Cours avancé de synthèse d'image

Illumination Globale : Théorie et Pratique

Kadi BOUATOUCH, Sumanta N. PATTANAIK et Pierre TELLIER
IRISA/IFSIC, Rennes, France

Contents

Chapter 1

Introduction

In the recent literature, significant progresses have been made in the field of rendering techniques through the use of improved global illumination model. A global illumination model describes the light transport mechanism between surfaces, that is, the way each surface element interacts with the others. Therefore, the global illumination model is a key problem when accuracy is needed in the rendering process (photorealism or photosimulation). So far, two approaches have been proposed to solve this problem : the former is based on ray tracing and the latter is based on the so-called radiosity solution. As these two approaches are complementary, several authors have tried to deal efficiently with all the light transport mechanisms by mixing both approaches in their model.

To attain realism in computer graphics, two main attempts have been adopted. The first one make use of empirical and ad-hoc illumination models, while the other makes use of the fundamental physical laws governing the interaction of light with materials and participating media, and of the characteristics of the human visual system, in order to produce images which are exact representations of the real world. This tutorial deals with this second approach, and shows how the real aspects of materials and the real simulation of global lighting can be simulated only with physics-based reflection and transmission models, and with a spectral representation of the emitted, reflected and refracted light powers.

The goal of this tutorial is threefold. First, it shows how a global illumination model can be derived from physics, optics, and photometry. Second, It provides a collection of information on colorimetry, visual perception and visualisation aiming at a full understanding. Third, different implementations of global illumination model are presented: radiosity, one-pass and two pass-methods.

Chapter 2

Photometry

2.1 Light

Light can be considered as a mixture of electromagnetic waves which propagate at the same velocity called *light velocity*. Each of these waves has a frequency, a period and a wavelength λ. Each wave of a light, called from now on spectral component, has an energy proportional to the the square of its amplitude. A radiation is no more than an emission or transport of light energy through a medium. In addition, each spectral component of light seen in isolation has a characteristic color appearance. Long wavelengths of visible light, say around $700nm$, generally appear red. Short wavelength components, say around $400nm$, generally appear blue. These facts explain why Newton observed a rainbow.

Because the wavelength decomposition is fundamental, we can describe any light completely by the amount of power (energy per unit time) in each of its spectral wavelength components. This description consists of a fairly long series of values called *the spectral power distribution* of the light. The precise number of values necessary to describe the light depends on how finely the wavelength samples are spaced. Typical wavelength spacing ranges between $1nm$ and $10nm$, depending on the goals of the specification and how rapidly the power may vary with wavelength. The human visual system is only sensitive to the wavelengths of light ranging between $380nm$ and $780nm$. If we space the wavelength samples at $10nm$, 31 values are required. Later on, we will see that 4 or 10 not equally spaced values are sufficient to precisely produce colored synthetic images.

A another fundamental fact about light is that the spectral power distribution of the mixture of two lights is the sum of the spectral power distributions of the individual lights.

A light power Φ and its spectral distribution $\Phi(\lambda)$ are related by the following formula:

$$\Phi = \int_{380nm}^{780nm} \Phi(\lambda) d\lambda.$$

Note that, to an energetic power Φ_e (expressed in watts) of spectral distribution $\Phi_e(\lambda)$ corresponds a luminous power (expressed in lumens) having $\Phi_v(\lambda)$ as spectral distribution. The relationship between these quantities are:

$$\begin{aligned} \Phi_e &= \int_0^\infty \Phi_e(\lambda) d\lambda \\ \Phi_v(\lambda) &= 680 V(\lambda) \Phi_e(\lambda) \\ \Phi_v &= \int_0^\infty \Phi_v(\lambda) d\lambda, \end{aligned}$$

where $V(\lambda)$ is the sensitivity function of the human eye. This function is null out of the range $[380nm, 780nm]$.

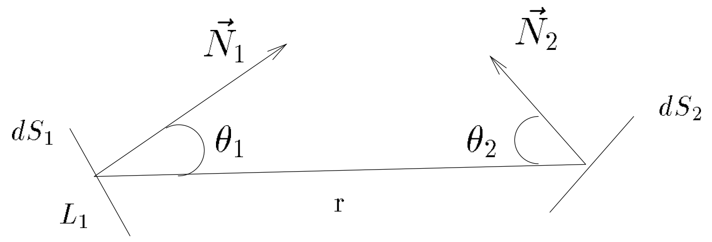

Figure 2.1: Geometry for radiometric quantities

In the following, the expressions of all radiometric quantities are valid for light powers (Φ) as well as for each spectral component ($\Phi(\lambda)$).

2.2 Important radiometric quantities

We recall herewith the expressions of the radiometric quantities used for the determination of a global illumination model (see figure 2.1).

- **light power or flux**: is the energy leaving a surface or impinging onto a surface per unit time.
- **radiant intensity**: is the flux leaving a surface per unit solid angle:

$$I = \frac{d\Phi}{d\Omega_1}$$

- **radiance**: is the flux leaving a surface per unit projected surface and per unit solid angle. Therefore, the flux transmitted from surface dS_1 to surface dS_2 is

$$\begin{aligned} d^2\Phi &= L_1 \cos\theta_1 dS_1 d\Omega_1 \\ &= L_1 \cos\theta_1 \cos\theta_2 \frac{dS_1 dS_2}{r^2} \end{aligned}$$

  where

$$d\Omega_1 = \frac{\cos\theta_2 dS_2}{r^2}$$

  and L_1 is the radiance of surface dS_1. Radiance is sometime improperly called intensity but this latter will be avoided here because it could be confused with radiant intensity which is the radiant power per unit solid angle. It is important to note that the human eye is only sensitive to radiance. Therefore, a global illumination model must account for the computation of radiance of surfaces seen by the observer. In general, radiance is direction dependent.

- **radiant exitance** : is also called radiant emittance or radiosity. It represents the light power leaving a surface, per unit area and is given by

$$B = \frac{d^2\Phi}{dS_1} = L_1 \cos\theta_1 d\Omega_1$$

  .

- **irradiance**: is the light power, per unit area, impinging onto a surface. It is expressed as

$$A = \frac{d^2\Phi}{dS_2} = L_1 \cos\theta_2 d\Omega_2 \tag{2.1}$$

where

$$d\Omega_2 = \frac{dS_1 \cos\theta_1}{r^2}.$$

The following table recalls all these radiometric quantities.

	energetic quantity	luminous quantity
Flux	$\phi_e(\lambda)$ $(Watt)$	$\phi_v(\lambda)$ $(lumen)$
Radiant intensity	$I_e(\lambda) = \frac{d\phi_e(\lambda)}{d\Omega}$ $(Watt.strd^{-1})$	I_v $(candela)$
Irradiance or Exitance	$E_e(\lambda) = \frac{d\phi_e(\lambda)}{dA}$ $(Watt.m^{-2})$	E_v (lux)
Radiance	$L_e(\lambda) = \frac{d\phi_e(\lambda)}{d\omega dS \cos\alpha}$ $(W.strd^{-1}.m^{-2})$	L_v $(candela.m^{-2})$

Chapter 3

Reflection and Transmission

3.1 Definitions

A lot of quantities can be found in the radiometry theory to characterize reflectivity but only two of them will be used here : the bidirectional reflectance and the coefficient of reflection. As for transmission, it is characterized by the bidirectional transmittance. This coefficient of reflection or reflectance is simply the ratio of the reflected power by the incident power for a small surface element. It is given by

$$\rho = \frac{d^2\Phi_r}{d^2\Phi_i} = \frac{B_r}{A_i} = \frac{radiosity}{irradiance}$$

For defining the bidirectional reflectance, let us consider a small surface receiving light power contained in a small solid angle $d\Omega_2$ about a given direction of incidence. This light is reflected in all directions but let us consider the radiance dL_r in a direction of reflection Θ_r (figure 3.1). Then, the bidirectional reflectance is the ratio

$$\mathbf{f_r}(\Theta_i, \Theta_r) = \frac{dL_r}{dA} = \frac{radiance}{irradiance}.$$

Or

$$\mathbf{f_r}(\Theta_i, \Theta_r) = \frac{dL_r(\Theta_r)dS}{d^2\Phi_i(\Theta_i)}.$$

If $d^2\Phi_i(\Theta_i)$ comes from a small light emitting surface of radianceL_i, given equation 2.1, we have :

$$\mathbf{f_r}(\Theta_i, \Theta_r) = \frac{dL_r}{L_i \cos\theta_i \, d\Omega_i}. \tag{3.1}$$

Now, if we consider a small solid angle about Θ_r, the ratio of the reflected power $d^3\Phi_r$ in direction Θ_r to the incident power $d^2\Phi_i$ coming from direction Θ_i is

$$\frac{d^3\Phi_r}{d^2\Phi_i} = \mathbf{f_r}(\Theta_i, \Theta_r)\cos\theta_r \, d\Omega_r.$$

Thus,

$$\rho(\Theta_i) = \frac{d^2\Phi_r}{d^2\Phi_i} = \int_{2\pi} \mathbf{f_r}(\Theta_i, \Theta_r)\cos\theta_r \, d\Omega_r. \tag{3.2}$$

From the above formula, it appears that only the bidirectional reflectance function can describe accurately the spatial distribution of the reflected light power coming from an incident ray of light in a given direction. The reflectance function only deals with reflection in term of energy exchanges. Note, however, that the

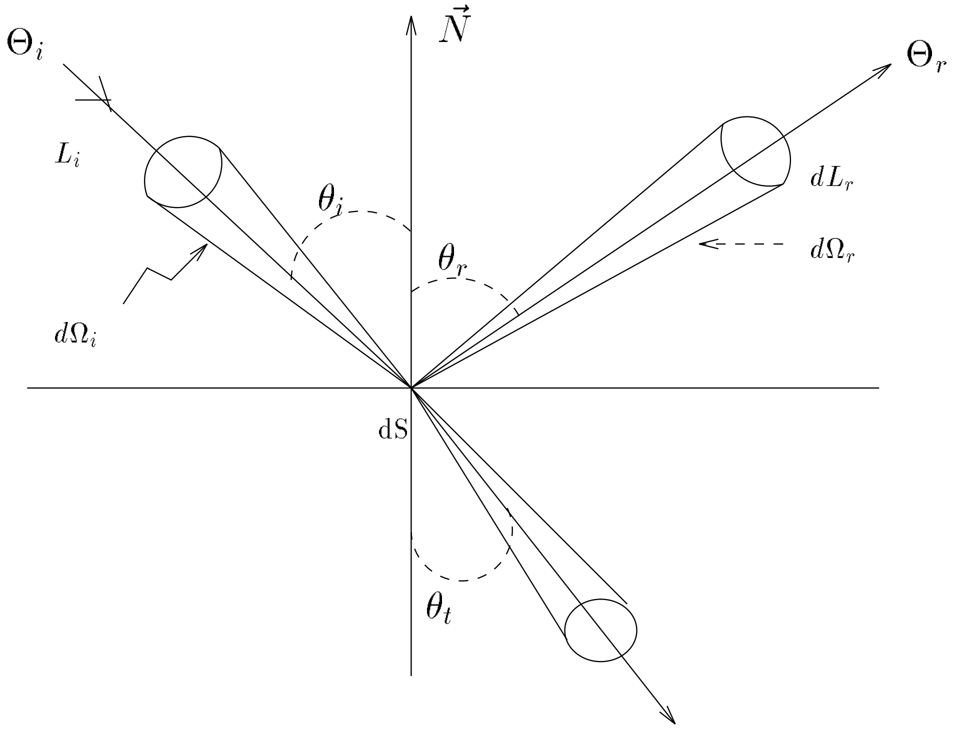

Figure 3.1: Geometry of reflection

bidirectional reflectance is irrelevant in case of perfect specular surfaces because it becomes a Dirac function δ :

$$\mathbf{f_r}(\Theta_i, \Theta_r) = \frac{\delta(\theta_i - \theta_r, \phi_i - \phi_r)\rho(\Theta_i)}{\cos\theta_r \sin\theta_r}$$

where θ_r, ϕ_r define the direction of perfect mirror reflection. In this case, the reflectance is given by the Fresnel formula [4]

Similarly, the bidirectional transmittance $\mathbf{f_t}(\Theta_i, \Theta_r)$ is the ratio of the radiance dL_t in a direction of refraction Θ_t by the irradiance for a direction of incidence. It is given by:

$$\mathbf{f_t}(\Theta_i, \Theta_t) = \frac{dL_t}{dA} = \frac{radiance}{irradiance}.$$

3.2 Reflection Models

Several physics-based reflection models have been proposed in the literature [29, 12, 49]. Let us focus on that of Cook and Torrance [12]. With this model, the reflected light depends on the wavelength, the incidence angle, the roughness parameter, and the surface refractive index (this index is a complex number for metallic materials). This model takes into account the polarization of the light, the roughness and the masking/shadowing of the materials. Let us briefly review this model. This model can be easily extended to account for interference [16].

This model is expressed as:

$$R = sR_s + dR_d \quad \text{with} \quad s + d = 1$$

where R_d and R_s are respectively the diffuse and specular components, d and s are the proportions of the incident light which give rise to the diffuse and specular components respectively.

R_d is independent of the incident angle, and can be approximated by $\frac{F(\lambda,0)}{\pi}$ [12], where $F(\lambda, 0)$ is the Fresnel factor for a normal incidence.

R_s accounts for the roughness as well as for the masking/shadowing effects, and is expressed as:

$$R_s = \frac{1}{4\pi}\frac{F(\lambda,\theta).D.G}{\cos\theta_i \cos\theta_r},$$

where $F(\lambda, \theta)$ is the Fresnel factor, θ_i is the incidence angle (direction Θ_i), θ_r the reflection angle (direction Θ_r) and θ equals half of the angle between the directions Θ_i and Θ_r. G is the masking/shadowing function, and D models the roughness effect. In our implementation, D is the Beckman function. The Fresnel factor is given by the Fresnel formula.

In several books [51, 3, 50], we can find, for several materials, Fresnel factor curves $F(\lambda, 0)$ for normal incidence, as well as the refraction index $\hat{n}$ for the wavelength $\tilde{\lambda} = 589$ (Sodium D lines) which corresponds to the center of the visible spectrum. Given these data, $F(\lambda, \theta)$ can be approximated [12], for each wavelength, by:

$$F(\lambda_i,\theta) = F(\lambda_i,0) + \Big(F(\lambda_i,\frac{\pi}{2}) - F(\lambda_i,0)\Big)\frac{F(\tilde{\lambda},\theta) - F(\tilde{\lambda},0)}{F(\tilde{\lambda},\frac{\pi}{2}) - F(\tilde{\lambda},0)},$$

where $F(\tilde{\lambda}, \theta)$ is given by the Fresnel formula for $\hat{n}$.

In [50], for several materials, values of the refraction index are given for a certain number of wavelengths. In this case, $F(\lambda, \theta)$ can be exactly expressed with the Fresnel formula.

Knowing the expression of $F(\lambda, \theta)$, we can precompute it for each sample wavelength and for different values of θ (20 seem enough). These values allow to create a look-up table, from which any $F(\lambda, \theta)$ can be computed by a simple linear interpolation.

3.2.1 Transmission Model

So far, no physics-based transmission models have been proposed in the literature, but only an empirical one [23]. Rather than using an empirical transmission model, it is more realistic, for each material, to use transmittance values experimentally obtained with the help of a spectrophotometer [19]. In case of ideal specular refraction, R_s is no more than $1 - F(\lambda, \theta)$, and $s = 1$.

Chapter 4

Light Sources

4.1 Physical Light Sources

Real light sources are often assumed to be perfectly Lambertian for many global illumination algorithms. Such algorithms have been extended to handle light sources with any intensity distribution curves to simulate the outdoor lighting [7]. In this case light sources are assumed to be far enough from the ground and buildings to be considered as points. This section shows how area light sources with any spatial distribution function may be accounted for.

4.1.1 Light Sources Representation

Most of the time, data specifying area light sources (given by manufacturers) are:

- the geometry of the light source (size, materials, ...) and the luminous intensity distribution curves for both transversal and longitudinal planes. This intensity distribution is provided while assuming that the total luminous light power emitted in all the directions by the source is equal to 1000 lumens.
- the spectral energy distribution $\Phi(\lambda)$ of the lamp, and its total luminous light power Φ_t. Note that the spectral energy distribution $\Phi(\lambda)$ corresponds to a luminous light energy of 1 lumen. In this implementation, we assume that the spectral energy distribution is not modified by the internal reflecting components of the light source.

