}_j+\mathbf{m}'))(0 l_k (2\mathbf{m}_k+\mathbf{m}''))} \tag{21}$$

6.4. Surface Interaction

Using (7), let us define a mixed primal-dual, four-dimensional, nonstandard wavelet basis:

$$F_{\mathbf{j}}(\kappa_s, \lambda_s, \kappa_r, \lambda_r) =$$

$$\begin{cases} \tilde{\phi}_{l^j m^i_\kappa}(\kappa_s)\tilde{\phi}_{l^j m^i_\lambda}(\lambda_s)\phi_{l^j m^j_\kappa}(\kappa_r)\phi_{l^j m^j_\lambda}(\lambda_r) & \nu^j = 0 \\ \tilde{\psi}_{l^j m^i_\kappa}(\kappa_s)\tilde{\phi}_{l^j m^i_\lambda}(\lambda_s)\phi_{l^j m^j_\kappa}(\kappa_r)\phi_{l^j m^j_\lambda}(\lambda_r) & \nu^j = 1 \\ \tilde{\phi}_{l^j m^i_\kappa}(\kappa_s)\tilde{\psi}_{l^j m^i_\lambda}(\lambda_s)\phi_{l^j m^j_\kappa}(\kappa_r)\phi_{l^j m^j_\lambda}(\lambda_r) & \nu^j = 2 \\ \vdots & \\ \tilde{\psi}_{l^j m^i_\kappa}(\kappa_s)\tilde{\psi}_{l^j m^i_\lambda}(\lambda_s)\psi_{l^j m^j_\kappa}(\kappa_r)\psi_{l^j m^j_\lambda}(\lambda_r) & \nu^j = 15 \end{cases}$$

where

$$\mathbf{j} = (\nu_j, l_j, m^i_\kappa, m^i_\lambda, m^j_\kappa, m^j_\lambda)$$

So $F_{\mathbf{j}}(\kappa_s, \lambda_s, \kappa_r, \lambda_r)$ is $B_{\mathbf{j}}(\kappa_s, \lambda_s, \kappa_r, \lambda_r)$ with dual scaling functions and wavelets substituted for the primal scaling functions and wavelets in the incident directional components only.

If we then represent the BRDF in Nusselt coordinates with this basis:

$$f_r(\kappa_s, \lambda_s, \kappa_r, \lambda_r) = \sum_{\mathbf{j}} f^r_{\mathbf{j}} F_{\mathbf{j}}(\kappa_s, \lambda_s, \kappa_r, \lambda_r)$$

and

$$L_i(x, y, \kappa_s, \lambda_s) = \sum_{\mathbf{k}} b_{\mathbf{k}} B_{\mathbf{k}}(x, y, \kappa_s, \lambda_s)$$

where

$$\mathbf{k} = (\nu_k, l_k, m^k_u, m^k_v, m^k_\kappa, m^k_\lambda)$$

then, applying (5) and again requiring either f_r or L_i (or both) to vanish outside the UDC (thus allowing us to extend our integration to $(-\infty..\infty)$), the reflected radiance is

$$\begin{aligned} L_r &= 4\int_0^1 \int_0^1 f_r(\mathbf{S}^+, \mathbf{V})\, L_i(\mathbf{S}^+)\, d\kappa_s\, d\lambda_s \\ &= 4\sum_{\mathbf{j}} \sum_{\mathbf{k}} f^r_{\mathbf{j}} b_{\mathbf{k}} \left\langle F_{\mathbf{j}} \mid B_{\mathbf{k}} \right\rangle_{\kappa_s, \lambda_s} \end{aligned} \tag{22}$$

We seek a wavelet representation of the post-interaction radiance:

$$L_r = \sum_{\mathbf{n}} b^r_{\mathbf{n}} B_{\mathbf{n}}(x, y, \kappa_r, \lambda_r)$$

Again using the basis representation of the reflected radiance as in (15) and (16), it follows that

$$b^r_{\mathbf{n}} = \left\langle L_r \mid \tilde{B}_{\mathbf{n}} \right\rangle_{x, y, \kappa_r, \lambda_r}$$

So, substituting (22), we find

$$b^r_{\mathbf{n}} = 4\sum_{\mathbf{j}} \sum_{\mathbf{k}} f^r_{\mathbf{j}} b_{\mathbf{k}} \left\langle \left\langle F_{\mathbf{j}} \mid B_{\mathbf{k}} \right\rangle_{\kappa_s, \lambda_s} \mid \tilde{B}_{\mathbf{n}} \right\rangle_{x, y, \kappa_r, \lambda_r} \tag{23}$$

Let us now simplify notation. We can rewrite $F_{\mathbf{j}}$ and $B_{\mathbf{k}}$ compactly by defining a function $\xi^\beta_{lm}(x)$ that takes on the value of the smoothing function or the wavelet depending on a single binary value β:

$$\xi^\beta_{lm}(x) = \begin{cases} \phi_{lm}(x) & \beta = 0 \\ \psi_{lm}(x) & \beta = 1 \end{cases}$$

and similarly for a $\tilde{\xi}$ with $\tilde{\phi}$ and $\tilde{\psi}$ in place of ϕ and ψ, respectively.

We also take indexable (from 0 to 3) representations of the arguments to $F_{\mathbf{j}}$ and $B_{\mathbf{k}}$, respectively

$$\mathbf{p} = (\kappa_s, \lambda_s, \kappa_r, \lambda_r)$$

and

$$\mathbf{q} = (u, v, \kappa_s, \lambda_s).$$

If we now adopt the notation where $i_{\langle\alpha\rangle}$ refers to the αth bit of the binary form of i (i.e. $i2^{-\alpha}$ mod 2), we can express the innermost inner product of (23) as

$$\begin{aligned} &\left\langle F_{\mathbf{j}} \mid B_{\mathbf{k}} \right\rangle_{\kappa_s, \lambda_s} = \\ &\int_0^1 \int_0^1 \left[\prod_{\alpha=0}^{1} \tilde{\xi}^{\nu^j_{\langle\alpha\rangle}}_{l^j m^j_\alpha}(p_\alpha) \xi^{\nu^k_{\langle\alpha\rangle}}_{l^k m^k_\alpha}(q_\alpha) \right. \\ &\left. \prod_{\alpha=2}^{3} \xi^{\nu^j_{\langle\alpha\rangle}}_{l^j m^j_\alpha}(p_\alpha) \xi^{\nu^k_{\langle\alpha\rangle}}_{l^k m^k_\alpha}(q_\alpha) \right] d\kappa_s\, d\lambda_s \end{aligned} \tag{24}$$

and removing factors that do not depend on the variables of integration from the integral, we have

$$\begin{aligned} &\left\langle F_{\mathbf{j}} \mid B_{\mathbf{k}} \right\rangle_{\kappa_s, \lambda_s} = \\ &\left\langle \tilde{\xi}^{\nu^j_{\langle 0\rangle}}_{l^j m^j_0} \mid \xi^{\nu^k_{\langle 2\rangle}}_{l^k m^k_2} \right\rangle_{\kappa_s} \left\langle \tilde{\xi}^{\nu^j_{\langle 1\rangle}}_{l^j m^j_1} \mid \xi^{\nu^k_{\langle 3\rangle}}_{l^k m^k_3} \right\rangle_{\lambda_s} \end{aligned}$$

$$\xi^{\nu^k_{\langle 0\rangle}}_{l^k m^k_0}(x)\,\xi^{\nu^k_{\langle 1\rangle}}_{l^k m^k_1}(y)\,\xi^{\nu^j_{\langle 2\rangle}}_{l^j m^j_2}(\kappa_r)\,\xi^{\nu^j_{\langle 3\rangle}}_{l^j m^j_3}(\lambda_r) \qquad (25)$$