The actual intensity distribution of the light source is then obtained by scaling the provided distribution curves by 1000 since the provided data representing the intensity distribution corresponds to a flux of 1000 lumens.

4.1.2 Making These Informations Usable

As a global illumination model is expressed in terms of *spectral radiance*, we need to transform the spatial luminous intensity data into spatial spectral radiance data [32].

Luminous Intensity to Luminous Radiance

Let us consider the geometry of a light source element dS illuminating a surface element dA as shown in figure 4.1. The radiance of surface dS as seen from dA is:

$$L_d = \frac{d^2\Phi}{d\Omega dS \cos\alpha}$$

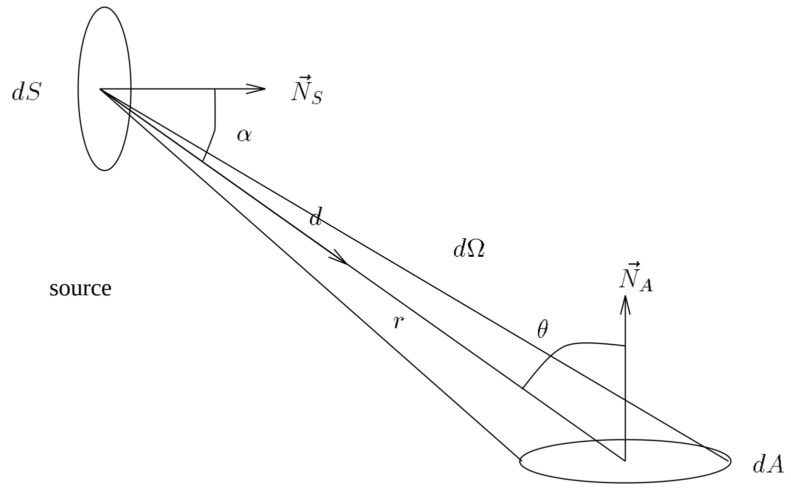

Figure 4.1: Geometry of illumination

with

$$d\Omega = \frac{dA \cos\theta}{r^2}.$$

Let us recall that the intensity is the light power per unit solid angle. The intensity in direction d is then given by $I_d = d\Phi_d/d\Omega$, where $d\Phi_d$ is the light power emitted in solid angle $d\Omega$. Since the light power $d\Phi_d$ can be rewritten as $I_d d\Omega$, the radiance in direction d can be expressed as:

$$L_d = \frac{dI_d}{dS \cos\alpha}.$$

If we suppose that the spatial intensity distribution is the same for any point of the light source, then we can write:

$$L_d = \frac{I_d}{S \cos\alpha} \tag{4.1}$$

This transformation has to be performed for all the sample directions in the transversal and longitudinal planes.

Luminous Radiance to Spectral Radiance

The next transformation consists in transforming the luminous radiance L_d into spectral radiance. The luminous radiance is related to the spectral radiance $L_d(\lambda)$ by the formula:

$$L_d = 680 \int_{\lambda_{min}}^{\lambda_{max}} L_d(\lambda) V(\lambda) d\lambda$$

where $V(\lambda)$ is the spectral sensitivity function. The available information is the spatial luminous radiance distribution L_d (previously computed) as well as the spectral energy distribution $\Phi(\lambda)$ to which the spectral radiance distribution is proportional. If we assume that the spectral radiance and the spectral energy distribution $\Phi(\lambda)$ (provided by manufacturers) have the same shape, then the spectral radiance $L_d(\lambda)$ for a sample direction d is obtained by solving the system:

$$\begin{cases} L_d = 680 \int_{\lambda_{min}}^{\lambda_{max}} L_d(\lambda) V(\lambda) d\lambda \\ L_d(\lambda) = scale\_factor * \Phi(\lambda), \forall\lambda \end{cases}$$

The integral can then be written as:

$$\begin{aligned} L_d &= 680 \int_{\lambda_{min}}^{\lambda_{max}} scale\_factor * \Phi(\lambda)V(\lambda)d\lambda \\ &= scale\_factor * 680 \int_{\lambda_{min}}^{\lambda_{max}} \Phi(\lambda)V(\lambda)d\lambda \end{aligned}$$

Since $680 \int_{\lambda_{min}}^{\lambda_{max}} \Phi(\lambda)V(\lambda)d\lambda = 1$ (by definition, see 4.1.1), $scale\_factor = L_d$ and the spectral radiance in direction d is simply:

$$L_d(\lambda) = L_d * \Phi(\lambda), \forall\lambda \tag{4.2}$$

4.1.3 Computing the Radiance in any Direction

We have shown how the spectral radiance is computed for the sample angles in both longitudinal and transversal planes. Now, let $d = (\theta, \phi)$ be a lighting direction, θ and ϕ being the polar and azimuthal angles respectively. This direction is projected onto both longitudinal and transversal planes, giving thus the two angles ϕ_l and ϕ_t. The spectral radiance $L(\lambda)(\theta, \phi)$ is obtained as follows: for both reference planes, the radiances $L_l(\lambda)(\phi_l)$ and $L_t(\lambda)(\phi_t)$ in directions ϕ_l and ϕ_t are computed by a linear interpolation involving the closest sample polar angles. Then the radiance in direction (θ, ϕ) is obtained by elliptic interpolation from $L_l(\lambda)(\phi_l)$ and $L_t(\lambda)(\phi_t)$:

$$L(\lambda)(\theta, \phi) = \sqrt{L_t^2(\lambda)(\phi_t)\cos^2\theta + L_l^2(\lambda)(\phi_l)\sin^2\theta} \tag{4.3}$$

In contrast to a linear interpolation, this method eliminates the discontinuities along the transversal and longitudinal axes of the lamp.

4.2 Sky Lighting

To obtain photorealistic images, sky lighting has to be accounted for. Indeed, architects and interior decorators need to know the lighting of their room at a given time, and under different weather situations. They wish to visualize the lighting as a function of the number of windows and their location in the scene. Let us see now how a radiosity algorithm handles sky lighting through windows.

Recall that the radiosity algorithm computes light interreflections between objects in a closed environment (energy balance). To extend this algorithm to environments containing windows we had to adapt this algorithm and make the following assumptions.

First we assume that there is no interaction with the objects lying outdoors (energy leaving the scene through the window will not come back into the scene).

On the other hand, the sky cannot be considered as any other area light source, since its spectral radiance distribution depends on the direction along which a point sees the sky. The sky radiance distribution cannot be represented as for artificial light sources. Indeed, the two longitudinal and transversal planes are not sufficient to accurately model the sky radiance.

Finally, the window panes absorb the incoming light but do not refract it.

4.2.1 Indoor lighting

As the sky light can enter the environment only through the windows, these latters can be considered as area light sources which reemit the sky light.

To compute the sky contribution to a patch P within the environment, the windows are sampled into small patches P_W. Let k and s be the centers of patches P and P_W respectively. If we assume that the sky radiance is constant over the solid angle dw subtended by P_W and having point k as apex, then the irradiance E_{ks} at point k is given by:

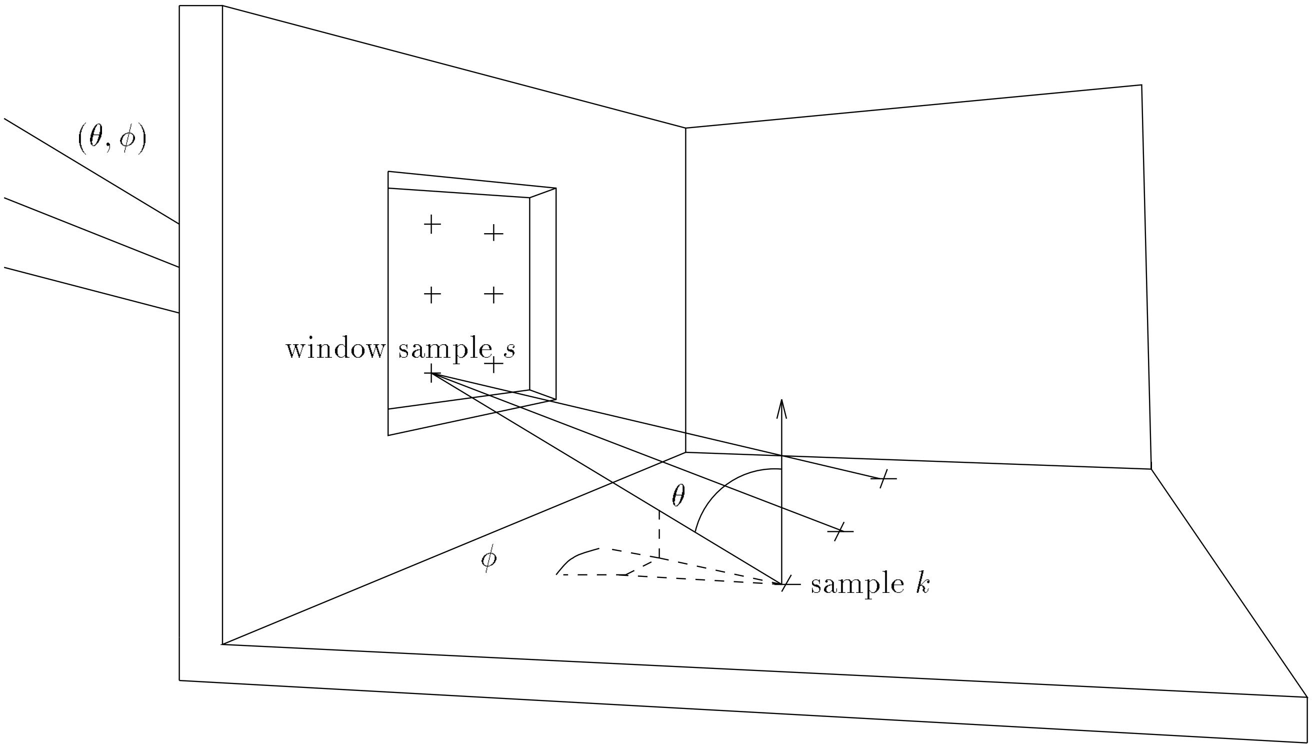

Figure 4.2: Window sampling

$$E_{ks} = dw \times SkyRadiance_{ks} \times \cos\theta, \tag{4.4}$$

where $SkyRadiance_{ks}$ is the sky radiance in direction ks and θ the angle formed by $\vec{ks}$ and the normal to patch P. If we assume that irradiance due to a window patch P_W is constant over a receiving patch P, then the total flux Φ_P received by patch P from patch P_W is:

$$\Phi_P = A_P \times E_{ks},$$

where A_P is the surface area of patch P.

In this way, the contribution of all the window patches is evaluated before starting the shooting process of the radiosity method.

4.2.2 Outdoor lighting

For each patch of the environment the contribution of the sky light is calculated as follows. A hemisphere is placed at the center C of each patch P, and is discretized into small surface elements named pixels. To each pixel corresponds a solid angle dw subtended by this pixel and having C as apex. A ray is traced from C through each pixel in order to determine if the sky is seen by C in the ray direction D. If yes, the irradiance due to the sky at point C is given by:

$$E_D = dw \times SkyRadiance_D \times \cos\theta,$$

where $\cos\theta$ is the polar angle associated with D.

Making the same assumptions as for window lighting, the total flux of patch P received from the sky is

$$\Phi_P = A_P \sum_{i \in [1,N]} E_{\Theta_i},$$

where Θ_i is a ray direction and N the number of rays for which C sees the sky. As before the shooting process will start once the sky contribution to all the patches has been calculated.

4.2.3 Sky Models

To compute the sky contribution, we have seen that the algorithm needs the sky radiance value for each direction (θ, ϕ). This sky radiance can be computed according to several models.

Uniform Sky

The uniform sky model simulates a cloudy sky and assumes that the radiance is the same in all directions. This radiance is often set to 3183 $candela/m^2$. The only advantage of the uniform sky model is its simplicity. Note that this model is not realistic.

Moon and Spencer sky

The Moon and Spencer sky is also cloudy. The sky radiance is not uniform but is three times greater for zenithal direction ($4093 cd/m^2$) than for the horizontal one ($1364 cd/m^2$). The Moon and Spencer formula is:

$$L_{\theta,\phi} = L_z \times \frac{1 + 2 \times \sin\theta}{3}, \tag{4.5}$$

where L_z is the zenithal radiance and θ, ϕ polar and azimuthal angles. This kind of sky normalized by the C.I.E in 1955 is, to date, the only one modeling the worst weather conditions. It seems a good representation of a dense cloudy sky.

Clear Sky

The clear sky model has been normalized by the C.I.E in 1973 [14]. It accounts for the sun location. In this model, the sky radiance depends on three angles (see figure 4.3):

- polar angle of the sun: z_0,
- polar angle of the direction in which the sky is seen: ϕ,
- azimuthal angle between the sun direction and the direction in which the sky is seen: α.

We need a last angle γ between the sun direction and that in which the sky is seen.

The radiance in the direction (ϕ, α) for a clear sky is then:

$$L_{\phi,\alpha} = L_z \times \frac{f(\gamma) \times \psi(\phi)}{f(z_0) \times \psi(0)}, \tag{4.6}$$

where f and ψ are two diffusion functions of variable x:

$$f(x) = 0.91 + 10exp(-3x) + 0.45\cos^2 x, \tag{4.7}$$

$$\psi(x) = 1 - exp(-0.32 \times \frac{1}{\cos x}), \tag{4.8}$$

Where L_z is the zenithal radiance. One empirical expression of L_z given by KROCHMANN is:

$$L_z = 100 + 63h_0 + h_0(h_0 - 30)exp((h_0 - 68)0.0346), \tag{4.9}$$

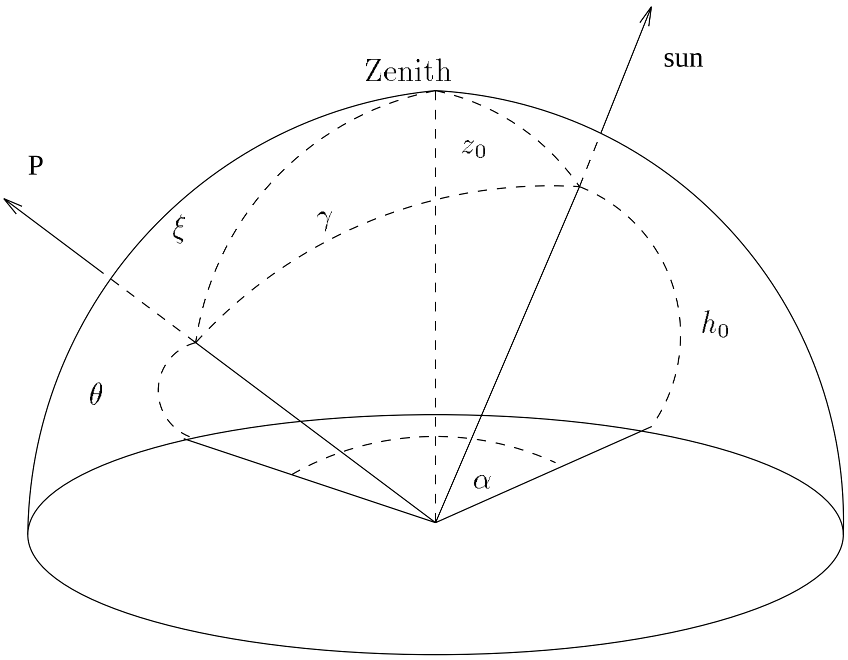

Figure 4.3: Angles for computation of clear sky radiances

where h_0 is expressed in degrees.

This sky model is the most time expensive but seems to be more realistic.

The models described above are only approximation made thanks to measurements. A better model should simulate sun light passing through participating media (atmosphere) as done in [31].

For cloudy sky models such as the uniform sky or Moon and Spencer sky, the direct sun contribution is not accounted for. But, in case of a non cloudy sky, the direct sun lighting could be added.

Chapter 5

Global Illumination

5.1 Global Illumination Model

Our goal is now to derive an expression of the light power leaving a surface S_i at a point $\bar{x}_i$ and reaching surface S_j at a point $\bar{x}_j$ (figure 5.1). This light power is due to the self emittance of S_i but also to its reflection and refraction properties. At this point, we leave out of account the occlusion problem and consider that any surface S_k or S_l contributes entirely to the illumination of surface S_i.

From now on, the following conventions will be adopted in all our notations :

- α_i and β_i refer, respectively, to angle of incidence and angle of reflection at point $\bar{x}_i$ of a surface S_i.
- α_i^b and β_l^b refer, respectively, to angle of incidence on the back of surface S_i and angle of transmission at point $\bar{x}_l$.
- in all our subscript or function argument notations, the order of the subscripts or the arguments follows the propagation of light with the source being the leftmost.

Let us call $L(\bar{x}_i, \bar{x}_j)$ the radiance of surface S_i at point $\bar{x}_i$ as seen from point $\bar{x}_j$ at surface S_j. Summing the contributions of all surfaces S_k, we have , using equation 3.1 (figure 5.1),

$$\begin{aligned} L(\bar{x}_i, \bar{x}_j) &= L^e(\bar{x}_i, \bar{x}_j) + \sum_k \int_{\Omega_{ik}} \mathbf{f_r}(\bar{x}_k, \bar{x}_i, \bar{x}_j) L(\bar{x}_k, \bar{x}_i) \cos\alpha_i d\Omega_{ik} \\ &+ \sum_l \int_{\Omega_{il}^b} e^{-\sigma\tau}\ \mathbf{f_t}(\bar{x}_l, \bar{x}_i, \bar{x}_j) L(\bar{x}_l, \bar{x}_i) \cos\alpha_i^b d\Omega_{il}^b \end{aligned} \tag{5.1}$$

where $\mathbf{f_r}(\bar{x}_k, \bar{x}_i, \bar{x}_j)$ is the bidirectional reflectance value for direction of incidence $\vec{\bar{x}_k\bar{x}_i}$ and direction of reflection $\vec{\bar{x}_i\bar{x}_j}$. $\mathbf{f_t}(\bar{x}_l, \bar{x}_i, \bar{x}_j)$ is the bidirectional transmittance value for direction of incidence $\vec{\bar{x}_l\bar{x}_i}$ and direction of transmission $\vec{\bar{x}_i\bar{x}_j}$. Ω_{ik} is the solid angle under which surface S_k is seen at point $\bar{x}_i$. Ω_{il}^b is the solid angle corresponding to the incident directions on the back of surface i, under which surface S_l is seen at point $\bar{x}_i$. $L^e(\bar{x}_i, \bar{x}_j)$ is the radiance due to self-emittance. σ is the absorption coefficient which is wavelength dependent, and τ the traveling length of the incident ray into the transparent object, say $\bar{x}_l\bar{x}_i$.

As

$$d\Omega_{ik} = \frac{dS_k \cos\beta_k}{||\bar{x}_k\bar{x}_i||^2},\ d\Omega_{il}^b = \frac{dS_l \cos\beta_l^b}{||\bar{x}_l\bar{x}_i||^2}$$

we have

$$L(\bar{x}_i, \bar{x}_j) = L^e(\bar{x}_i, \bar{x}_j) + \sum_k \int_{S_k} \mathbf{f_r}(\bar{x}_k, \bar{x}_i, \bar{x}_j) L(\bar{x}_k, \bar{x}_i) G(\bar{x}_k, \bar{x}_i) dS_k \tag{5.2}$$

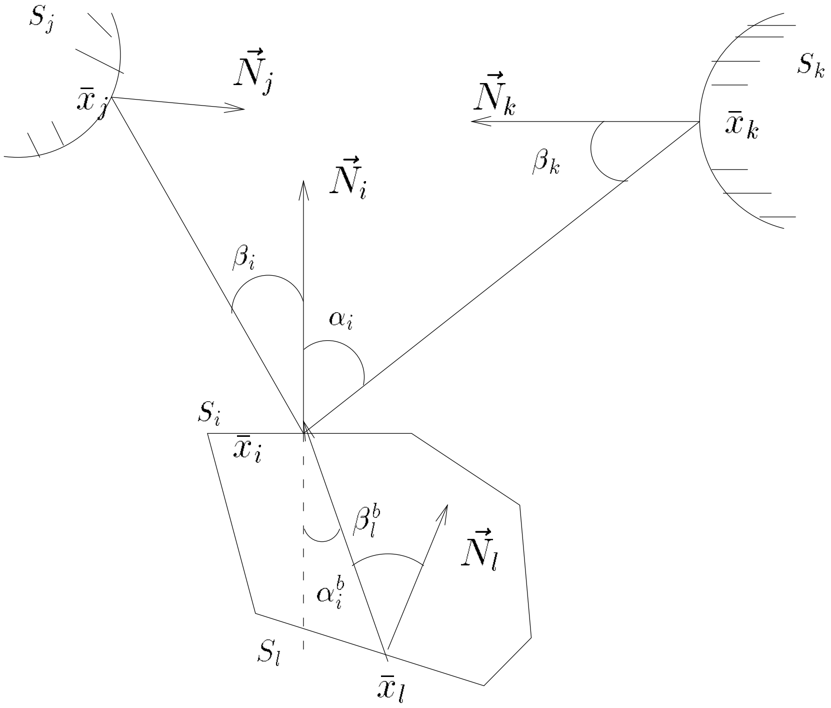

Figure 5.1: Geometry of light transport mechanism

$$+ \quad \sum_l \int_{S_l} e^{-\sigma\tau}\ \mathbf{f_t}(\bar{x}_l, \bar{x}_i, \bar{x}_j) L(\bar{x}_l, \bar{x}_i) G'(\bar{x}_l, \bar{x}_i) dS_l$$

where $G(\bar{x}_k, \bar{x}_i), G'(\bar{x}_l, \bar{x}_i)$ are purely geometric terms as

$$G(\bar{x}_k, \bar{x}_i) = \frac{\cos\alpha_i \cos\beta_k}{||\bar{x}_k\bar{x}_i||^2},\ \ G'(\bar{x}_l, \bar{x}_i) = \frac{\cos\alpha_i^b \cos\beta_l^b}{||\bar{x}_l\bar{x}_i||^2}.$$

The light occlusion effect can be accounted for by introducing a function $h(\bar{x}_i, \bar{x}_j)$ taking the value 1 if point $\bar{x}_i$ is visible from point $\bar{x}_j$ and 0 otherwise. Equation 5.3 then becomes

$$\begin{aligned} L(\bar{x}_i, \bar{x}_j) \quad &= \quad h(\bar{x}_i, \bar{x}_j)[L^e(\bar{x}_i, \bar{x}_j) + \sum_k \int_{S_k} \mathbf{f_r}(\bar{x}_k, \bar{x}_i, \bar{x}_j) L(\bar{x}_k, \bar{x}_i) G(\bar{x}_k, \bar{x}_i) dS_k \qquad (5.3) \\ &+ \quad \sum_l \int_{S_l} e^{-\sigma\tau}\ \mathbf{f_t}(\bar{x}_l, \bar{x}_i, \bar{x}_j) L(\bar{x}_l, \bar{x}_i) G'(\bar{x}_l, \bar{x}_i) dS_l] \end{aligned}$$

Equation 5.4 expresses the global illumination model. A similar equation can easily be derived from the solid angle formulation of equation 5.2. The above equation completely describes the light transport mechanisms between surfaces. The knowledge of $L(\bar{x}_i, \bar{x}_j)$ is sufficient to describe the spatial distribution of the light radiating from surface S_i.

5.2 One-Pass Model

If equation 5.4 is applied for all couples (S_i, S_j), a system of integral equations is obtained with the $L(\bar{x}_i, \bar{x}_j)$ as unknown functions. This system of equations expresses no more than the energy balance between self-emittance, absorption and reflection. Using a more condensed notation, this system can be written as follows

$$L_{ij} = h_{ij}\left[L^e_{ij} + \sum_k \mathcal{M}_{kij}(L_{ki})\right], \quad i,j \in [1,N] \tag{5.4}$$

where $L_{ij} = L(\bar{x}_i, \bar{x}_j)$, $L^e_{ij} = L^e(\bar{x}_i, \bar{x}_j)$ and $h_{ij} = h(\bar{x}_i, \bar{x}_j)$. As for the term $\mathcal{M}_{kij}(L_{ki})$, it is an integral operator which can be expressed as

$$\mathcal{M}_{kij}(L_{ki}) = \int_{S_k} \mathbf{f_r}(\bar{x}_k, \bar{x}_i, \bar{x}_j)L(\bar{x}_k, \bar{x}_i)G(\bar{x}_k, \bar{x}_i)dS_k + \int_{S_k} e^{\sigma\tau}\mathbf{f_t}(\bar{x}_k, \bar{x}_i, \bar{x}_j)L(\bar{x}_k, \bar{x}_i)G'(\bar{x}_k, \bar{x}_i)dS_k.$$

The solution of this system provides the radiance function of a surface as seen from any other surface. However, this is not sufficient to determine the image of the scene as seen by the observer unless the viewing system is included into the system of equations. This can be performed by placing a small non reflecting surface at the observer location or by adding a non reflecting surface that covers the whole virtual screen as suggested by Neumann et al [38]. In this last method, the additional computations can be limited to the L_{ij}between the screen surface and the visible surfaces of the screen. To some extent, this solution can be considered as view independent because only the L_{ij} between the observer and the visible surfaces need to be computed if the viewing system is changed.

5.3 Two-Pass Model

We shall see why the global diffuse and global specular components may be separated. In other words, why the specular component of the radiance of a patch can be computed after having completely evaluated the global diffuse component. For this purpose, we shall use an approach nearly similar to the one proposed in [43]. For the sake of simplicity, the self-emittance is assumed to be lambertian, the following demonstration holds even for directional light sources.

From [12, 39] we have :

$$\begin{aligned} \mathbf{f_r}(\bar{x}_k, \bar{x}_i, \bar{x}_j) &= d_i\mathbf{f_r}^d(\bar{x}_i) + s_i\mathbf{f_r}^s(\bar{x}_k, \bar{x}_i, \bar{x}_j) \\ \mathbf{f_t}(\bar{x}_k, \bar{x}_i, \bar{x}_j) &= d^t_i\mathbf{f_t}^d(\bar{x}_i) + s^t_i\mathbf{f_t}^s(\bar{x}_k, \bar{x}_i, \bar{x}_j) \end{aligned} \tag{5.5}$$

Let us rewrite system 5.4 as :

$$L = h[\mathcal{M}(L) + L^e]$$

where $L = \{L_{ij}\}, \mathcal{M} = \sum_k \mathcal{M}_{kij}, h = \{h_{ij}\}$ and $\{\}$ represents tensor. Separating the diffuse and specular components, the above system becomes

$$L = h[L^e + D(L)] + \mathcal{M}_s(L), \tag{5.6}$$

where the integral operators used in this system are given by :

- $\mathcal{M}_s = \{h_{ij}\sum_k \mathcal{M}^s_{kij}\}$,
- $\mathcal{M}^s_{kij} = h_{ij}s_i\int_{S_k} L(\bar{x}_k, \bar{x}_i)\mathbf{f_r}^s(\bar{x}_k, \bar{x}_i, \bar{x}_j)G(\bar{x}_k, \bar{x}_i)dS_k$
  $+ h_{ij}s^t_i\int_{S_k} L(\bar{x}_k, \bar{x}_i)\mathbf{f_t}^s(\bar{x}_k, \bar{x}_i, \bar{x}_j)G'(\bar{x}_k, \bar{x}_i)dS_k$
- $D = \{\sum_k \Theta_{kij}\}$,
- $\Theta_{kij} = d_i\int_{S_k} L(\bar{x}_k, \bar{x}_i)\mathbf{f_r}^d(\bar{x}_i)G(\bar{x}_k, \bar{x}_i)dS_k + d^t_i\int_{S_k} L(\bar{x}_k, \bar{x}_i)\mathbf{f_t}^d(\bar{x}_i)G'(\bar{x}_k, \bar{x}_i)dS_k$.

Let us call L^D the global diffuse component :

$$L^D = L^e + D(L).$$

Thus we have :

$$L = hL^D + \mathcal{M}_s(L).$$

Assuming that L^D is known, system 5.6 can be solved in a recursive manner by iterating the equation :

$$L^n = hL^D + \mathcal{M}_s(L^{n-1}.) \tag{5.7}$$

We see that the solution of equation 5.7 is :

$$L = (\sum_{m=0}^{\infty} \mathcal{M}_s^m)(hL^D) = S(hL^D), \tag{5.8}$$

where $S = \sum_{m=0}^{\infty} \mathcal{M}_s^m$ is the global specular operator. Note that D is the global diffuse operator.

Since

$$L^D = L^e + D(L),$$

we obtain :

$$L^D = L^e + D \cdot S(hL^D). \tag{5.9}$$

From equation 5.9, we see that we can separate the computation of the global diffuse component from the global specular one.

The notion of form factor can then be extended so as to take into account the specular effects contributing to the global diffuse component. These extended form factors are obtained by a Rieman's sum approximation of the integral operator $D \cdot S$.

In conclusion, the global illumination model can be computed in two passes :

- compute the extended form factors $(D \cdot S)$ for each patch, and solve the radiosity equation 5.9 to evaluate the global diffuse component of each patch [47, 43].
- evaluate the total radiance values (using equation 5.8) by means of either a combination of both Z-buffer and ray tracing techniques [47] or simple ray tracing [43] or distributed ray tracing [11].

However, the operator $D \cdot S$ involves complex computations [43].

5.4 Radiosity Model

The above light energy balance equation greatly simplifies in case of diffuse reflection because the bidirectional reflectance $\mathbf{fr}()$ depends only on $\bar{x}_i$, and L is independent of the view direction (Lambert's law). Thus :

$$\mathbf{fr}(\bar{x}_k, \bar{x}_i, \bar{x}_j) = \mathbf{fr}^d(\bar{x}_i)$$

$$L(\bar{x}_i, \bar{x}_j) = L^d(\bar{x}_i)h(\bar{x}_i, \bar{x}_j)$$

where the exponent d denotes the diffuse component.

Thus, we have :

$$\begin{aligned} T_{kij}(L_{ki}) &= \mathbf{fr}^d(\bar{x}_i) \int_{S_k} L^d(\bar{x}_k) G(\bar{x}_k, \bar{x}_i) h(\bar{x}_k, \bar{x}_i) dS_k \\ &= D_{ki}(L_k^d) \end{aligned}$$

For small surfaces, we may assume that $L^d(\bar{x}_k)$ is constant over surface S_k. Then we get:

$$D_{ki}(L_k^d) = \mathbf{f_r}^d(\bar{x}_i)L^d(\bar{x}_k)\tilde{F_{ki}^d},$$

where :

$$\tilde{F_{ki}^d} = \int_{S_k} G(\bar{x}_k, \bar{x}_i)h(\bar{x}_k, \bar{x}_i)dS_k$$

is the point to surface form-factor from point $\bar{x}_i$ to surface S_k.

In this case, the system of equations (5.4) becomes linear :

$$L_i^d = L_i^e + \mathbf{f_r}^d \sum_k L_k^d \tilde{F_{ki}^d}$$

Let us multiply by π each term of this equation, we obtain then:

$$\pi L_i^d = \pi L_i^e + \mathbf{f_r}^d \sum_k \pi L_k^d \tilde{F_{ki}^d}$$

As for a small diffuse surface the radiosity is equal to its radiance times π, and since $\mathbf{f_r}^d = \rho_i/\pi$, then this equation becomes:

$$B_i = E_i + \rho_i \sum_k B_k \frac{1}{\pi} \tilde{F_{ki}^d},$$

where B_i is the total radiosity of surface i, and E_i its self-emitted radiosity.

As we assume that the radiosity (or radiance) is constant over a surface, we replace $\frac{1}{\pi}\tilde{F_{ki}^d}$ by its average value over surface i , which gives:

$$\begin{aligned} F_{ki} &= \frac{1}{\pi S_i} \int_{S_i} \tilde{F_{ki}^d} dS_i \\ &= \frac{1}{\pi S_i} \int_{S_i} \int_{S_k} G(\bar{x}_k, \bar{x}_i)h(\bar{x}_k, \bar{x}_i)dS_k dS_i \end{aligned}$$

The above equation becomes then:

$$B_i = E_i + \rho_i \sum_k B_k F_{ki},$$

which is the well-known radiosity equation. F_{ki} is called form factor.

Chapter 6

Colorimetry

6.1 Introduction to Colorimetry

Colorimetry is the science of measuring color based on the physical properties of light and the psychovisual properties of the human visual system. This chapter summarizes some of the important points of works already done in colorimetry.