If we now define a *multiresolution delta tensor*

$$\Delta^{\sigma\tau}_{\mathbf{jk}} = \left\langle \tilde{\xi}^{\nu^j_{\langle\sigma\rangle}}_{l^j m^j_\sigma} \mid \xi^{\nu^k_{\langle\tau\rangle}}_{l^k m^k_\tau} \right\rangle$$

we can rewrite (25) as

$$\left\langle F_{\mathbf{j}} \mid B_{\mathbf{k}} \right\rangle_{\kappa_s,\lambda_s} = \qquad (26)$$
$$\Delta^{02}_{\mathbf{jk}}\,\Delta^{13}_{\mathbf{jk}}\,\xi^{\nu^k_{\langle 0\rangle}}_{l^k m^k_0}(x)\,\xi^{\nu^k_{\langle 1\rangle}}_{l^k m^k_1}(y)\,\xi^{\nu^j_{\langle 2\rangle}}_{l^j m^j_2}(\kappa_r)\,\xi^{\nu^j_{\langle 3\rangle}}_{l^j m^j_3}(\lambda_r)$$

and we can continue to apply the Δ symbol to simplify the whole nested inner product in (23)

$$\left\langle \left\langle F_{\mathbf{j}} \mid B_{\mathbf{k}} \right\rangle_{\kappa_s,\lambda_s} \mid \tilde{B}_{\mathbf{n}} \right\rangle_{x,y,\kappa_r,\lambda_r} =$$
$$\Delta^{02}_{\mathbf{jk}}\,\Delta^{13}_{\mathbf{jk}}\,\Delta^{00}_{\mathbf{kn}}\,\Delta^{11}_{\mathbf{kn}}\,\Delta^{22}_{\mathbf{jn}}\,\Delta^{33}_{\mathbf{jn}}. \qquad (27)$$

The Δ's are very much dependent upon our choice of wavelet, so we will defer discussing their computation to Section 7.

We can do the same thing with a BTDF and so represent general surface interactions: reflection, refraction, and transmission.

7. Implementation

In this section, we discuss the implementation of the WRT algorithm (hereafter, "*WRT*"). We apply some of the concepts of Section 6 to a classic illumination problem: the transport of radiation between two arbitrarily-oriented polygons.

7.1. Major Implementation Details

In this subsection, we will discuss design decisions that are of general concern to anyone attempting to build a WRT system.

7.1.1. Representing Transport Geometry

To establish the $\mathbf{q}_s \leftrightarrow \mathbf{q}_d$ mapping we need in order to transport radiance from source to destination, we must deal with several coordinate systems, as shown in Figure 3: source parametric, source object, world, destination object, and destination parametric. Obviously, source object to destination object is best done with a conventional affine transform (with projection), but there are several choices possible for the parametric $\leftrightarrow$ object mappings: rectilinear, perspective, and bilinear. All of these are local to the sending or receiving surface.

7.1.1.1. Rectilinear This is the simplest possible mapping:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} W & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

W and H are the dimensions of a bounding rectangle. If the object is a rectangle, an obvious strategy is to choose coordinates in which W is its width and H is its height. This will ensure that the (inverse) transformed object completely fills the unit square. Otherwise, or in the more general case of an arbitrarily-sided polygon, when computing $T_{\mathbf{jk}}$ we must clip the (square) support of $B_{\mathbf{j}}$ (source) or $B_{\mathbf{k}}$ (destination) against the polygon when integrating.

A major advantage of the rectilinear mapping is the ease of computation of the Jacobian determinant:

$$\left| \frac{\partial(x,y)}{\partial(u,v)} \right| = WH$$

which makes the power flux computation shown in Equation (14) very easy:

$$\Phi = 4WH \sum_{\mathbf{j}} b_{\mathbf{j}} \left\langle B_{\mathbf{j}} \mid 1 \right\rangle_q .$$

especially in the case of Haar wavelets:

$$\Phi = 4WHb_{\mathbf{0}}.$$

7.1.1.2. Perspective This mapping allows us to represent the more general quadrilaterals without the need to clip:

$$\begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where the A_{ij}'s are easily-determined functions of the quadrilateral vertices.

A straight line in perspective parametric coordinates $au + bv + c = 0$ transforms to a straight line $a'x + b'y + c'$ in object coordinates. This means that a quadrilateral in object coordinates will map to a quadrilateral in parametric coordinates and vice versa. This has favourable implications for transport coefficient computation that we will discuss below.

One drawback of a perspective mapping is that if the quadrilateral approaches degeneracy (a triangle, for instance), the appearance of a uniform grid in parametric space becomes increasingly nonuniform.

The power flux computation of Equation (14) is not as easy as in the rectilinear case, but may still be analytically done for basis functions $B_{\mathbf{j}}$ with closed-form representations, such as splines.

7.1.1.3. Bilinear As with 2-D perspective, this mapping also allows us to represent quadrilaterals without clipping. If the quadrilateral is defined by four points $\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ (in CCW order), the customary bilinear mapping applies:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-v & v \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 \\ \mathbf{p}_3 & \mathbf{p}_2 \end{bmatrix} \begin{bmatrix} 1-u \\ u \end{bmatrix}$$

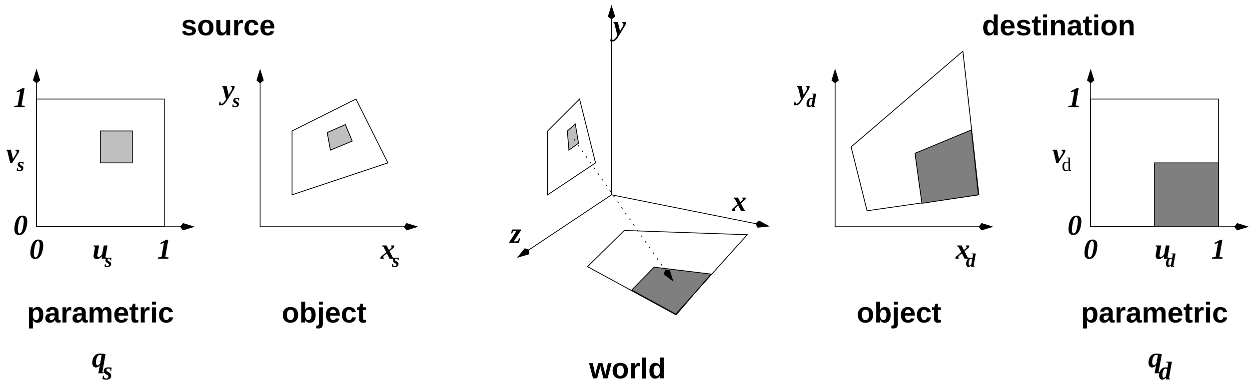

Figure 3: *Coordinate Systems. The gray quadrilaterals represent the support of source and destination bases.*

Unlike the perspective case, we can treat triangles as degenerate quadrilaterals, albeit with some irregular object space meshing. However, this mapping is not without its own drawbacks. While the parametric-to-object mapping is straightforward, the inverse object-to-parametric mapping is, in fact, double valued: a given (x,y) usually has two solutions (u,v), one of them inside the unit rectangle, one outside. In order to distinguish the two cases, we must clip in object space before inversion.