6.2 Trichromatic Theory of Color

Maxwell was the first one that tried to generate a large set of colors, by mixing three standard lights called color primaries. The results of his experiments showed that most of the colors of the visible spectrum could be reproduced by combining only the three color primaries: Red, Green and Blue [34]. These three color primaries must be linearly independent.

The three color primaries used in television are red, green and blue. These color primaries may be slightly different from one monitor to another. They act as a basis of a vector space called also *color space*. The coordinates of a color in this space are called *trichromatic components* or *tristimulus values*.

The trichromatic components P_i of a light of spectral distribution $E(\lambda)$ is given by :

$$P_i = \int_{380nm}^{780nm} E(\lambda)\sigma_i(\lambda)d\lambda,$$

where the $\sigma_i(\lambda)$'s are called *matching functions*.

6.2.1 CIE RGB

The *Commission Internationale de l'Eclairage* (CIE) proposed in 1930, three color primaries: Red, Green and Blue. The three associated matching functions are $\bar{r}, \bar{g}, \bar{b}$. They depend on the display device.

6.2.2 CIE XYZ

The CIE has normalised a color space, in which the three color primaries X, Y, and Z are not physical colors. The advantage of this color space is that it is independent of the used display device. The particularity of this color space is that the Y component corresponds to the visual luminance of the spectrum and is obtained by taking into account the sensitivity of a reference observer.

The trichromatic components X,Y and Z of a light of spectral distribution $E(\lambda)$ are obtained by applying the following formula:

$$X = K \int_{380}^{780} E(\lambda)\bar{x}(\lambda)d\lambda \tag{6.1}$$

$$Y = K \int_{380}^{780} E(\lambda)\bar{y}(\lambda)d\lambda \tag{6.2}$$

$$Z = K \int_{380}^{780} E(\lambda)\bar{z}(\lambda)d\lambda \tag{6.3}$$

where K is a constant used to normalize the results, and $\bar{x}(\lambda), \bar{y}(\lambda)$ and $\bar{z}(\lambda)$ are the matching functions of the system XYZ (see figure 6.1). If $E(\lambda)$ is an absolute spectral energy distribution, then K can be selected to equal K_m ($680\, lumen/watt$). On the other hand, if $E(\lambda)$ is a relative spectral energy distribution, then K is selected such that bright white has a Y value of 100, then other Y values will be in the range of 0 to 100. Thus,

$$K = \frac{100}{\int E_w(\lambda)\bar{y}(\lambda)d\lambda},$$

where $E_w(\lambda)$ is the spectral energy distribution for any standard white light source (D6500).

Note that the tristimulus values are positive across the entire visible spectrum.

The CIE standard chromaticity coordinates x, y, z are generated by projecting the tristimulus values on the $x + y + z = 1$ plane so that:

$$\begin{aligned} x &= X \,/\, (X+Y+Z) \\ y &= Y \,/\, (X+Y+Z) \\ z &= Z \,/\, (X+Y+Z) \\ 1 &= x + y + z \end{aligned}$$

A common specification for color is Y, x, y, where Y describes the luminance of the color (response to brightness) and x, y defines a point on the chromaticity diagram. The chromaticity diagram gives an indication of the color independent of its brightness.

The CIE standard are widely used in industry for describing colors.

6.2.3 Transformation from XYZ to RGB

The transformation of a color from space RGB to space XYZ is expressed as:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

The coefficients of the transformation matrix are given by [36]:

$$\begin{aligned} X_r &= x_r C_r, \quad X_g = x_g C_g, \quad X_b = x_b C_b \\ Y_r &= y_r C_r, \quad Y_g = y_g C_g, \quad Y_b = y_b C_b \\ Z_r &= z_r C_r, \quad Z_g = z_g C_g, \quad Z_b = z_b C_b \end{aligned}$$

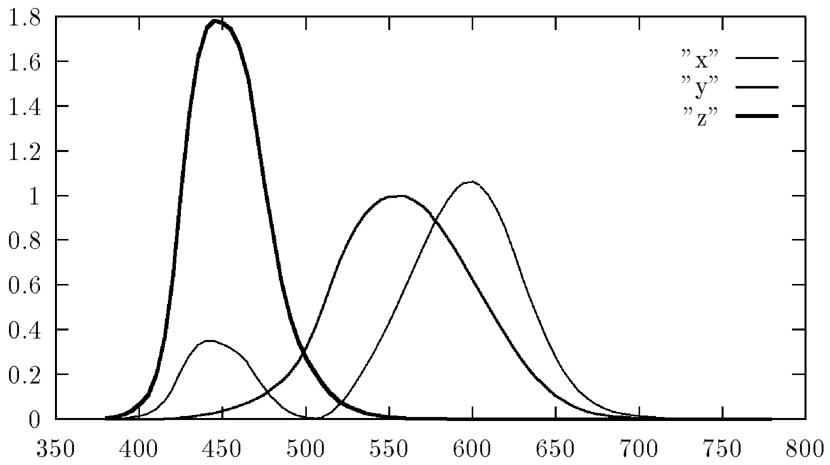

Figure 6.1: XYZ spectral matching functions

$$
\begin{aligned}
C_r &= \frac{Y_w}{y_w} \frac{x_w(y_g - y_b) - y_w(x_g - x_b) + x_g y_b + x_b y_g}{D} \\
C_g &= \frac{Y_w}{y_w} \frac{x_w(y_b - y_r) - y_w(x_b - x_r) + x_r y_b + x_b y_r}{D} \\
C_b &= \frac{Y_w}{y_w} \frac{x_w(y_r - y_g) - y_w(x_r - x_g) + x_r y_g + x_g y_r}{D} \\
D &= x_r(y_g - y_b) + x_g(y_b - Y_r) + x_b(y_r - y_g)
\end{aligned}
$$

where x_r, y_r, x_g, y_g, x_b and y_b are the chromaticity coordinates of the phosphors of the display device. Generally, these data are provided by manufactures, or can be measured. x_w et y_w are the chromaticity coordinates of the *white point* of the display, and Y_w its luminance. This last values has to be measured, because it depends on the calibration of contrast and brightness of the monitor. For these reasons, the coordinates of the white point are often assumed to be the coordinates of a reference white (ex: for normalized D6500 white, $x_w = 0.313$ and $y_w = 0.329$). If the luminance of the white point cannot be measured, we may use any arbitrary value as $Y_w = 1$.

6.2.4 Chromatic distance between colors

Two main color spaces are used to express the difference between two colors [17].

CIELUV space

This color space (also known as $L^*u^*v^*$) has been established in 1964 and adopted by the CIE in 1978. The three components in this space are expressed by:

$$
\begin{aligned}
L^* &= 166(Y/Y_n)^{0.5} - 16, Y/Y_n > 0.01, \\
u^* &= 13L^*(u' - u_n) \\
v^* &= 13L^*(v' - v_n)
\end{aligned}
$$

where

$$
\begin{aligned}
u' &= 4X/(X + 15Y + 3Z) \\
v' &= 9Y/(X + 15Y + 3Z) \\
u_n &= 4X_n/(X_n + 15Y_n + 3Z_n) \\
v_n &= 9Y_n/(X_n + 15Y_n + 3Z_n)
\end{aligned}
$$

X_n, Y_n and Y_n being the trichromatic components of the reference white (ex: D6500). In this space the difference between two colors is expressed as:

$$\Delta E = (\Delta L^{*2} + \Delta u^{*2} + \Delta v^{*2})^{0.5} \tag{6.4}$$

This system is well suited for the evaluation of small color differences, and can be used to detect and to eliminate aliasing defects.

CIELAB space

Another system is sometimes used, called CIELAB or L^*a^*b* and is more suitable for measuring important differences between colors. The difference is also expressed as

$$\Delta E = (\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2})^0.5 \tag{6.5}$$

with

$$
\begin{aligned}
L^* &= 166(Y/Y_n)^0.33 - 16 \\
a^* &= 500[(X/X_n)^{0.33} - (Y/Y_n)^{0.33}] \\
b^* &= 200[(Y/Y_n)^{0.33} - (Z/Z_n)^{0.33}]
\end{aligned}
$$

if (X/X_n), (Y/Y_n) and (Z/Z_n) are bigger than 0.01.

6.3 Spectrum sampling

To display a light on a display device, the three trichromatic components RGB of its spectral distribution have to be calculated. The accuracy of this calculation strongly depends on the way the visible spectrum is sampled. It depends on both the sample values and their number. This section describes a wavelength selection method due to Meyer [35]. Moreover two approaches may be adopted for computing a synthetic image: trichromatic and spectral. They are also described in this section.

6.3.1 Trichromatic and spectral approaches

In case of a trichromatic approach, the light sources, the objects' color and their reflection and refraction properties are expressed in the RGB space by three components.

Such a trichromatic approach is in fact only an approximation of a spectral approach which considers spectra (spectral distribution of light, spectral reflectance, transmittance and absorption, refraction index depending on wavelength...) instead of trichromatic components. It provides rather acceptable results but our experiments have shown that the quality of images containing metallic materials is far enhanced when using a spectral approach.

Let us show now what is the approximation made in a trichromatic approach. If $E(\lambda)$ is the incoming light illuminating a surface, and $BRDF(\lambda)$ the bidirectional reflectance function depending on the incident angle, the reflected direction and the physical properties of the surface. The reflected spectrum is then:

$$S(\lambda) = BRDF(\lambda) \times E(\lambda)$$

The RGB components S_R, S_G and S_B of the reflected light are obtained by:

$$\begin{aligned} S_R &= \int_{380}^{780} BRDF(\lambda) \times E(\lambda)\, \bar{r}(\lambda)\, d\lambda \\ S_G &= \int_{380}^{780} BRDF(\lambda) \times E(\lambda)\, \bar{g}(\lambda)\, d\lambda \\ S_B &= \int_{380}^{780} BRDF(\lambda) \times E(\lambda)\, \bar{b}(\lambda)\, d\lambda \end{aligned} \tag{6.6}$$

For a trichromatic approach, all the quantities must be described by their RGB components: E_R, E_G and E_B for the incident light, and $BRDF_R$, $BRDF_G$ and $BRDF_B$ for the reflectance function. These triplets are obtained by:

$$\begin{aligned} E_R &= \int_{380}^{780} E(\lambda)\, \bar{r}(\lambda)\, d\lambda \\ E_G &= \int_{380}^{780} E(\lambda)\, \bar{g}(\lambda)\, d\lambda \\ E_B &= \int_{380}^{780} E(\lambda)\, \bar{b}(\lambda)\, d\lambda \\ BRDF_R &= \int_{380}^{780} BRDF(\lambda)\, \bar{r}(\lambda)\, d\lambda \\ BRDF_G &= \int_{380}^{780} BRDF(\lambda)\, \bar{g}(\lambda)\, d\lambda \\ BRDF_B &= \int_{380}^{780} BRDF(\lambda)\, \bar{b}(\lambda)\, d\lambda. \end{aligned}$$

The RGB components of the reflected light are then:

$$\begin{aligned} S_R &= BRDF_R \times E_R \\ S_G &= BRDF_G \times E_G \\ S_B &= BRDF_B \times E_B \end{aligned} \tag{6.7}$$

We can compare the two expressions of the red component of the reflected light for both approaches. The spectral approach leads to:

$$S_R = \int_{380}^{780} BRDF(\lambda) E(\lambda)\, \bar{r}(\lambda)\, d\lambda,$$

while the trichromatic approach gives:

$$S_R = \int_{380}^{780} BRDF(\lambda)\, \bar{r}(\lambda)\, d\lambda \int_{380}^{780} E(\lambda)\, \bar{r}(\lambda)\, d\lambda$$

These two last equations show that the trichromatic approach approximates an integral by the product of two integrals, which is not mathematically correct. Nevertheless, this approach may be satisfying, especially in the case of non conductor materials or other complex surfaces.

6.3.2 Meyer's method

an optimal color space

The AC1C2 color space [35] has been derived from the SML color system, where the matching functions $\bar{s}(\lambda)$, $\bar{m}(\lambda)$ and $\bar{l}(\lambda)$ are the fundamental spectral sensitivities. It is oriented so that its axes are oriented along the most dense color regions. Its three axes have an importance that is proportional to the density of these regions. A is more important than C1, and C1 more important than C2.

transformation XYZ - AC1C2

the transformation from XYZ space to AC1C2 space is obtained by:

$$\begin{pmatrix} A \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} -0.0177 & 1.0090 & 0.0073 \\ -1.5370 & 1.0821 & 0.3209 \\ 0.1946 & -0.2045 & 0.5264 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

evaluation of the AC1C2 components

The three AC1C2 components are obtained by evaluating the following integrals:

$$A = \int_{380}^{780} E(\lambda)\bar{a}(\lambda)d\lambda$$

$$C_1 = \int_{380}^{780} E(\lambda)\bar{c_1}(\lambda)d\lambda$$

$$C_2 = \int_{380}^{780} E(\lambda)\bar{c_2}(\lambda)d\lambda$$

The computation of these integrals can be approximated using Gaussian quadrature by:

$$A = \sum_{i=1}^{n_1} H_i E(\lambda_i)$$

$$C_1 = \sum_{i=1}^{n_2} H_i E(\lambda_i)$$

$$C_2 = \sum_{i=1}^{n_3} H_i E(\lambda_i)$$

where coefficients H_i are the weights of the samples λ_i depending on n_1, n_2 and n_3 the number of samples for each component A, C1 and C2.

wavelength selection technique

The wavelengths will depend on the number of samples. Their values and associated weight have been computed for different number of samples, in order to keep the computation of A, C1 and C2 as accurate as possible.

The error produced using this method has been evaluated. It is minimized if more samples are used and if they are selected according to the following strategy: if you do not use the same number of samples for the three components, use more samples for A than for C1, and more samples for C1 than for C2. In the case we use only 4 samples, the wavelengths and the associated weights for each component are:

A		C1		C2	
λ_i	H_i	λ_i	H_i	λ_i	H_i
490.9	0.18892	490.9	0.31824	456.4	0.54640
557.7	0.67493	631.4	-0.46008		
631.4	0.19253				

Whatever the number of samples, this method keeps error under that produced when using other methods. Its main drawback is that it cannot be used in the case of spikey spectral distributions.

Chapter 7

Image Display and Visual Perception

7.1 Visual perception

The human eye converts luminance into a visual sensation, called *brightness*. The range of visible luminance is 10^{-6} to 5.10^4 cd/m^2. The visual sensation is related to the luminance, but is not linear, and depends on the ambiant level of illumination. As a first approximation, the law of sensitivity can be considered as logarithmic. The variation of sensitivity is mainly due to the following phenomena: the the size of the iris varies as the luminance changes, and the sensitivity of the retina is modified. The sensitivity function of the eye has been normalized by the CIE and is known as Weber's law (see figure 7.1).

To generate a luminous signal that will impress the eye the same way, one should take into account the sensitivity of the eye, in the context of a real scene, as well as in the context of an observation of the image on the display device.

Because of the lack of knowledge on the eye response, it is very difficult to find a function [46] relating the luminance of the real world to the values to be displayed on a monitor. These values depend on the characteristics of the monitor, and on the illumination level of the room containing the monitor.

To face this difficulty of displaying a calculated image, two processes may be applied: the first one consists in scaling the image so that it fits in the color range of the monitor, while the second aims at correcting the non-linearities of the monitor.

7.2 Color Clipping and Gamma Correction

Synthetic image generation results in a set of radiances that have to be displayed on a monitor. The range of these values may sometimes be very important, especially due to highlights resulting from specular reflections. Visible windows are also source of important radiance variations. The transformation of a light spectrum into the color space of a monitor may also lead to negative values or to values that are larger than the highest displayable value. The color corresponding to negative values cannot be displayed. The problem is to scale an image so that it can be displayed and can provide the better visual impact on the observer, knowing that most of the time, a monitor offers only 256 levels for each RGB component. One technique aiming at obtaining images with a maximum dynamic and a minimum loss of information consists of the following steps. First, the range of displayed radiances is determined, then the corresponding colors are scaled and clipped.

7.2.1 Scaling the image

In order to find the best range for the radiances to be displayed, we make the following approximations commonly made by lighting engineers.

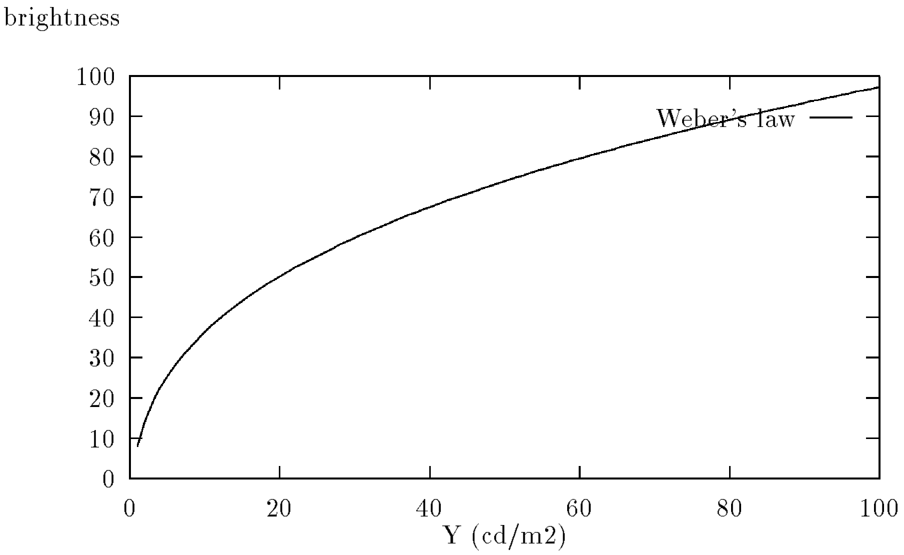

Figure 7.1: Weber's law

First, let us consider a scene lit by artificial light sources. The known datum (which is the total emitted flux) is the sum of all the luminous fluxes emitted by these sources. The assumptions made are the following. We assume that the total emitted flux reaches only the floor of the scene. The average emittance E of the floor is $\Phi_{total} \, / \, floor\_area$, and is expressed in lux/m^2. Assume also that the floor has an average reflectivity ρ_{ave}. Then, the approximated average radiance of the floor is simply :

$$L_{ave} = \frac{\rho_{ave} \times \Phi_{total}}{\pi \, floor\_area}.$$

If we assume that the maximum radiance L^{max} to be displayed is fixed to twice the approximated average radiance, then the range for RGB components is

$$[0, 2L_R^{max}] \times [0, 2L_G^{max}] \times [0, 2L_B^{max}],$$

where $L_R^{max}, L_G^{max}, L_B^{max}$ are the three RGB components of L^{max}.

These three components are determined as follows. Assume that the spectral distribution $L^{max}(\lambda)$ associated with L^{max} is equal to $C\Phi(\lambda)$ where C is a constant and $\Phi(\lambda) = 1 \, \forall \lambda$. Then the three XYZ components of L^{max} are

$$L_X^{max} = K \int C \, \bar{x}(\lambda) \, d\lambda, \; L_Y^{max} = K \int C \, \bar{y}(\lambda) \, d\lambda, \; L_Z^{max} = K \int C \, \bar{z}(\lambda) \, d\lambda.$$

This yields approximatively $L_X^{max} = L_Y^{max} = L_Z^{max} = KC$. These XYZ components are transformed to RGB components to give $L_R^{max}, L_G^{max}, L_B^{max}$.

Let MAXDISPLAY be equal to twice the maximum value of these three components. The final RGB components of the calculated radiances are computed by the algorithm of figure 7.2.

7.2.2 Color clipping

Let us recall that it is often not possible to display all the calculated colors on a monitor, some of them being out of the gamut of the monitor, and others exceeding its range.

There are several ways allowing clipping the colors [24]:

```
for each pixel
{
        if at lest one component is -ve or > MAXDISPLAY
                clip it
        /* scale */
        for each component C of the pixel
                Cdisplay = (C/MAXDISPLAY)*255
}
```

Figure 7.2: Color clipping

- Scale and clip the entire image until there are no luminances too high for display.
- Or, maintain the chromaticity and scale the luminance of the offending color.
- Or, maintain the dominant hue and luminance and desaturates the color.
- Or, clamp any color component exceeding 1 to 1.

Since no method gives the best results in any case, we simply set the negative value to 0, and values bigger than MAXDISPLAY to MAXDISPLAY. The value of MAXDISPLAY must be appropriately chosen.

7.2.3 Gamma Correction

The luminance L (Y component) produced by the phosphors of a monitor is not proportional to the input signal I. This response is non linear and is $L = kI^{\gamma}$, where γ is a parameter depending on the monitor, and is about 2.3 for the three RGB channels of typical rasters. If we want the displayed values to be proportional to the computed values, we have to transform the input signal I to $I^{1/\gamma}$. The luminance produced by the new input signal becomes then $L = kI$. In other terms, for example the component R of a pixel is replaced by $R^{1/\gamma}$.

7.2.4 Conditions for display

The best conditions for viewing pictures on a screen are met if the following instructions [5] are applied. First make sure that the room containing the display device is very dark , in order to avoid reflects on the screen. Then calibrate the monitor. To do this, display a totally black picture and set the brightness control so that you are just under the perception level. This setup must not be modified until the viewing conditions are changed.

Chapter 8

Implementation of Radiosity

8.1 Radiosity

Recall that the radiosity method assumes that all surfaces are assumed to be perfectly diffuse, i.e. they reflect light with equal radiance in all directions. The surfaces are subdivided into planar patches for which the radiosity at each point is assumed to be constant. Let us recall the radiosity equation, valid for each sample wavelength:

$$B_i = E_i + \rho_i \sum_{j=1}^{N} F_{ij} B_j,$$

where

- B_i : Exitance of patch i (Radiosity) ;
- E_i: self-emitted radiosity of patch i;
- ρ_i : reflectivity of patch i;
- F_{ij} : form-factor giving the fraction of the energy leaving patch i that arrives at patch j;
- N : number of patches.

This system of simultaneous equations represents the energy interchange via multiple interreflections and emission in the environment. The solution of this system is the patch radiosities which provide a discrete representation of the diffuse shading of the scene. This solution is independent of the view direction. Once the system of equation has been solved, each vertex of a patch is evaluated by averaging the radiosities of the patches sharing it. The image of the scene is then computed by applying Gouraud shading [21, 8].

Let us recall the expression of form-factor:

$$F_{ij} = \frac{1}{\pi A_i} \int_{A_i} \int_{A_j} \frac{h(\bar{x}_i, \bar{x}_j) \cos\theta_i \cos\theta_j}{r^2} dA_i dA_j,$$

where $\bar{x}_i$ is a point of patch i, and A_i the surface area of this patch. The term $h(\bar{x}_i, \bar{x}_j)$ expresses the visibility between a point of patch i and a point of patch j.

The form factor between a differential element of patch i (around a point $\bar{x}_i$) and patch j (figure 8.1) is:

$$F_{dA_iA_j} = \frac{1}{\pi} \int_{A_j} \frac{h(\bar{x}_i, \bar{x}_j) \cos\theta_i \cos\theta_j}{r^2} dA_j,$$

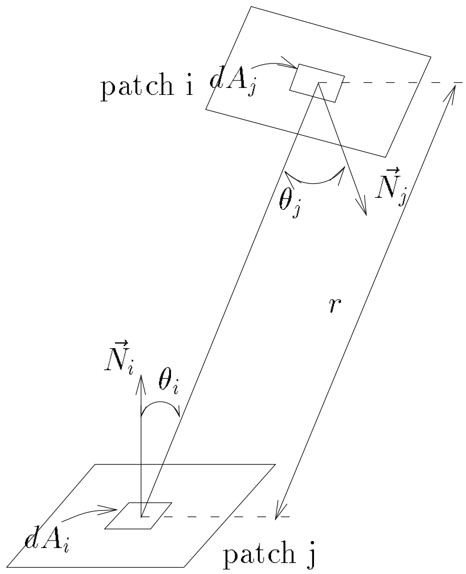

Figure 8.1: Form-factor

Note that if the two patches are far enough, this form factor is a good guess for F_{ij}. To compute F_{ij}, patch i is subdivided into R small elements dA_i^q (equivalent to differential elements) and all the form-factors $F_{dA_i^q A_j}$ are evaluated. F_{ij} is then equal to:

$$F_{ij} = \frac{1}{A_i} \sum_{q=1}^{R} F_{dA_i^q A_j} dA_i^q$$

8.1.1 Analytic expression for form factor

Point to Disk form factor

For a differential area dA_i and a disk A_j aligned axially, the form factor is given by (see figure 8.2):

$$F_{dA_i A_j} = \frac{A_j}{\pi r^2 + A_j}.$$

Differential area to polygon form factor

The form factor between a differential area dA_i and a polygon A_j is given by the following formula :

$$\begin{aligned} F_{dA_i A_j} &= \frac{1}{2\pi} \sum_{i=1}^{n} \beta_i \cos \alpha_i \\ &= \frac{1}{2\pi} \sum_{i=1}^{n} \beta_i N_i \bullet (\vec{E_i} \times \vec{E_{i+1}}). \end{aligned}$$

The geometry for this formula is given in figure 8.3 where $\vec{E_i}$ is the vector the endpoints of which are the center of dA_i and a vertex of polygon A_j, and N_i is the normal to dA_i. Note that this formula does not account for occlusion. However combined with visibility and clipping algorithms, this formula can be used in case of occlusion.

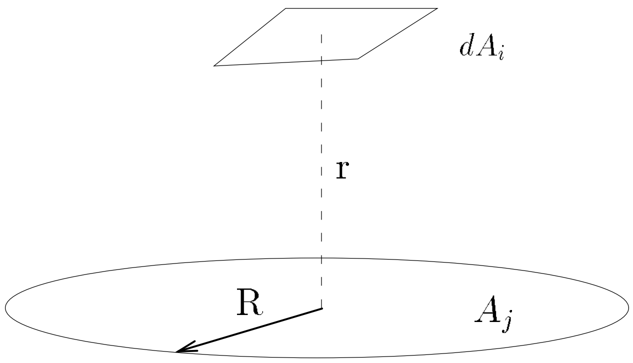

Figure 8.2: point to disk form factor

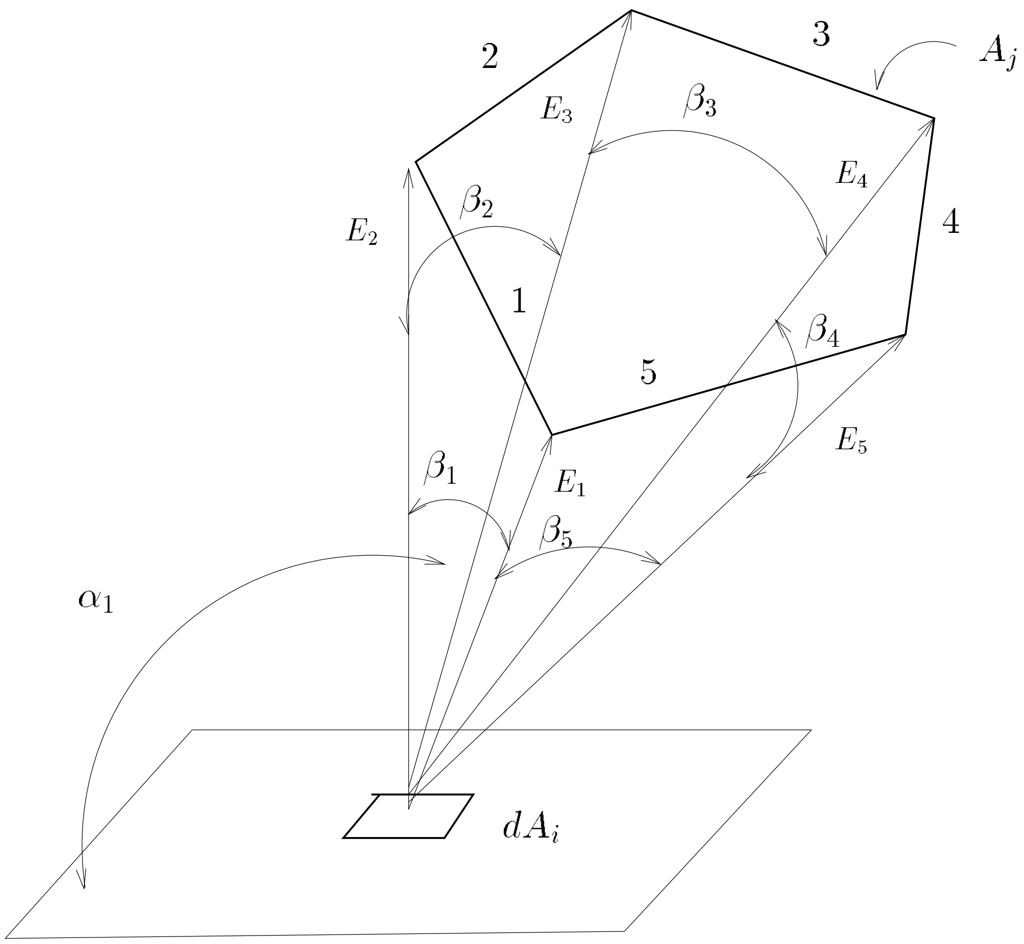

Figure 8.3: differential area to polygon form factor

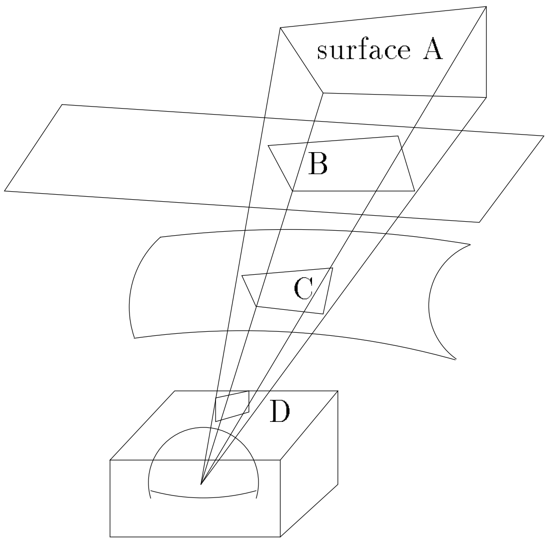

Figure 8.4: Similar projections

Contour integration

If surfaces A_i and A_j are completely visible to one another, the form factor between these surfaces can be expressed as a double contour integral by applying Stokes' theorem to the original formula. This yields:

$$F_{A_iA_j} = \frac{1}{2\pi A_i} \int_{C_i} \int_{C_j} \ln r \, dx_i dx_j + \ln r \, dy_i dy_j + \ln r \, dz_i dz_j ,$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) are points lying on the contours C_i and C_j of surfaces A_i and A_j respectively.
This formula is used when the visibility between two surfaces is determined a priori.

8.1.2 Form factor calculation by projection

Now, let us show how the differential to area form factor $F'_{dA_iA_j}s$ (see previous section for the notations) are calculated. The terms in $\cos\theta$ define in fact the projection of a surface onto another. If two patches similarly project on a given projection surface, then their form-factor (with a differential element of another patch) is thus similar (figure 8.4). Different kinds of projection surfaces have been proposed in the literature : Hemi-cube and Hemisphere.

Hemi-cube

The goal is to calculate the form-factor $F_{dA_iA_j}$ between a differential element of patch i and patches j. To simplify this computation, a simple projection surface is placed around dA_i that allows us to make use of classical projective rendering techniques (clipping, scan-conversion and z-buffering). This projection surface is an imaginary half-cube (figure 8.5) placed at the center of the receiving patch element dA_i [8]. A coordinate system is associated with this Hemi-cube, whose positive Z axis coincides with the normal to Patch A_i. It is made up of five faces: one full face facing in Z direction, and four half-faces in the $+X$, $-X$, $+Y$ and $-Y$ directions. The environment is transformed into this coordinate system. The five faces are divided into square pixels. The environment is then projected onto the five face planes. Each full face covers exactly a 90 degrees frustrum as viewed from the center of the hemi-cube. This creates clipping planes of equation $Z = X, Z = -X, Z = Y$ and $Z = -Y$, allowing for simple calculation to determine the part of a patch

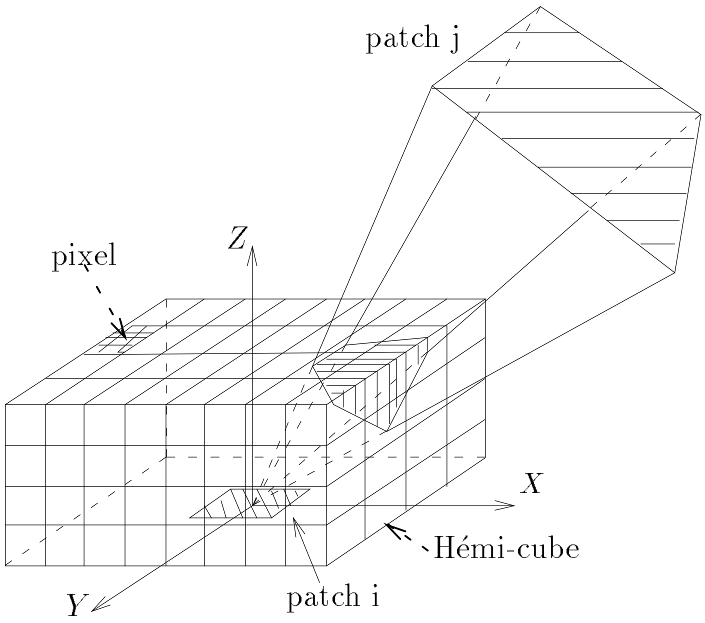

Figure 8.5: Hemi-cube

j projected on a face. After clipping, a patch j is projected on each face and then scan-converted. If two patches project on the same pixel of the hemi-cube, a Z-buffering technique is used to determine the one whose projection is closest to the center of the hemi-cube. An item buffer is maintained, giving for each pixel the patch seen at through the center of the pixel from the origin of the coordinate system.