A more serious drawback is that straight lines are not, in general, preserved. The inverse projection of a straight line $ax+by+c=0$ into parametric space is, in general, a hyperbola. This also has implications for transport coefficient computation: it complicates determination of the limits of integration.

Again, while the still more complicated Jacobian of this transform makes the power flux computation of Equation (14) more difficult, it may still be done for choices of $B_{\mathbf{j}}$ with closed-form representations, such as splines.

7.1.2. Choice of Wavelet

Except where noted, our discussions in Section 6 did not depend on any particular choice of wavelet. For implementation purposes, we have to choose one. There are several good reasons for choosing Haar wavelets.

The fast wavelet transform can be performed in $O(N)$ time (see Beylkin /em et al.[1]), but if we allow for a varying dimensionality D and wavelet basis, it is easy to see from Equation (10) that the complexity is actually $O(W_h^D N)$ where W_h is the (varying) maximum width of the $\{h_j\}$ and $\{\tilde{h}_j\}$ (and, consequently, $\{g_j\}$ and $\{\tilde{g}_j\}$) coefficient sets.

For this reason, as the dimensionality increases, the rapidly-increasing operation count makes narrower filters more and more desirable, even though wider filters generally have better approximation properties. Since Haar is the narrowest possible wavelet filter ($W_h = 2$), it seems a wise strategy to make any multidimensional efforts first with Haar and move to wider bases later if Haar proves unsatisfactory.

An additional advantage of Haar wavelets over the others is the simplification of the calculation of the transport coefficients, irradiance and power flux. As Equations (19) and (20) have shown, these coefficients can be computed entirely in terms of pure smoothing calculations. A four-dimensional Haar pure smoothing basis is a function that is constant ($= 4^l$) within a hypercube and zero outside of it. The resulting transport coefficients are volume integrals of the overlap between such a hypercube in destination parametric space and the object which is a projection of a hypercube in source parametric space into the destination space. (This may not be such a great simplification. After all, any arbitrarily complex 3-dimensional integral may be trivially turned into a 4-dimensional volume integral!)

The overwhelming advantage of choosing Haar wavelets, however, is the great simplification they make in computing surface interaction. Because the Haar functions are piecewise constant, the individual integrals in the $\Delta_{\mathbf{jk}}^{\sigma\tau}$'s used in (27) become trivial. Upon incorporating these (conceptually) as sums of weighted Kronecker delta-functions into the evaluation of (22), it is possible to derive a very efficient algorithm to compute the surface interaction coefficients. See Lewis[16] for additional details.

7.1.3. Problems with Transport Coefficient Computation

As the preceding subsection suggests, using Haar wavelets turns transport coefficient computation into volume integral evaluation. There are two practical problems that complicate the computation of that volume.

7.1.3.1. Some Source Points Do Not Project Into Destination Space The source hypercube defines a range of positional and directional coordinates $\mathbf{q}_s$. Not all of these coordinates may map to points in the destination plane, much less the destination quadrilateral. This complicates any attempt at direct evaluation of the transport integrals.

7.1.3.2. The Projected Hypercube Has Curved Sides Even if all points in the source hypercube map to the destination plane, the nature of the resulting volume is not trivial. Needless to say, the projection of a source parametric hypercube into destination parametric space is not a hypercube. If, however, we could choose coordinate systems such that the source hypercube mapped to a polytope in destination space, we could take advantage of computational geometric techniques to first clip it against the destination hypercube and then compute the volume of the resulting polytope. Unfortunately, this is not possible because the source hypercube does not project to a polytope.

Consider the coordinate systems described in Section 7.1.1. Even if we choose rectilinear parametric $\leftrightarrow$ object mappings, the source object to destination object transform involves a projection. Hence, at best, the $q_s \to q_d$ mapping has a nonlinear dependence on the directional components.

As a result, the projection of the source hypercube into destination parametric space has curved sides. Furthermore, the curvature is such that we cannot guarantee that the convex hull of the polytope formed by projecting the 16 corners of the source hypercube into destination space contains the hypervolume. We have not fully pursued the possibility of an approximation of the projected hypercube by its convex hull here.

7.1.4. Integration Techniques

For these reasons, we must resort to multidimensional numerical integration schemes. Before considering candidate techniques, we first make an observation about the dimensionality required for numerical integration.

7.1.4.1. Reducing the Dimensionality As Equation (18) indicates, the computation of transport coefficients is intrinsically four-dimensional: two directional integrals and two positional integrals. If we take the two outermost integrals over direction and restrict our discussion to pure Haar smoothing components ($\nu_j = \nu_k = 0$, as Equation (21) permits), it is then evident that

$$T_{\mathbf{jk}} = 4^{l_j+l_k} \iint_{S_\mathbf{k}} G_\mathbf{j}(\kappa_d,\lambda_d)\, A_{\mathbf{jk}}(\kappa_d,\lambda_d)\, d\kappa_d\, d\lambda_d$$

$S_\mathbf{k}$ is the (square) directional support of $\tilde{B}_\mathbf{k}$. $G_\mathbf{j}(\kappa_d,\lambda_d)$ is a geometric function which is equal to one if a ray from the destination plane projected backwards along the (κ_d,λ_d) direction reaches the source plane and falls within the directional support of $B_\mathbf{j}$. Otherwise, it is zero. $A_{\mathbf{jk}}(\kappa_d,\lambda_d)$ is the area of the intersection of the spatial support of $B_\mathbf{k}$ in destination parametric space with the projection (in the κ_d,λ_d direction) to destination parametric space of the spatial support of $B_\mathbf{j}$.

We are now in a position to evaluate the coordinate mappings given in Section 7.1.1 to see which of them makes $A_{\mathbf{jk}}(\kappa_d,\lambda_d)$ easy to compute. All the mappings transform lines of constant u or v to lines in object space, so any of them would work for the source parametric to source object mapping. Only rectilinear and perspective mappings, however, transform arbitrary lines in object space to lines in parametric space. If we choose either of them for our destination object to destination parametric mappings, computation of $A_{\mathbf{jk}}(\kappa_d,\lambda_d)$ amounts to clipping the projected quadrilateral to the spatial support of $B_\mathbf{k}$ using a conventional polygon clipping algorithm and computing the resulting area. This allows us to reduce the dimensionality that we need to integrate numerically from four to two.

7.1.4.2. Numerical Quadrature Regardless of the number of dimensions, numerical integration techniques are all based on some form of quadrature:

$$\int f(\mathbf{x})d\mathbf{x} \approx \sum_{i=1}^{N_{samp}} w_i f(\mathbf{x}_i)$$

where N_{samp} is the number of samples. Techniques differ principally in their choices of weights w_i and sample points $\mathbf{x}_i$. Zwillinger[21] provides an extensive survey of these. The ones we have chosen to evaluate are:

- *trapezoidal*: a regular grid approximating f linearly in each dimension
- *Romberg*: a multilevel (Richardson) extrapolation (in terms of the grid spacing) of trapezoidal results
- *Monte Carlo*: $\mathbf{x}_i$'s chosen from a (possibly stratified) pseudo-random sequence (Glassner[9], p. 310 is a good overview of pseudo- and quasi-random integration methods in image synthesis.)
- *Halton*: similar to Monte Carlo, but using *quasi-random* numbers generated according to number theoretical considerations
- *Hammersley*: an alternative quasi-random method

While pseudo-random and quasi-random techniques are generally preferred for multidimensional quadrature, we include trapezoidal and Romberg techniques to explore the two-dimensional case, where Monte Carlo has error of order $O(N_{\text{samp}}^{-1/2})$ (This is true regardless of the dimensionality of the problem and is one of the appeals of Monte Carlo integration.) and the trapezoidal rule has error of order $O(N_{\text{samp}}^{-3/2})$. We must be careful to use these error estimates cautiously, however, since our integrand, $G(\kappa_d,\lambda_d)A_{\mathbf{jk}}(\kappa_d,\lambda_d)$ contains discontinuities that error analyses do not account for.