The contribution of each pixel to the form-factor value varies and dependent on the pixel location. A delta form-factor is precalculated for a differential element dA_i to a pixel area A_{pixel} and stored in a look-up table. After determining which patch A_j is visible at each pixel on the hemi-cube, a summation of the delta form-factors for each pixel occupied by patch A_j determines the form-factor from the patch element dA_i to patch A_j. Then the hemi-cube is placed around another differential element of patch A_i and the above operations are repeated. Once all the differential elements of patch A_i have been considered, the form-factors F_{ij}i, for $j = 1, N$, are evaluated as seen before and the hemicube is positioned at the center of a differential element of another patch.

Let us recall that when patches A_i and A_j are assumed to be far-away, $F_{ij} = F_{dA_iA_j}$ and the center of the hemi-cube is the center of patch A_i.

Hemisphere

The method of form factor calculation makes use of a hemisphere as a projection surface and ray tracing.

A hemisphere is placed at the center of a patch p and is discretized by sampling the two polar angles θ and ϕ as shown in figure 8.6. The hemisphere is then discretized into surface elements ΔS, each one corresponds to a small form-factor called delta form-factor. To calculate the form-factors, a ray is cast from the hemisphere center and through each surface element ΔS of this hemisphere, i.e. in directions (θ_i, ϕ_j). Note that each ray corresponds to a delta form-factor. The intersection process between a ray and the scene may result in several points. Only the point closest to the ray origin is considered, and the identifier of the patch containing it is stored in an item buffer. Once all the rays have been cast from the center of a patch p toward all directions (θ_i, ϕ_j), the form-factors from this patch are calculated by scanning the item buffer and summing the delta form-factors associated with the rays along which a particular patch is visible.

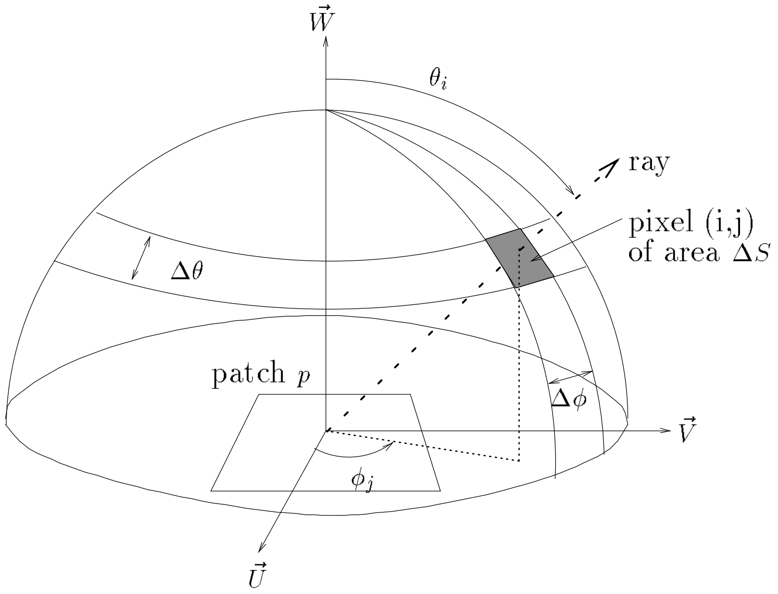

Figure 8.6: Hemisphere

This form-factor calculation technique is simpler and faster than the hemicube approach, since it avoids several processings such as polygon clipping, polygon filling and geometric transformations.

Another method of projection onto a hemisphere can be found in [45].

8.1.3 Area sampling method

Ray tracing

The problem is to compute the form factor between a differential area dA_i and a surface A_j using ray tracing. To this end, surface A_j is uniformly subdivided into R small surface elements dA_j^q and $F_{dA_i dA_j^q}$ is computed by using a modified version of the point-to-disk formula (see figure 8.7).

This modified point-to-disk formula is given by:

$$F_{dA_i dA_j^q} = \frac{dA_j^q \cos\theta_i^q \cos\theta_j^q}{\pi (r^q)^2 + dA_j^q}$$

Then

$$F_{dA_i A_j} = \sum_{q=1}^{R} \frac{dA_j^q \cos\theta_i^q \cos\theta_j^q}{\pi (r^q)^2 + dA_j^q} h(dA_i, dA_j^q),$$

where $h(dA_i, dA_j^q)$ is the visibility function which equals 1 if dA_i and dA_j^q are visible to one another and 0 otherwise. This visibility function is evaluated by tracing a ray from the center of dA_i to the center of dA_j^q).

The point-to-disk formula breaks down if the distance r is small relative to the differential area.

Monte-Carlo method

Both surfaces A_i and A_j are randomly sampled into points $\bar{x_i}$ and $\bar{x_j}$ respectively (figure 8.9). Rays, whose endpoints are $\bar{x_i}$ and $\bar{x_j}$, are then traced to determine the visibility function $h(\bar{x_i}, \bar{x_j})$. The form factor between these two surfaces is calculated by using the modified point-to-disk formula (see figure 8.8).

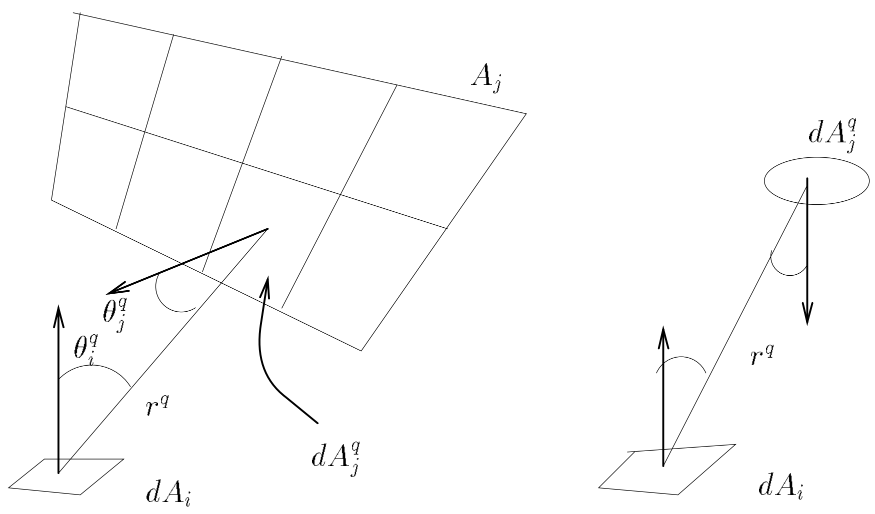

Figure 8.7: form factor calculated by ray tracing

$F_{ij} = 0$
for $k = 1$ to n do
    randomly select point $\bar{x}_i$ on the element i
    randomly select point $\bar{x}_j$ on the element j
    determine visibility between $\bar{x}_i$ and $\bar{x}_j$
    if *visible*
        compute $r^2 = (\bar{x}_i - \bar{x}_j)^2$
        compute $\cos\theta_i = \vec{r}_{ij} \bullet \vec{N}_i$
        compute $\cos\theta_j = \vec{r}_{ji} \bullet \vec{N}_j$
        compute $\Delta F = \frac{\cos\theta_i \cos\theta_j}{\pi r^2 + \frac{A_j}{n}}$
        if $(\Delta F > 0)$ $F_{ij} = F_{ij} + \Delta F$
$F_{ij} = F_{ij} * A_j$

where $\vec{r}_{ij}$ is the normalised vector from $\bar{x}_i$ to $\bar{x}_j$,
and $\vec{N}_i$ is the unit normal to element i at point $\bar{x}_i$
(and vice versa for switching i and j).

Figure 8.8: Pseudo code for Monte Carlo area-to-area form factor computation

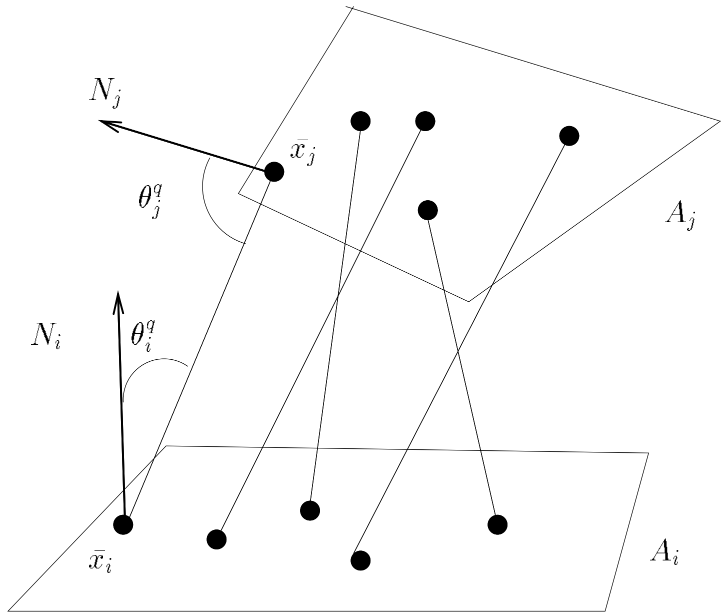

Figure 8.9: area-area form factor calculated by Monte Carlo

```
for all i do
        for each wavelength do
                B_i = E_i;
while not convergence do
        for all i do
                for each wavelength do
                        B_i = E_i + ρ_i Σ_{j=1,j≠i}^{n} F_ij B_j ;
```

Figure 8.10: Pseudo code for gathering

8.1.4 Solving the system of equations

Form-factors must be computed from every patch to every other patch resulting in memory and time complexities of $O(n^2)$. The very large memory required for the storage of these form-factors limits the radiosity algorithm practically. This difficulty was addressed by the progressive radiosity approach [10].

In the conventional radiosity approach, the system of radiosity equations is solved using Gauss-Seidel method. At each step the radiosity of a single patch is updated based on the current radiosities of all the patches. At each step illumination from all other patches is gathered into a single receiving patch (see figure 8.10). The convergence is met when $\| B^{k+1} - B^k \|_\infty$is below a certain threshold, where k is the iteration number. B^k is the radiosity vector computed at iteration k.

In the progressive radiosity approach, the solution is evaluated as shown in figure 8.11:

At each step, the illumination due to a single patch is distributed to all other patches within the scene. During the initial steps, the light source patches are chosen to shoot their energy since the other patches will have received very little energy. The subsequent steps will select secondary sources, starting with those surfaces that receive the most light directly from the light sources, and so on. Each step increases the

for all i do
  for each wavelength do
    $\Delta B_i = E_i$;
while not convergence do
  j = patch-of-max-delta-flux();
  for all i do
    for each wavelength do
      $\Delta Rad = \rho_i \Delta B_j F_{ji} \frac{A_j}{A_i}$;
      $\Delta B_i = \Delta B_i + \Delta Rad$;
      $B_i = B_i + \Delta Rad$;
  $\Delta B_j = 0$;

Figure 8.11: Pseudo code for shooting

accuracy of the result that can be displayed. Useful images can thus be produced very early in the shooting process. Note that, at each step, only a column of the system matrix is calculated, avoiding thus the memory storage problem.

The convergence criterion is met if $\| \Delta B \cdot A \|_\infty$ is below a certain threshold. This threshold could be a certain percentage of the sum of the total fluxes of the light sources. $\Delta B \cdot A$ is the vector of unshot fluxes.

Note that after convergence or a certain number of iterations of the shooting process, some residual fluxes remain unshot. The effects of these residuals can be approximated by an ambiant term B_{amb} [10]:

$$B_{amb} = R \sum_{j=1}^{N} \Delta F_{*j} B_j,$$

where F_{*j} corresponds to the contribution of patch j to the other patches and R characterizes the multiple interreflections:

$$F_{*j} = \frac{A_j}{\sum_{k=1}^{N} A_k}$$

$$R = 1 + \rho_{ave} + \rho_{ave}^2 + \rho_{ave}^3 + \ldots = \frac{1}{1 - \rho_{ave}},$$

where ρ_{ave} is the average reflectivity of the objects and is given by:

$$\rho_{ave} = \frac{\sum_{k=1}^{N} \rho_k A_k}{\sum_{k=1}^{N} A_k}.$$

To account for the the residual fluxes, the computed radiosities are updated as:

$$B_i = B_i + \rho_i B_{amb}.$$

8.1.5 Adaptive meshing

Improvements can be brought to the radiosity algorithm by adaptively discretizing the surfaces of the scene.

Discretizing a scene is a sampling problem which may give rise to aliasing effects. Indeed, areas in the scene with high radiosity gradients are poorly represented, particularly when the patches are large relative to the area over which the high radiosity gradients occur. To remedy this, one solution consists in subdividing the patches with high gradients into finer and finer patches [9]. Each subdivision level requires solving the linear system of radiosity equations to obtain the radiosities, and consequently the radiosity gradients. More elegant solutions, called *hierarchical radiosity*, have been proposed in [25, 26, 32]. They deal with an adaptive discretization of the objects' surfaces as shown in the next section.

8.1.6 Rendering step

Once the radiosities have been computed, all the view parameters have to be specified (viewer position, position and size of the screen, pixel resolution...) so as to start the rendering step.

This can be done by ray tracing. In this case, rays are traced from the viewpoint through each pixel. Suppose that a ray intersects a patch i of radiosity B_i at point P. The radiosity B_P of P is calculated by bilinear interpolation (in case of polygonal patch). The radiance L_{pixel} of the corresponding pixel is then:

$$L_{pixel} = \frac{B_P}{\pi}.$$

Note that B_i and L_{pixel} are wavelength dependent. For image display, L_{pixel} is converted into RGB components.

8.1.7 Texture mapping

Texture mapping can be accounted for in the radiosity algorithm. A first method has been proposed in [9]. A more accurate solution can be found in [20].

Let us describe now the method given in [9]. For a textured patch the reflectivity varies over its surface for each wavelength. First of all, the average reflectivity ρ_{ave} (for each wavelength) is calculated and the system of radiosity equations is solved to give the average radiosity B_{ave} for each patch. The texture values are accounted for only at the rendering step. Indeed, the final radiance of a pixel is calculated as:

$$L_{pixel} = \frac{\rho_{pixel}}{\rho_{ave}} \frac{B_{ave}}{\pi},$$

where ρ_{pixel} is the reflectivity surface point as seen through the pixel.

8.2 Hierarchical Radiosity

Hierarchical radiosity has been introduced in [25, 26]. The objective of this method is to avoid a finer meshing of the surfaces that make up an environment and to reduce the number of form factor calculations by adaptively subdividing these surfaces into a hierarchy (quadtree) of surface elements (see figure 8.12). A leaf of a hierarchy is called element while a node is no more than a group of elements. Interaction between a node of surface A and a node of surface B takes place if these two nodes can exchange energy. Each interaction requires one form factor calculation. That means one has not to compute form factors for each pair of leaf nodes (which is the case for traditional radiosity) but for each pair of nodes in interaction. As a result, the number of form factor calculations is reduced drastically as well as the memory storage. When two nodes of different surfaces interact one to another a link is established between them.

All the following data structures and algorithms, describing hierarchical radiosity, are extracted from [37].

8.2.1 Data structures

The quadtrees (hierarchy) and the link nodes are given in figure 8.13.