7.1.4.3. Comparison To evaluate the accuracies of the various methods, we have applied them to a test problem: computing the set of all $T_{\mathbf{jk}}$ for a given $\mathbf{j}$ and geometry. To an-

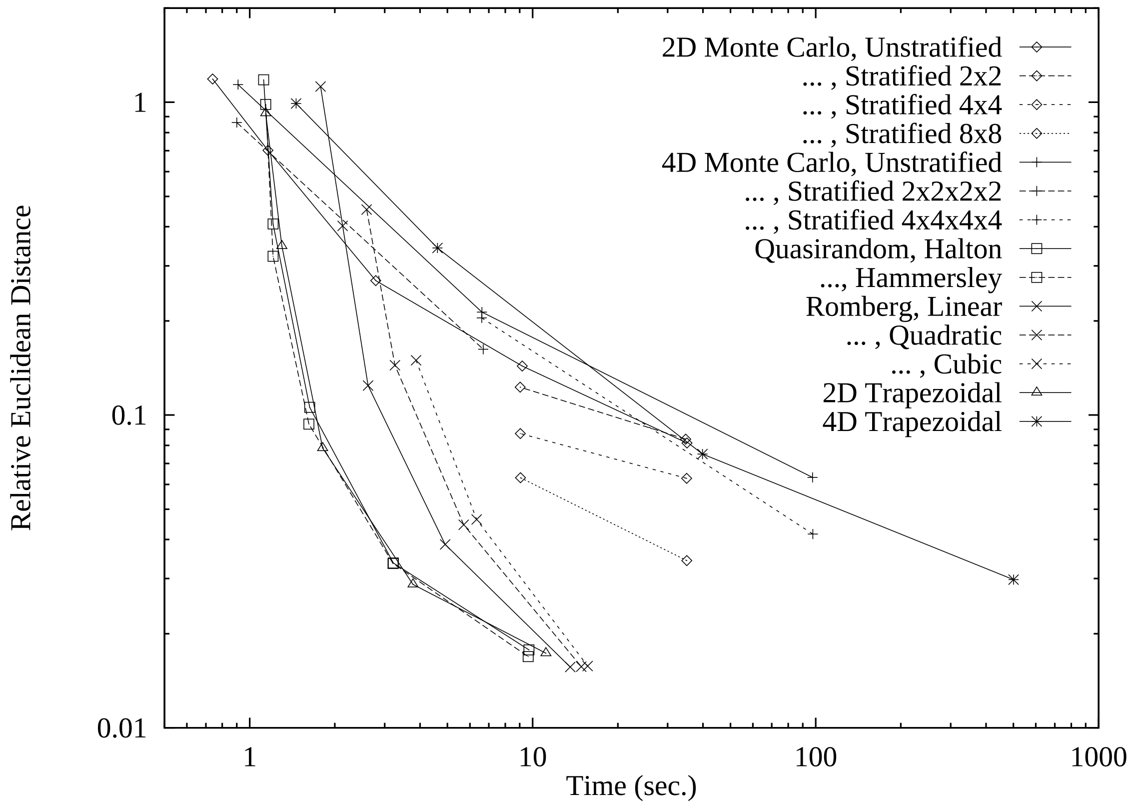

Figure 4: *Relative Euclidean Distance vs. Time for Different Integration Techniques and Numbers of Samples*

alyze the results we can treat this set as a vector in a K-dimensional space, where $K = 16^{l_{\max}+1}$ and $l_{\max}$ is the maximum level we have coefficients for. (I.e., we have K values of $\mathbf{k}$.)

To compare results against a reference, we create a reference vector $T_{\mathbf{jk}}^{\text{ref}}$ using an extremely long (25 CPU-minutes) integration time and then adopt two metrics. The first, which we call the *relative Euclidean distance* (hereafter, RED) is the Euclidean distance (L^2 norm) of $T_{\mathbf{jk}}$ from $T_{\mathbf{jk}}^{\text{ref}}$ divided by the magnitude of $T_{\mathbf{jk}}^{\text{ref}}$:

$$\frac{\sqrt{\sum_{\mathbf{k}} \left(T_{\mathbf{jk}} - T_{\mathbf{jk}}^{\text{ref}}\right)^2}}{\sqrt{\sum_{\mathbf{k}} \left(T_{\mathbf{jk}}^{\text{ref}}\right)^2}}.$$

Figure 4 shows how the RED varies with integration time and N_{samp} for the various techniques. Rather than N_{samp}, time is the appropriate abscissa here. Since the time required to evaluate the integrand varies with dimensionality, fewer samples do not necessarily imply faster computation. Note that we allow the degree of extrapolation for Romberg integration to vary from linear to quadratic to cubic. The times shown are for an IBM Model RS/6000 POWERserver 560 workstation.

The second metric, which we call *relative maximum departure* (hereafter, RMD), is more conservative. It is the maximum absolute difference between $T_{\mathbf{jk}}$ and $T_{\mathbf{jk}}^{\text{ref}}$ (L_∞ norm) divided by the maximum absolute value of $T_{\mathbf{jk}}^{\text{ref}}$:

$$\frac{\max_{\mathbf{k}} \left|T_{\mathbf{jk}} - T_{\mathbf{jk}}^{\text{ref}}\right|}{\max_{\mathbf{k}} \left|T_{\mathbf{jk}}^{\text{ref}}\right|}.$$

Figure 5 shows how the RMD varies for the same parameters as Figure 4.

From these plots, we can draw several conclusions:

- The two figures are qualitatively similar: a method that does well by one metric generally does well by the other. This gives us some confidence that these metrics are valid.
- Stratification improves Monte Carlo results. (This comes as no surprise.)
- The time required for Monte Carlo methods being linear in N_{samp}, they all approximate the expected $O(N_{\text{samp}}^{-1/2})$ behaviour.