8.2.2 Refinement

Refinement consists in subdividing each surface, with respect to the others, into a hierarchy of elements. When two nodes are allowed to interact, a link is built between them. The refinement process is made by the procedure **Refine()** described in figure 8.14.

The role of the **oracle()** procedure is very important. It decides if two nodes can be linked or not. Recall that a node corresponds to either an element or a group of elements. We will see that if a link is established

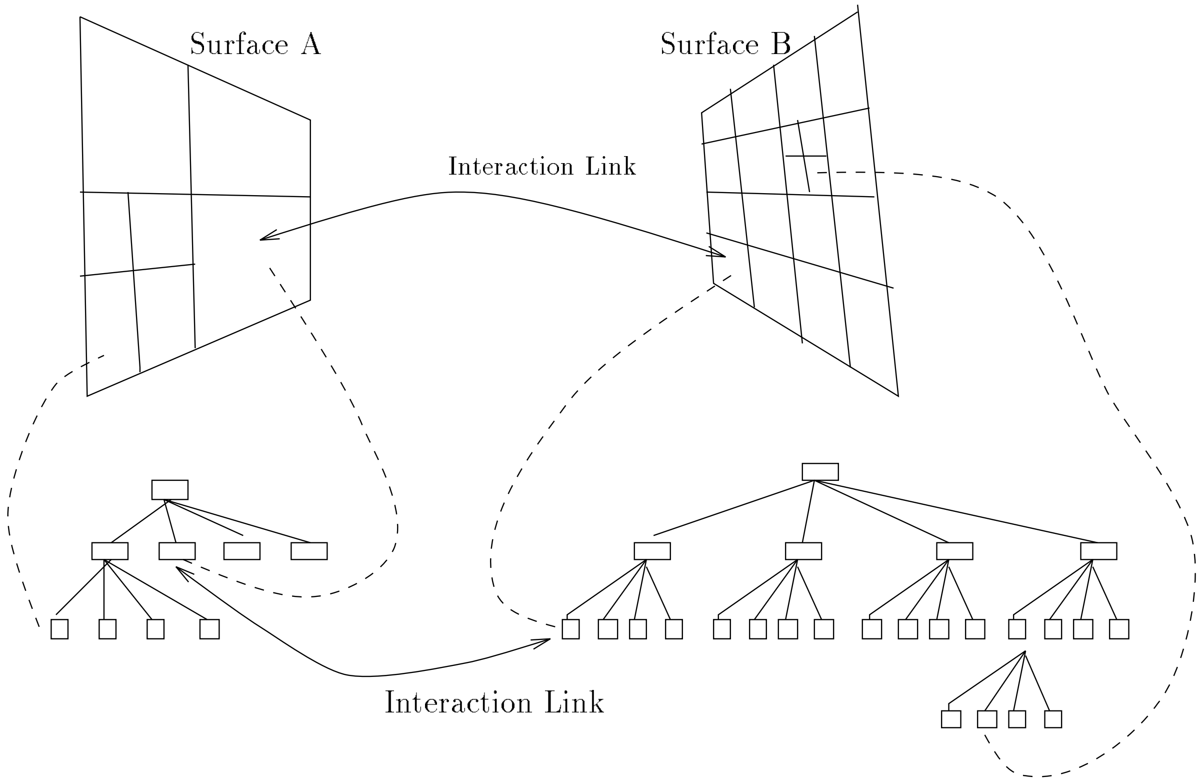

Figure 8.12: Hierarchy and interactions

```
struct Quadnode {
        float Bg[] ; /* gathering radiosity at sample λ's */
        float Bs[] ; /* shooting radiosity at sample λ's*/
        float E[] ; /* self emittance at sample λ's*/
        float area;
        float ρ[] /* reflectivity at sample λ's;
        struct Quadnode** children; /* pointer to list of
                                     four children*/
        struct Linknode* L; /* first gathering link of node */
}

struct Linknode {
        Quadnode* q; /* gathering node */
        Quadnode* p; /* shooting node */
        float Fqp; /* form factor from q to p */
        struct Linknode* next; /* next gathering link of node q */
};
```

Figure 8.13: Data structures

```
Refine(Quadnode *p, Quadnode *q, float F_ε)
{
        Quadnode which, r ;
        if (oracle(p, q, F_ε)) Link(p, q)
        else
                which = Subdiv(p, q);
                if (which == q )
                        for (each node r of q) Refine(p, r, F_ε)
                else if (which == p )
                        for (each node r of p)Refine(r, q, F_ε)
                else
                        Link(p,q);
}
```

Figure 8.14: Refine pseudo code.

then the form factor between the two associated nodes can be approximated by a **differential area-area** form factor. To do that, **oracle1()** computes an upper bound of the **differential area-area** form factor F_{pq} between patch p and q and the one between q and p, i.e. F_{qp}. There are two versions of **oracle()**: **oracle1()** and **oracle2()**. The first one relies on a geometric subdivision criterion while the second on a radiometric criterion (flux).

Subdiv(p,q) returns true if nodes below p and q (children) should be used. It returns p if it appears that lower level nodes of p will probably satisfy the subdivision criterion more rapidly, else the procedure returns q.

Link(p,q) establishes a link between p and q, computes the form factor between p an q and stores it in the link data structure. This form factor is in fact estimated as an upper bound of differential area-area form factor as will be shown later on.

After each pair of surfaces have been considered by **Refine**, the result is a set of links connecting two nodes of different quadtrees.

8.2.3 Solving the hierarchical system

The **SolveSystem()** procedure computes the solution of the hierarchical system of equations (figure 8.15).

GatherRad() gathers energy over each link at each receiving node. The gathered energy is stored in the field B_g (figure 8.16). **GatherRad()** corresponds to Jacobi's resolution method.

PushPull() pushes the gathered radiosity down to the children of each receiving node, and pulls the results back up the quadtrees by area averaging (see figure 8.17), thus preparing the radiosities B_s for the next iteration in **SolveSystem()**. *Pushing* and *Pulling* correspond, in signal processing, to reconstruction and decomposition respectively.

The convergence criterion used in **SolveSystem()** is: the maximum change in the radiosity values is below a threshold specified by the user.

8.2.4 The Oracle

The oracle function takes the decision to link or not two nodes p and q. In fact, a link is built if the form factor F_{pq} is small enough to consider the energy contribution of p to q as small, which amounts to say that the radiosity of q due to p can be considered constant over q. In **Oracle1()** (figure 8.18), F_{pq} is estimated

```
SolveSystem()
{
        Until Converged
                for(all surfaces p) GatherRad(p);
                for(all surfaces p) PushPullRad(p, 0.0);
}
```

Figure 8.15: **SolveSystem** pseudocode.

```
GatherRad(Quadnode *p)
{
        Quadnode *q;
        Link *L;

        p → B_g = 0;
        for(each gathering link L of p)
                /* gather energy across link */
                p → B_g += p → ρ * L → F_pq * L → q → B_s;
        for(each child node r of p)
                GatherRad(r);
}
```

Figure 8.16: **GatherRad** pseudocode.

```
PushPullRad(Quadnode *p, float B_down)
{
        float B_up, B_tmp;

        if(p → children == NULL)          /* p is a leaf */
                B_up = p → E + p → B_g + B_down;
        else
                B_up = 0;
                for (each child node r or p)
                        B_tmp = PushPullRad(r, p → B_g + B_down)
                        B_up += B_tmp * (r→area)/(p→area);
        p → B_s = B_up;
        return(B_up);
}
```

Figure 8.17: **PushPullRad** pseudocode.

```
float Oracle1(Quadnode *p, Quadnode *q, float F_ε)
{
        if (p → area < A_ε and q → area < A_ε)
                return(FALSE);
        if (EstimateFormFactor(p, q) < F_ε )
                return(FALSE);
        else
                return(TRUE);
}
```

Figure 8.18: **Oracle1** pseudocode.

as :

$$F_{pq} \approx \frac{\cos\theta}{\pi}\Omega_q,$$

where Ω_q is the solid angle whose appex is the center of p and subtented by a disk surrounding q.

This estimate is in fact an upper bound of the form factor between a differential area of p and q. In case of occlusion, this estimate can be weighted by a coefficient giving the percentage of visibility between p and q as given in [26].

Note that one can use a better estimate by calculating area-area form factor with Monte Carlo method.

Oracle1() estimates both F_{pq} and F_{qp}. If these two form factors are larger than a given threshold F_ϵ, then the node (or element) corresponding to the larger form factor is subdivided. If only one form factor is larger than F_ϵ, for example F_{pq}, then p is subdivided. When both are below F_ϵ, then a bidirectional link is established between p and q.

Note that **Oracle1()** uses a geometric subdivision criterion based on form factors. This may results in a large number of fine elements. It is more subtle to use a criterion based on the amount of energy transferred between two nodes. If this energy is smaller than a certain threshold, then a link is established. More precisely, if $F_{pq} \cdot B_q \cdot A_q \leq BF_\epsilon$ then a link is established. Since the radiosities are not known a priori, the refinement algorithm proceeds adaptively by using another oracle **Oracle2()** (figure 8.21).

8.2.5 Hierarchical radiosity algorithm

The hierarchical radiosity algorithm is given in figure 8.19. In the first pass of this algorithm, **Refine()** uses **Oracle2()** and establishes links at the highest levels unless the shooting (i.e. emitting) surface is a light source. Most of these links are built even though the shooting radiosities of most the surfaces are zero. These links will be refined in the second pass through **RefineLink()** (figure 8.20).

8.2.6 Rendering

The rendering process is same as in traditional radiosity. Only the leaf nodes (elements) of the quadtrees are rendered by ray tracing or by Gouraud shading. Attention has to be paid to the problem of discontinuities which may give rise to artifacts. Discontinuity comes from the non regular subdivision of the surfaces.

8.3 Error Estimates

If we broadly summerise the various steps that we have so far taken to carry out radiosity solution, then we will come up with 3 main points: (i) we are making certain assumptions (say constant radiosity over each patch) to convert the continuous integral equation to discrete linear system, (ii) to make sure that the

```
HierachicalRad(float BF_ε)
{
        Quadnode *p, *q;
        Link *L;
        int Done = FALSE;

        for (all surfaces p) p → B_s = p → E;
        for (each pair of surfaces p, q)
                Refine(p, q, BF_ε);
        while (not Done){
                Done = TRUE;
                SolveSystem();
                for (all links L)
                        /* RefineLink returns FALSE if any
                           subdivision occurs */
                        if (RefineLink(L, BF_ε) == FALSE)
                                Done = FALSE;
        }
}
```

Figure 8.19: **HierarchicalRad** pseudocode.

```
int RefineLink(Linknode *L, float BF_ε)
{
        int no_subdivision = TRUE;
        Quadnode *p = L → p;        /* shooter */
        Quadnode *q = L → q;        /* receiver */

        if (Oracle2(L, BF_ε))
                no_subdivision = FALSE;
                which = Subdiv(p, q);
                DeleteLink(L);
                if(which == q)
                        for (each child node r of q) Link(p,r);
                else
                        for (each child node r of p) Link(r,q);
        return(no_subdivision);
}
```

Figure 8.20: **RefineLink** pseudocode.

```
float Oracle2(Linknode *L, float BF_ε)
{
        Quadnode *p = L → p;        /* shooter */
        Quadnode *q = L → q;        /* receiver */
        if (p → area < A_ε and q → area < A_ε)
                return(FALSE);
        if (p → B_s == 0.0)
                return(FALSE);
        if ((p → B_s * p → Area * L → F_pq) < BF_ε)
                return(FALSE);
        else
                return(TRUE);
}
```

Figure 8.21: **Oracle2** pseudocode.

discretisation is correct, we are looking at the interaction kernel between every pair of surfaces (for example, in hierarchical radiosity method) of the environment and finally (iii) we are terminating the solution process by looking at certain convergence criterion.

There is something very important remaining to be done. It is the estimation of the error in the computation. We have so far not discussed any method to find out how close are the computed solution to the actual solution. Obviously it is a difficult problem and there does not exist any method to find out exact error in the computation. The best we can hope for is to get an estimate of this error. In this section we shall discuss a method which gives a reliable error estimate in the radiosity computation. The method was proposed by Lischinski *et al* in [33]. The method : (i) derives accurately two piecewise constant radiosity functions $\overline{B}$ and $\underline{B}$ that bound the exact radiosity function $B(\bar{x})$ *i.e.* $\overline{B}(\bar{x}) \leq B(\bar{x}) \leq \underline{B}(\bar{x})$, (ii) estimates the radiosity function as the average of these two bounds and (iii) estimates the error as half the difference between two bounds.

Thus the key step to the error estimation is the evaluation of radiosity bounds. We describe below the methods for their computation.

8.3.1 Computation of Radiosity Bounds

Lower Bound of Radiosity:

We have a linear radiosity system

$$B_i = E_i + \rho_i \sum_{j=1}^{N} F_{i,j} B_j$$

where N is the number of patches. If we substitute E_i and ρ_i in this equation by $\underline{E_i}$ and $\underline{\rho_i}$ which are respectively the infima of emission function and of reflectivity function over the surface patch i, and substitute $F_{i,j}$ by $\underline{F_{i,j}}$ which is the infimum on the point to area form-factor between the points in i and patch j

$$i.e.\ \ \underline{F_{i,j}} = \inf_{\bar{x} \in \mathrm{patch}_i} \int_{\mathrm{patch}_i} K(\bar{x}, \bar{y}) d\bar{y}$$

then we can write another linear system

$$\underline{B_i} = \underline{E_i} + \underline{\rho_i} \sum_{j=1}^{N} \underline{F_{i,j}}\, \underline{B_j}.$$

The solution of this system will be the lower bound radiosity $\{\underline{B}_i\}$.

Upper Bound of Radiosity:

Similar to the above paragraph we can substitute the E_i, ρ_i and $F_{i,j}$ by the corresponding supremums $\overline{E}_i$, $\overline{\rho}_i$ and $\overline{F}_{i,j}$ respectively and the solution of the resulting system should give us the upper bound of the radiosity. Unfortunately, the resulting linear system may not be amenable to iterative solution and even may not have a solution at all. Authors of [33] proposed a modification to the linear system which has a solution and also gives the upper bounds of the radiosity. They brought-in the modification while applying the Jacobi iteration to the system. In the standard Jacobi iteration the radiosity values at the $(k+1)$-th iteration step are calculated from the values obtained after the k-th step as follows:

$$B_i^{(k+1)} = E_i + \rho_i \sum_{j=1}^{N} F_{i,j} B_j^{(k)}.$$

Whereas the modified iteration method the radiosity values at the $(k+1)$-th iteration step are calculated as :

$$\overline{B}_i^{(k+1)} = \overline{E}_i + \overline{\rho}_i \left[\sum_{j=1}^{l} \overline{F}_{i,j} \overline{B}_j^{(k)} + \left(1 - \sum_{j=l+1}^{N} \overline{F}_{i,j} \right) \overline{B}_{l+1}^{(k)} \right]$$

where l is the largest index such that

$$\sum_{j=1}^{l} \overline{F}_{i,j} \leq 1.$$

This modified method converges and resulting solution has been shown in [33] to be the upper bound of the actual radiosity function.

Form-factor Bounds

In the above we bounded emittance, reflectivity and form-factor. In an environment emittance and reflectivity are mostly assigned to be constant known values. So bounding them makes no change in their values. Thus key to the radiosity bounds computation is the evaluation of supremum and infimum of the form-factor. One can evaluate them using a numerical optimization method as given in [2]. This evaluation could be expensive. One can get an approximation of these values by computing analytical point to polygon form-factor at various points of the receiver surface and finding a minimum and a maximum from them. Though these bounds are not exact, as the size of the patches decrease they converge to the exact value.

8.3.2 Bound Computation in Hierarchical Framework

We shall modify the hierarchical radiosity method to compute the radiosity bounds more efficiently. The modifications necessary are : (i) computation of $\overline{F}$ and $\underline{F}$ values for each link (ii) gathering of the radiosity bounds (iii) push/pull of the radiosity bounds. The gathering of upper bound must take in to consideration the transformation given in the earlier paragraph for convergence. The pseudo code of the gather algorithms are given in figure 8.22.