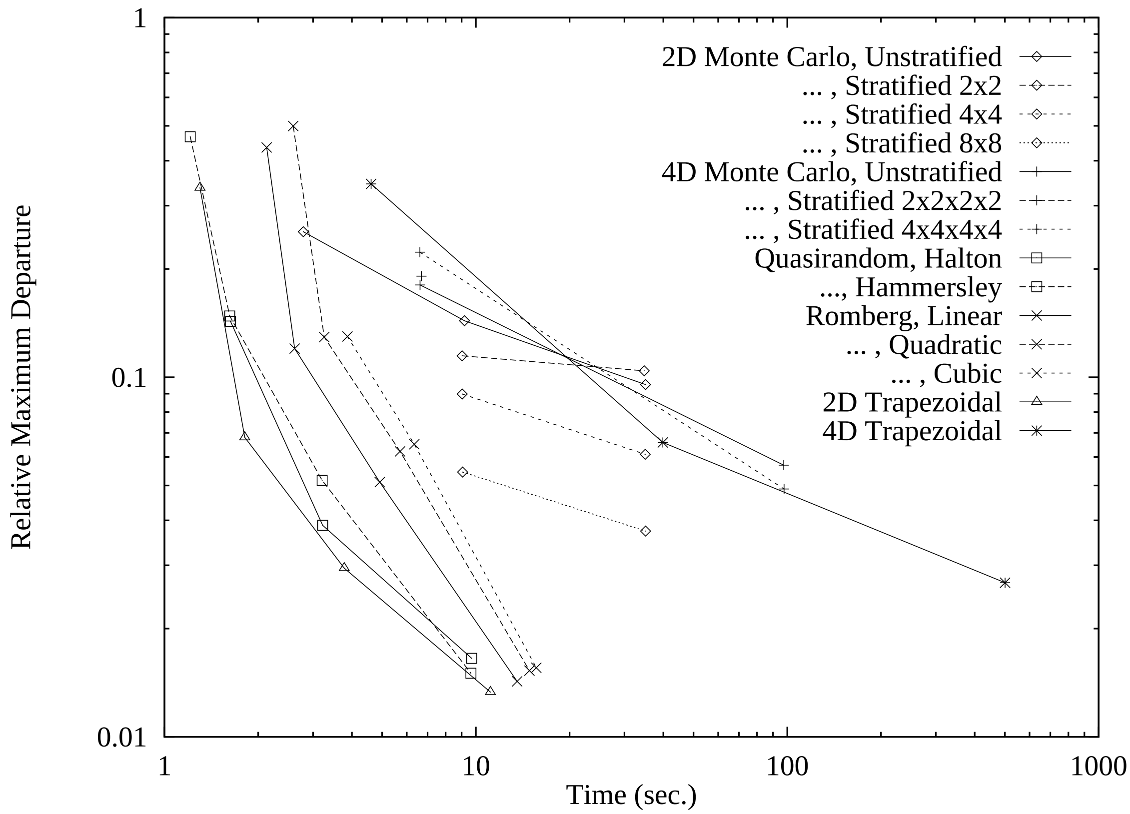

Figure 5: *Relative Maximum Departure vs. Time for Different Integration Techniques and Numbers of Samples*

- For a given integration time, the 4D trapezoidal method does generally worse than the other methods.
- Increasing the degree of extrapolation for Romberg (2D) integration generally makes matters worse. This is presumably a result of the discontinuities in the integrand we discussed above.
- Quasi-random methods give similar and comparatively good results. This reinforces the findings of Keller[13].

The most surprising observation, however, is how well the straightforward 2D trapezoidal rule does. For the RED metric, it is comparable with the best of the other methods, quasi-random, and for the RMD metric it does noticeably better for short integration times.

7.2. Minor Implementation Details

In this subsection, we will discuss design decisions particular to our own WRT system design.

This implementation takes up approximately 30,000 lines of C code. We therefore emphasize that the classes and pseudocode we present here are in most cases simplified for clarity.

In what follows, as in our code, we have followed consistent and fairly obvious naming conventions that we feel have improved reliability and flexibility and allowed many of the benefits of object-oriented programming while developing in a highly portable environment. (Indeed, the code moves between IBM AIX, SGI IRIX, and Intel (RedHat) Linux with fewer than 50 system-dependent lines of code, mostly due to differing system header files.)

These conventions are an adapted form of what are known as “Hungarian” conventions, partly in deference to the nationality of their chief developer, Charles Simonyi[20]. (See McConnell[17] for an additional discussion of Hungarian naming, particularly as practiced at Microsoft.),

7.2.1. The *WaveletIndex* Class

Figure 6 shows the wavelet index class (i.e., `typedef`) we have adopted for our implementation. The names and sequence of the components are consistent with Equation 9.

For the time being, we have implemented all of these fields as `short` (16 bit) unsigned integers. This allows us to go as far as level *wi.l = 16* without overflowing the offsets. Experience suggests that the maximum level we will use will be much less than 16. We could achieve a minor reduction in memory usage by using `unsigned char` variables for

```
typedef struct {
    short int nu;
    short int l;
    short int m[4];
} WaveletIndex;
```

Figure 6: *The* WaveletIndex *Class*

wi.nu (but not *wi.l*, since it needs to represent levels from 0 to the maximum level inclusively).

7.2.2. Storing Wavelet Coefficients

Because wavelet coefficients are hierarchical in nature, we refer to the data structures we devise here to store them as “wavelet coefficient trees” (hereafter, “WCT”s).

Representing a WCT with a maximum level of resolution $l_{\max}$ requires $16^{l_{\max}+1}$ possible wavelet coefficients per channel. Clearly, compression is called for. The (bi)orthogonality of wavelets presents an L^2-optimal compression strategy — thresholding low-magnitude coefficients. In this case, the data was single-channel and single dimensional (i.e., each wavelet coefficient’s support was unique), but the results are independent of the wavelet dimensionality.

We not only want to guarantee a good approximation of the data with the compressed coefficients, but also efficient use of storage and fast reconstruction.

7.2.2.1. Hashing Coefficients Given a sparse set of wavelet radiance coefficients $\{b_{\mathbf{k}}\}$, we need to store them in a way that facilitates the mapping of the wavelet index $\mathbf{k} \rightarrow b_{\mathbf{k}}$ needed to perform the transport operation Equation (17). The obvious way to do this is with a hash table. Not knowing the set of destination indices $\{\mathbf{k}\}$ in advance prohibits perfect hashing, so it is necessary that the hashing scheme allows for collisions and that the hash table entries contain $\mathbf{k}$ as well as $b_{\mathbf{k}}$ values.

7.2.2.2. Multichannel Grouping As we mentioned in Section 2, we have been treating data monochromatically throughout this paper. In practical applications, however, we must evaluate Equation (17) for all three (or however many) channels. Having assumed a non-participating medium, the transport coefficients $T_{\mathbf{jk}}$ are achromatic. Evaluation of Equation (17) simply means multiplying each element of a (now) 3-vector $b^s_{\mathbf{j}}$ by the same value of $T_{\mathbf{jk}}$ and accumulating the result in another 3-vector $b^d_{\mathbf{k}}$.

In the absence of the need for compression, it would be convenient to group all channels of $b^s_{\mathbf{j}}$ into a single group, rather than create a separate representation for each channel. In the presence of compression, however, each channel has its own threshold. If one component is above its threshold but the other two are below theirs, saving the latter is a waste of storage. This would be mitigated, however, if there were a high degree of correlation between the magnitudes of the coefficients. Certainly, a wavelet representation of white light given off by a luminaire displays a high degree of correlation. When this light is reflected off a surface of varying spectral reflectivity, however, that correlation will be diminished. By how much depends on the nature of the surface.

```
typedef struct {
    int iWnOfHash[int mxnIWnOfHash];
    struct WaveletNode {
        WaveletIndex wi;
        unsigned short mskChild;
        unsigned short mskNu;
        int iWclFirst;
    } wnBase[int nWn];
    struct WaveletCoefficientList {
        float b[3];
        int iWclNext;
    } wclBase[int nWcl];
    double umrad;
} WaveletCoefficientTree;
```

Figure 7: *The* WaveletCoefficientTree *class*

We have two choices here: to group or not to group multichannel data. For the time being, we have chosen the former — believing that there is sufficient correlation and advantage in retrieval speed in most situations to justify grouping. This definitely requires further study.

7.2.2.3. Hashing Nodes Instead of Coefficients In one dimension, a node in the wavelet pyramid contains a single wavelet coefficient. In four dimensions, such a node has one pure wavelet ($\nu = 15$) coefficient and fourteen mixed wavelet/smoothing ($\nu \in \{1 \ldots 14\}$) coefficients. (Recall that the pure smoothing ($\nu = 0$) coefficient may be reconstructed from the node’s ancestors, if any, and only needs to be kept at the root node.) All coefficients at a given node correspond to basis functions with the same (Haar) or similar (other bases) support.

We might therefore expect to find a higher degree of correlation in magnitude between coefficients that belong to the same node and coefficients that do not. This suggests that we can reduce the index storage overhead by hashing entire nodes rather than individual coefficients.

7.2.2.4. The *WaveletCoefficientTree* Class Figure 7 shows the fundamental structure we use to represent wavelet coefficients: *WaveletCoefficientTree* (hereafter, WCT). Note that the code in the figure is not legal C code: We have combined several subsidiary classes and moved some component definitions around to indicate dynamically-sized structures.

Given a WCT *wct*, *wct.iWnOfHash[]* is the hash table itself. It is indexed by the hash of a (pure smoothing) wavelet

```
typedef struct {
    Transform3d tf3dSobjWld;
    Transform3d tf3dDobjWld;
    Transform3d tf3dSobjDobj;
    Transform3d tf3dR;
    Polygon *pgnSobj;
    Transform2d tf2dSparToSobj;
    Transform2d tf2dSobjToSpar;
    Polygon *pgnDobj;
    Transform2d tf2dDparToDobj;
    Transform2d tf2dDobjToDpar;
} TransportGeometry;
```

10

Figure 8: *The* TransportGeometry *Class*

index. Its entries are indices into *wct.wnBase[]*, the array of wavelet nodes. Note that the use of integer indices rather than pointers allows the dynamic resizing of *wct.wnBase[]*.

Each node contains the corresponding wavelet index. This is not only useful for hash collision detection, but by indexing *wct.wnBase[]*, we can traverse all nonzero entries in *wct* directly. *wct.wnBase[i].mskChild* is a 16-bit mask indicating which of node *i*'s 16 children are also present in *wct*. *wct.wnBase[i].mskNu* is a 16-bit mask indicating which basis selectors are present in *wct* for node *i*.

The array *wct.wnBase[i].wclBase[]* contains all wavelet coefficients. For a given wavelet node *wct.wnBase[i]*, *wct.wnBase[i].iWclFirst* is the index into that array of the first element in the list of coefficients belonging to the node. Successive elements are ordered in increasing basis selector value, which is derived from *wct.wnBase[i].mskNu*. Each element *wct.wnBase[i].wclBase[j]* contains the index *wct.wnBase[i].wclBase[j].iWclNext* of the next-higher set of multichannel coefficients belonging to node *i*. (The last element in the list has this index set to -1.)

7.2.3. The *TransportGeometry* Class

The *TransportGeometry* class shown in Figure 8 contains all geometric information necessary to compute wavelet radiative transfer between source and destination polygonal objects. Given a *TransportGeometry* object *tg*, *tg.tfSobjWld* is the affine 3D transform from source object to world positional coordinates, *tg.tfWldDobj* is the affine 3D transform from world to destination object positional coordinates, *tg.tfR* is the orthogonal 3D rotational transform from source to destination directional coordinates, *tg.pgnSobj* and *tg.pgnDobj* are the source and destination polygons (in object coordinates), *tg.tf2dSparToSobj* is the affine 2D transform from source parametric to source object coordinates, and *tg.tf2dDobjToDpar* is the affine 2D transform from source parametric to source object coordinates. The implemented *TransportGeometry* class contains inverses of most of these transforms as well, since some of the integration

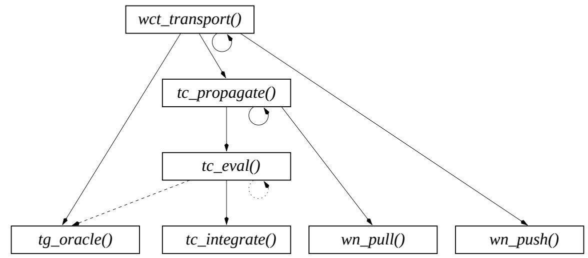

Figure 9: *Simplified Functional Decomposition for Wavelet Radiative Transfer*

schemes we use actually project destination points back into source space.

7.2.4. Functional Decomposition of *WRT*

Figure 9 shows a simplified functional decomposition for *WRT*. In this subsection, we will discuss each block individually and describe how it implements the ideas presented in Section 6. We will proceed in a bottom-up sequence.

7.2.4.1. *tg_oracle()* This function is a geometric query function. Given a *TransportGeometry* and source and destination *WaveletIndices*, it performs several fast query functions.

- If either source or destination wavelet directional supports lie entirely outside the UDC, they cannot interact.
- Projecting the 16 vertices of the destination support hypercube onto the plane that contains the source, the vertices can be classified by the “fast reject” part of a four-dimensional version of the Cohen-Sutherland line clipping algorithm (see Foley, et al.[6], p. 113, for example).
- If all 16 destination vertices map into points contained within the source support, we assume that the entirely of the destination support does, so an exact computation of the inner product (which is then proportional to the volume of the destination support hypercube) is possible.

The latter two items require an assumption that the projected destination support vertices define a convex hull for the entire projected support. As we noted in Section 7.1.3.2, this is not in general the case. Nevertheless, by disabling this oracle and comparing results, we have found empirically that it is a reasonable approximation, at least for oracular purposes.

7.2.4.2. *tc_integrate()* This function is responsible for performing the actual integration. It is capable of using any of the schemes described in Section 7.1.4.

7.2.4.3. *wn_pull()* and *wn_push()* These functions act on individual nodes of a WCT to perform, respectively, a single-level, four-dimensional Haar analysis (i.e., forward trans-

```
tc_propagate(WaveletIndex wiS, double bS[],
            WaveletIndex wiD,
            TransportGeometry tgSD,
            WaveletCoefficientTree wctD)
    if wiD.l == maximum wavelet level + 1,
        return tc_eval(wiS, wiD, tgSD)
    else if tg_oracle(wiS, wiD, tgSD) gives
            exact result,
        return it
    else
        for each child wiDChild of wiD,
            tc[wiDChild] = tc_propagate(wiS,
                    bS, wiDChild, tgSD, wctD)
        for each channel chan,
            for each child wiDChild of wiD,
                wn[chan][wiDChild] = bS[chan]
                        ∗ tc[wiDChild]
            wn_pull(wn)
            if wiD.l is not the top destination
                    level,
                wn[chan][wiD.nu = 0] = 0
        wctD[wiD] = wn
```

Figure 10: tc_propagate() *Pseudocode*

form or "pull" — cf. Hanrahan *et al.*[12]) or synthesis (i.e., inverse transform or "push").

7.2.4.4. *tc_eval()* Given source and destination wavelet indices, this function computes a single transport coefficient. It will call itself recursively if either source or destination basis selectors are nonzero, although as it is currently used in *WRT*, *tc_eval()* is only called to evaluate pure smoothing coefficients. (The ability to work with non-zero basis selectors is used in self-test mode.) At the user's request, it will call *tg_oracle()* to attempt to find an alternative to numerical quadrature. Otherwise, it will call *tg_integrate()* to perform that quadrature.

7.2.4.5. *tc_propagate()* This function propagates a single source wavelet coefficient to a destination wavelet coefficient tree *wctD*. Figure 10 shows its pseudocode. Like *tc_eval()*, it will work with a nonzero source basis selector, but in the context of *WRT* is only called upon with pure source smoothing coefficients. It is important to note that *wctD* will contain no pure smoothing coefficients except at a specified top level. The source pure smoothing coefficients corresponding to *wiS* are passed to the function as *b0[]*.

There are three alternatives:

- If the destination wavelet is one more than the maximum wavelet level, it returns the smoothing coefficient it gets from *tc_eval()*.
- If *tg_oracle()* gives an exact result (often zero), it returns that result. Note that *wctD* does not need to be updated in this case, since the exact result is either zero or a pure smoothing result below the top level, neither of which requires storage.
- Otherwise (and most importantly) the function applies itself recursively to the 16 child indices of *wiD*, collecting the 16 returned smoothing coefficients. For each channel, it then multiplies each coefficient by the source coefficient for that channel, and uses *wn_pull()* to apply a "pull" to convert the result to wavelet coefficients. If *wiD.l* is not the top destination level, the pure smoothing coefficient is redundant and is set to zero. The result is then ready to be stored in *wctD*.

```
wct_transport(WaveletIndex wiS, double bS0[],
            WaveletCoefficientTree wctS,
            TransportGeometry tgSD,
            WaveletIndex wiDTop,
            WaveletCoefficientTree wctD):
    if tg_oracle(wiS, tgSD, wiDTop) says an
            interaction is not possible,
        return
    wn = wctS[wiS]
    if wn exists,
        for each channel chan,
            wn[chan][0] = wn[chan][0] + bS0[chan]
        wn_push(wn)
        for each child wiSChild of wiS,
            wct_transport(wiSChild, wn[∗][wiSChild],
                    wctS, tgSD, wiDTop, wctD)
    else
        tc_propagate(wiS, bS0, wiDTop, tgSD, wctD)
```

Figure 11: wct_transport() *Pseudocode*

7.2.4.6. *wct_transport()* Figure 11 shows the pseudocode for this function, which transports an entire source wavelet coefficient tree *wctS* to a destination wavelet coefficient tree *wctD*.

The function first uses *tg_oracle()* to reject impossible interactions. If there is a node *wn*, indexed by *wiS* in *wctS*, it will call itself recursively to descend the tree. First, though, to the pure smoothing coefficients of *wn*, it adds *bS0[]* (which is zero when this function is initially called before recursion). It then invokes *wn_push()* to convert the wavelet coefficients to finer pure smoothing coefficients at the level of the children of *wiS*. *wct_transport()* then applies itself recursively to each of these children, passing the elements of *wn* as new values of *bS0[]*.

If *wn* does not exist in *wctS*, there are no nodes in *wctS* at or below *wiS*, so the only thing that needs to be propagated is the pure smoothing value *bS0[]*. *wct_transport()* calls *tc_propagate()* to do this. The change of prefix is an indication that below this call level, we are no longer concerned with a source WCT, but with individual pure smoothing (multichannel) coefficient vectors.

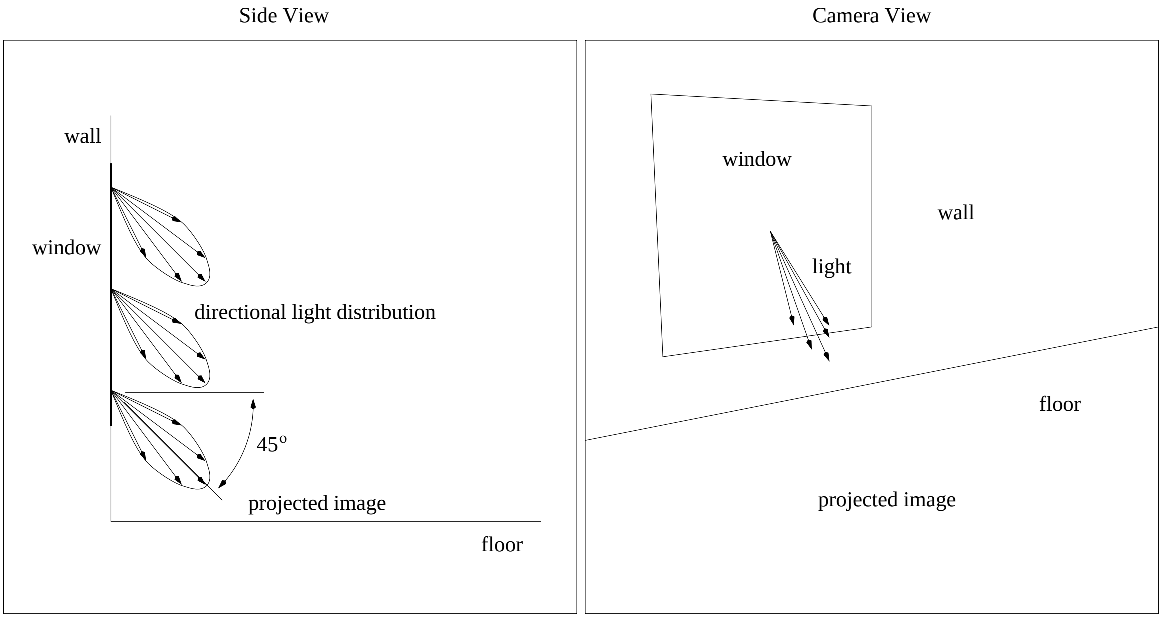

Figure 12: *Geometry Used for Example of Transport*

7.3. Example of Transport

For a test configuration, we imagine light shining through the square stained glass window shown in Figure 13.

As illustrated in Figure 12, the incident light shines down with a distribution peaking at an angle of 45° from the horizontal, diffused by the glass according to a distribution proportional to the 4th power of the cosine of the angle between the propagation direction and the peak direction. The light is transported to the floor and is collected there.

Figure 14 shows the complete, inversely-transformed 4D results. Each of the small images represents the spatial variation of radiance in the fixed direction given by the image's position in the matrix.

Figure 15 is a detailed view of the brightest part of Figure 14. This computation of $32 \times 32 \times 32 \times 32$ coefficients (in red, green, and blue channels) compressed by 95% required 12.4 hours of CPU time on an IBM RS/6000 POWERserver 560 workstation.

Needless to say, this is impractical for a single frame, but the result is reusable.

Figure 16 shows several frames generated with an otherwise conventional raytracer modified to treat a wavelet radiance distribution as a "4-D texture". Each frame is generated from a different camera position. Most of the "floor" is covered with 16 tiles with alternating mostly-diffuse ("lighter") and mostly-specular ("darker") Phong illumination models in a checkerboard pattern. The resulting images contain none of the "noise" common to the usual Monte Carlo approach to this sort of problem.

8. Work in Progress

We are continuing work with improved transport coefficient integration techniques and plan to take further advantage of wavelet representations, such as knowledge of the destination's reflective properties, to reduce the amount of computation required and allow us to go to larger sets of coefficients in less time.

We also hope to extend this work to wavelets with better representational properties than Haar wavelets and to develop more rapid reconstruction algorithms.

References

1. G. Beylkin, Ronald R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics*, 44:141–183, 1991.

2. Per Henrick 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.

3. Per Henrik 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.

4. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, San Diego, CA, 1993.

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

6. James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1990.

7. Alain Fournier. From Local to Global Illumination and Back. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 127–136, New York, NY, 1995. Springer-Verlag.

8. Alain Fournier, Eugene Fiume, Marc Ouellette, and Chuan K. Chee. Fiat lux. Technical Report 90-1, Dynamic Graphics Project, University of Toronto, January 1990.

9. Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.

10. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 43–54, 1996.

11. 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.

12. 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.

13. Alexander Keller. Quasi-monte carlo radiosity. In *Seventh Eurographics Workshop on Rendering*, Porto, Portugal, June 1996.

14. Paul Lalonde and Alain Fournier. Filtered local shading in the wavelet domain. In Julie Dorsey and Phillipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 163–174, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

15. Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4-9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 31–42. ACM SIGGRAPH, ACM Press, August 1996.

16. Robert R. Lewis. Light-driven global illumination with a wavelet representation of light transport. doctoral thesis, Department of Computer Science, University of British Columbia, 1998.

17. Steve McConnell. *Code Complete*. Microsoft Press, 1993.

18. W. Nusselt. Graphische Bestimmung des Winkelverhältnisses bei der Wärme-Strahlung. *Zeitschrift des Vereines Deutscher Ingenieure*, 72(20):673, 1928.

19. 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.

20. Charles Simonyi. Meta-programming: A software production method. Technical Report CSL-76-7, Xerox Palo Alto Research Center, 1976.

21. Daniel Zwillinger. *Handbook of Integration*. Jones and Bartlett Publishers, Inc., Boston, MA, 1992.

Figure 13: *A Stained Glass Window. This is the spatial component of radiance for the test configuration.*

Figure 15: *Detailed View of the Brightest Part of Figure 14*

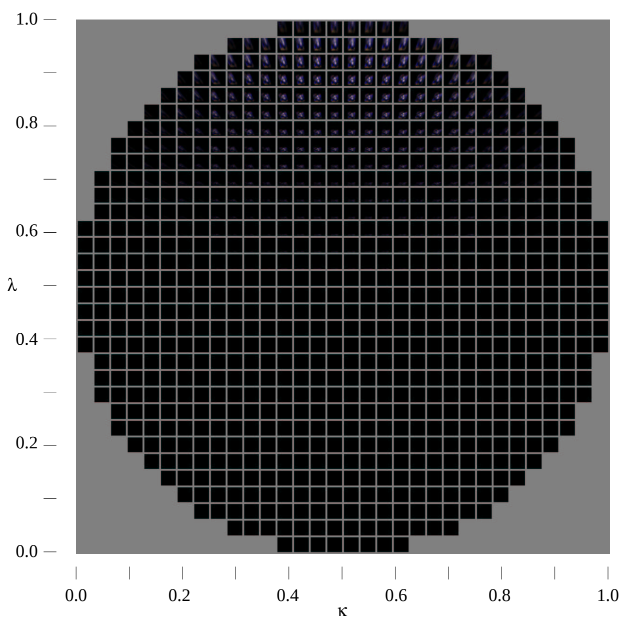

Figure 14: *Four-Dimensional Results of Wavelet Radiative Transport*

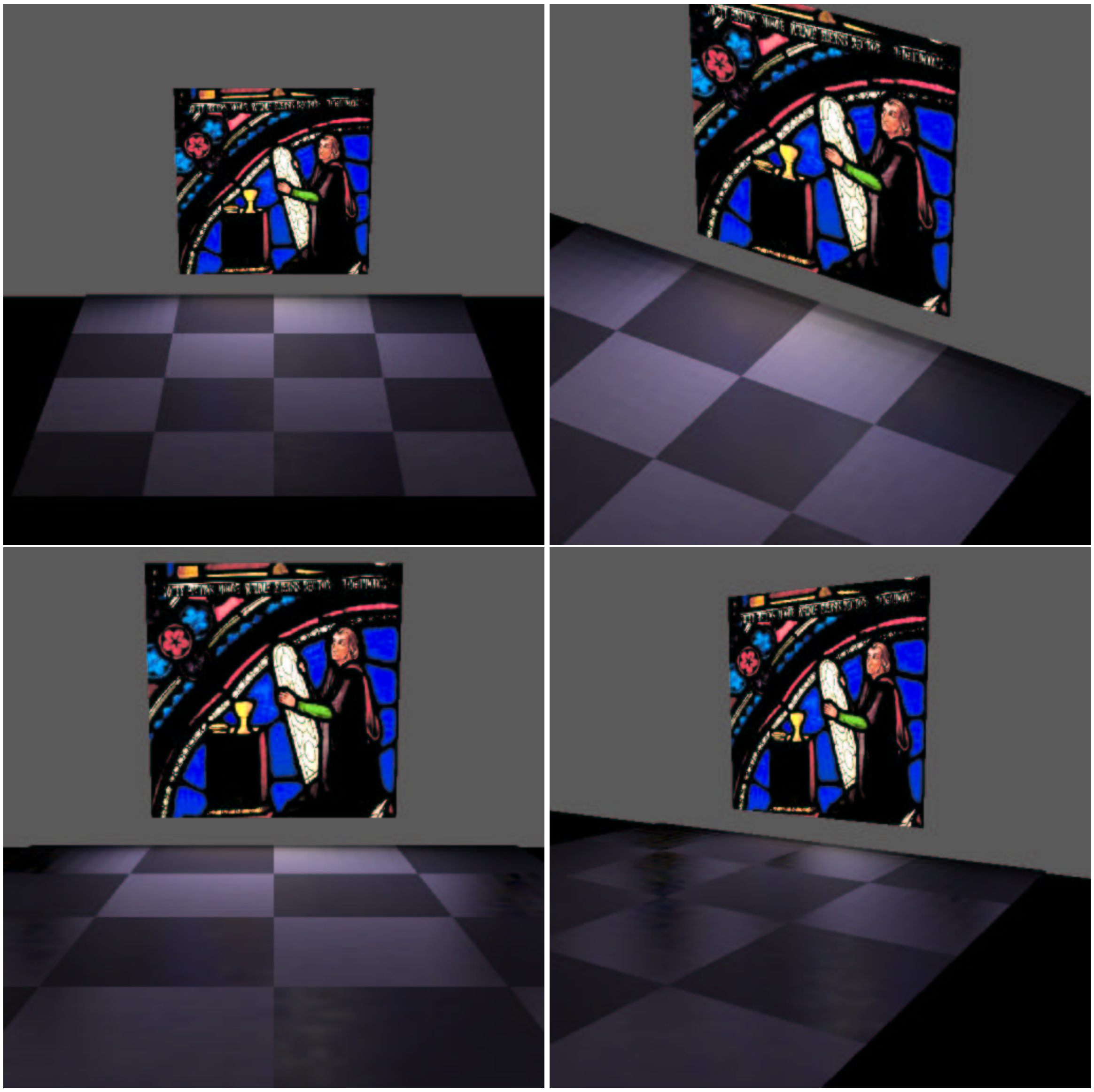

Figure 16: *Example of Wavelet Transport. Reflected wavelet representations of radiance as shown in Figure 14 can be sampled from different directions.*