8.3.3 Improved Link Refinement Strategies

Original hierarchical algorithm [27] proposed a form_factor based refinement, so that a pair of patches with form_factors below a threshold were linked. This refinement strategy was subsequently improved [28] by basing the refinement on the brightness-weighted form_factor $(B_j.F_{i,j})$ value. This latter refinement was an improvement over the earlier because it did not refine the links with higher form_factors if the energy

```
GatherLowerBounds(node, B̲)
{
    foreach link ∈ node.links do
        B̲ += node.ρ * link.F̲ * link.source.B̲;
    if IsLeaf(node) then
        node.B̲ = B̲ + node.E;
    else
        foreach child ∈ node.child do
            GatherLowerBounds(child, B̲);
    node.B̲=min(child_1.B̲, child_2.B̲, ...);
}


GatherUpperBounds(node, contribList)
{
    foreach link ∈ node.links do
        add (link.F̄ and link.source.B̄) to contribList;
    if IsLeaf(node) then
        F̄Sum = 0;
        node.B̄ = node.E;
        CreateNewSortedList(contribList, NewList);
        foreach pair (F̄, B̄) ∈ NewList do
            if F̄Sum + F̄ ≤ 1 then
                F̄Sum += F̄;
                node.B̄ += node.ρ * F̄ * B̄;
            else
                node.B̄ += node.ρ * (1 − F̄Sum) * B̄;
                break;
    else
        foreach child ∈ node.child do
            GatherUpperBounds(child, contribList);
        node.B̄=max(child_1.B̄, child_2.B̄, ...);
}
```

Figure 8.22: Pseudo code for gathering lower and upper bounds.

transferred between them were negligible. This strategy reduced the number of unnecessary links and hence saved a lot of computational effort. Authors of [33] propose further improvements to this refinement strategy. The improvements are based on the following facts:

(i) Error in the radiosity results primarily from incorrect linking. If the kernel between the receiving patch and the emitting patch is not a constant function then there is bound to be an error in the radiosity value of the receiver patch.

(ii) Further, the error on this patch, gets distributed to other surfaces, exactly like the radiosity, along its shooting links.

$\Delta F_{i,j} = \overline{F}_{i,j} - \underline{F}_{i,j}$ is an indicator of the quantitative deviation of the kernel function from the constant function. Thus the refinement be based on $\overline{B}_j.\Delta F_{i,j}$ will be a better strategy than the simple $B_j.F_{i,j}$. If the link is correct then the $\Delta F_{i,j}$ will be zero and the link will not be refined irrespective of the amount of light flowing though it. At the same time, even if the $\Delta F_{i,j}$ is large, if the source is deem then also the link will not be refined.

According to the second observation related to the distribution of error, a surface having many shooting links is likely to amplify the error and hence increase the total computational error in the environment. So it will be a better idea to refine first the links of those surfaces which have maximum amplification factor. As the error is propagated like radiosity, the error amplification factor of a surface is same as its radiosity amplification factor or the importance[44]. So a much improved refinement strategy will be to base the refinement on $\overline{B}_j.\Delta F_{i,j}$ times the importance of the receiver *i.e.* Z_i. However, to do this we require the Z_i values. It has been shown in [44] that the importance calculation uses exactly the same machinery of the radiosity calculation method and hence can be interlinked with the radiosity method. Thus we can modify the radiosity bounds computation method discussed above to evaluate the importance. Authors of [33] propose to compute only the lower bound of importance *i.e* $\{\underline{Z_i}\}$ and refine all the links with $\overline{B}_j.\Delta F_{i,j}.\underline{Z_i} > \epsilon$. Ofcourse there is a small price to pay for this improvement. It is the additional effort of computing the importance of each patch in the environment.

Chapter 9

Projection Methods for Radiosity Computation

The radiosity is a function over points of the surfaces in the environment. An integral equation relates the value of this function at any point to the values at every other points of the environment. Thus computing this function by solving the integral equation at every point is impossible because there are infinite number of points. Unless one is interested in evaluating the function at finite number of points, one must find out a numeric method which converts the infinite problem into a finite problem. In the earlier chapters we have discussed one such method. In that, we have assumed that all the points belonging to a surface patch has the same radiosity value. There are only a finite number of surface patches in the environment and hence finite number of equations to solve. In this chapter we shall discuss the extension of this constant assumption to higher degree polynomial assumptions. To do that we shall introduce a general mathematical framework called function projection.

9.1 Weighted Residual Method for Function Approximation

Like the projection of vectors, one can project a function on the space spanned by a set of basis functions. The projection will be represented as a set of components.

Let us assume $\{\mathcal{N}_i(t)\}$ to be one such finite set of basis functions and $B(t)$ the function to be projected. Then the projection will give rise to the set

$$\langle B_1, B_2, \ldots, B_N \rangle \tag{9.1}$$

containing B_i as a *component of projection* of $B(t)$ corresponding to each basis function $\mathcal{N}_i(t)$. We can construct a new function $\hat{B}(t)$ from these coefficients and the basis functions as:

$$\hat{B}(t) = \sum_{j=1}^{N} B_j \, \mathcal{N}_j(t) \tag{9.2}$$

This function can serve as an approximation to the original function $B(t)$ and the component B_j's may be called as the *coefficients of approximation*. The difference between the original function and the approximation is very likely to be a nonzero function. This difference function $r(t)$ is called error function or *residue*.

There are various methods to compute the approximation coefficients. *Weighted residual method* [53] is one of them. This method sets the scalar products of the residual function and N known functions to zero and creates a linear system containing the unknown coefficients. The solution of the linear system gives these coefficients. The known functions are called *weight functions* $W(t)$. The derivation of the linear system is as follows:

$$
\begin{aligned}
r(t) = B(t) - \hat{B}(t) &= B(t) - \sum_{j=1}^{N} B_j\, \mathcal{N}_j(t) \\
\int r(t) W_i(t) dt &= 0 \qquad\qquad \text{for } \; i = 1 \ldots N \\
\Longrightarrow \quad \int B(t) W_i(t) dt &= \sum_{j=1}^{N} B_j \int \mathcal{N}_j(t) W_i(t) dt
\end{aligned}
\tag{9.3}
$$

Equation 9.3 is the resulting linear equation. The other terms in the linear equation are scalar products of known functions and must be computed before solving the linear system.

Of the various choices of the weight functions, the basis function itself serving as weight function is a popular choice. Thus the linear equation becomes :

$$\int B(t)\mathcal{N}_i(t) dt = \sum_{j=1}^{N} B_j \int \mathcal{N}_j(t)\mathcal{N}_i(t) dt.$$

A consequence of this choice is : the projection component of the residue function on each of the basis functions is zero. The resulting computation method is known as Galerkin method. In the Galerkin method, if the functions in the basis set are orthogonal, *i.e.*

$$\int \mathcal{N}_i(t)\mathcal{N}_j(t) dt = \begin{cases} \mathcal{A}_i & \text{iff } (i == j) \\ 0 & \text{otherwise,} \end{cases} \tag{9.4}$$

then we get a much simplified expression for each approximation coefficient which is

$$B_i \mathcal{A}_i = \int B(t)\mathcal{N}_i(t) dt. \tag{9.5}$$

We have so far discussed the projection of a function of single variable. We can simply extend this projection to functions of multiple variables. For example, a function of 2 variables $K(s,t)$ will be projected on an orthogonal basis as:

$$\hat{K}(s,t) \quad = \quad \sum_i \sum_j K_{i,j} \mathcal{N}_i(s)\mathcal{N}_j(t) \tag{9.6}$$

$$\text{where} \quad K_{i,j}\mathcal{A}_i\mathcal{A}_j \quad = \quad \int\int K(s,t)\mathcal{N}_i(s)\mathcal{N}_j(t) dt ds \tag{9.7}$$

In this case we have chosen $\mathcal{N}(s)$ and $\mathcal{N}(t)$ are the same basis functions but defined over different variable space s and t respectively. However, one is free to choose different basis functions for the different variable space.

9.2 Galerkin Method for Solving Integral Equation

We stated in the beginning that we have a task of computing a function $B(s)$ which is defined by an integral equation of the second kind *i.e.*

$$B(s) \quad = \quad E(s) + \int K(s,t) B(t) dt \tag{9.8}$$

where $E(s)$ and $K(s,t)$ are the known functions. Here we shall restate the problem and say that instead of trying to compute the exact function, we wish to find an approximation $\hat{B}(s)$ for the function as given in equation 9.2. Using the principles of Galerkin method we can carry out this computation. The process is as follows:
We first substitute every occurrence of $B(s)$ in the integral equation by $\hat{B}(s)$ and define a residue function as:

$$r(s) = \hat{B}(s) - E(s) - \int K(s,t)\hat{B}(t)dt \tag{9.9}$$

Then we minimise the residue by making it orthogonal to each of the basis functions. *i.e.*

$$\begin{aligned}
\int r(s)\mathcal{N}_i(s)ds = 0 \quad \text{or} \quad & \int \left(\hat{B}(s) - E(s) - \int K(s,t)\hat{B}(t)dt\right)\mathcal{N}_i(s)ds = 0 \\
\text{or} \int \hat{B}(s)\mathcal{N}_i(s)ds \quad &= \quad \int E(s)\mathcal{N}_i(s)ds + \int\int K(s,t)\hat{B}(t)\mathcal{N}_i(s)dtds \\
\text{or} \quad \int \left(\sum_j B_j\mathcal{N}_j(s)\right)\mathcal{N}_i(s)ds \quad &= \quad \int E(s)\mathcal{N}_i(s)ds + \int\int K(s,t)\left(\sum_j B_j\mathcal{N}_j(t)\right)\mathcal{N}_i(s)dtds
\end{aligned}$$

Substituting the relation given in equations 9.4, 9.5 and 9.7 in the above equation we get the following expression:

$$\begin{aligned}
B_i\mathcal{A}_i &= E_i\mathcal{A}_i + \mathcal{A}_i\sum_j K_{i,j}\mathcal{A}_j B_j \\
\text{or} \\
B_i &= E_i + \sum_j K_{i,j}\mathcal{A}_j B_j
\end{aligned}$$

In an integral equation the functions $E(s)$ and $K(s,t)$ are known and hence E_i's and $K_{i,j}$'s can be evaluated by using any quadrature method.
Thus using Galerkin method we have converted the problem of solving an integral equation to the problem of solving a system of linear equations. One can solve this linear system to get the unknown B_j's and construct approximation of the unknown function $B(s)$ as $\hat{B}(s)$ from equation 9.2. The size of the linear system, N, is equal to the number of basis functions chosen for approximation. An iterative solution of this linear system has a $\mathcal{O}(N^2)$ complexity. Also, setting up of the linear system requires the evaluation of E_i's and $K_{i,j}$'s for all i and j and hence has a $\mathcal{O}(N^2)$ complexity.

9.3 Radiosity Solution

Here we shall look at the exact radiosity equation. It is as follows:

$$B(\bar{x}) = E(\bar{x}) + \rho(\bar{x})\int_{\text{env}} \frac{\cos\theta_{\bar{x}}\cos\theta_{\bar{y}}}{r^2_{\bar{x}\bar{y}}} h(\bar{x},\bar{y})B(\bar{y})d\bar{y} \tag{9.10}$$

where $\bar{x}$ and $\bar{y}$ are surface points in the environment, $\theta_{\bar{x}}$ and $\theta_{\bar{y}}$ are the angle that the line joining points $\bar{x}$ and $\bar{y}$ makes with the surface normals at $\bar{x}$ and $\bar{y}$ respectively, $h(\bar{x},\bar{y})$ is the visibility between $\bar{x}$ and $\bar{y}$.
If we define a kernel function $K(.,.)$ as follows:

$$K(\bar{x},\bar{y}) = \rho(\bar{x})\frac{\cos\theta_{\bar{x}}\cos\theta_{\bar{y}}}{r^2_{\bar{x}\bar{y}}} h(\bar{x},\bar{y}), \tag{9.11}$$

then the radiosity equation is exactly same as that given in equation 9.8 and we can apply the Galerkin formulation derived above to solve the radiosity equation. This will amount to setting up a linear system of the following type:

$$
\begin{aligned}
B_i &= E_i + \sum_j K_{i,j}\mathcal{A}_j B_j && (9.12)\\
\text{where}\quad E_i\mathcal{A}_i &= \int_{\text{env}} E(\bar{x})\mathcal{N}_i(\bar{x})d\bar{x} \;\text{ and }\; K_{i,j}\mathcal{A}_i\mathcal{A}_j = \int_{\text{env}}\int_{\text{env}} K(\bar{x},\bar{y})\mathcal{N}_j(\bar{y})\mathcal{N}_i(\bar{x})d\bar{y}d\bar{x}
\end{aligned}
$$

Classical Radiosity Method

In the classical radiosity method we assume that the radiosity is constant over the surface patches. That means we approximate the radiosity function over a surface as a combination constant pieces. We can restate this by saying that we use basis functions $\mathcal{N}_i(\bar{x})$ which are defined as follows:

$$
\mathcal{N}_i(\bar{x}) = \begin{cases} 1 & \text{iff } (\bar{x} \in \text{Patch}_i) \\ 0 & \text{otherwise,} \end{cases} \tag{9.13}
$$

where Patch_i is the i-th surface patch of the environment. Function $\mathcal{N}_i(\bar{x})$ can be said to be a function with finite support and its support is the surface area of Patch_i.
The functions which are non-zero over only a part of the variable space are called piece-wise functions. Thus $\mathcal{N}_i(\bar{x})$'s are called piece-wise constant functions.

If the radiosity of the patch is B_i then we can write the radiosity function over the patch i as $B_{(i)}(x) = B_i\mathcal{N}_i(\bar{x})$ and the radiosity function over the whole environment as $B(x) = \sum B_i\mathcal{N}_i(\bar{x})$. From the analogy of this equation to the equation 9.2 we can now say that in the classical method method we are in fact using a projection technique.

As each patch is disjoint from every other patch, the basis functions defined in equation 9.2 are orthogonal and

$$
\mathcal{A}_i = \int_{\text{env}} \mathcal{N}_i(\bar{x})\mathcal{N}_i(\bar{x})d\bar{x} = \int_{A_i} d\bar{x} = A_i
$$

where A_i is the area of the i-th patch and the expressions of E_i and $K_{i,j}$ become

$$
A_iE_i = \int_{\text{env}} E(\bar{x})d\bar{x} \;\text{ and }\; A_iA_jK_{i,j} = \int_{A_i}\int_{A_j} K(\bar{x},\bar{y})d\bar{y}d\bar{x}
$$

Substitituing this expression of $K_{i,j}$ in equation 9.12 we will get:

$$
\begin{aligned}
B_i &= E_i + \frac{1}{A_i}\sum_j\left[\int_{A_i}\int_{A_j} K(\bar{x},\bar{y})d\bar{y}d\bar{x}\right]B_j\\
&= E_i + \frac{1}{A_i}\sum_j\left[\int_{A_i}\int_{A_j} \rho(\bar{x})\frac{\cos\theta_{\bar{x}}\cos\theta_{\bar{y}}}{r^2_{\bar{x}\bar{y}}}h(\bar{x},\bar{y})d\bar{y}d\bar{x}\right]B_j.\\
\text{Denoting}\quad F_{i,j} &= \frac{1}{A_i}\int_{A_i}\int_{A_j}\frac{\cos\theta_{\bar{x}}\cos\theta_{\bar{y}}}{r^2_{\bar{x}\bar{y}}}h(\bar{x},\bar{y})d\bar{y}d\bar{x} \;\text{ and assuming } \rho(\bar{x}) \text{ to be constant over patch } i\\
\text{we will get}\quad B_i &= E_i + \rho_i\sum_j F_{i,j}B_j.
\end{aligned}
$$

This last expression is the exact form of the linear radiosity equation used in the classical radiosity method, coefficient B_i in this expression is the radiosity over the patch i and $F_{i,j}$ is the well known form-factor between patch i and patch j. Thus we see that the classical radiosity method is a special case of Galerkin radiosity solution method using piecewise constant basis functions defined in equation 9.13.

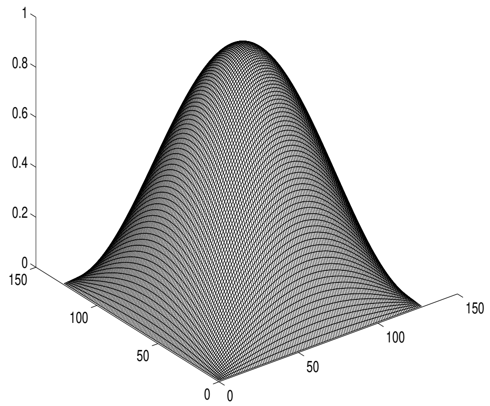

Figure 9.1: Original function.

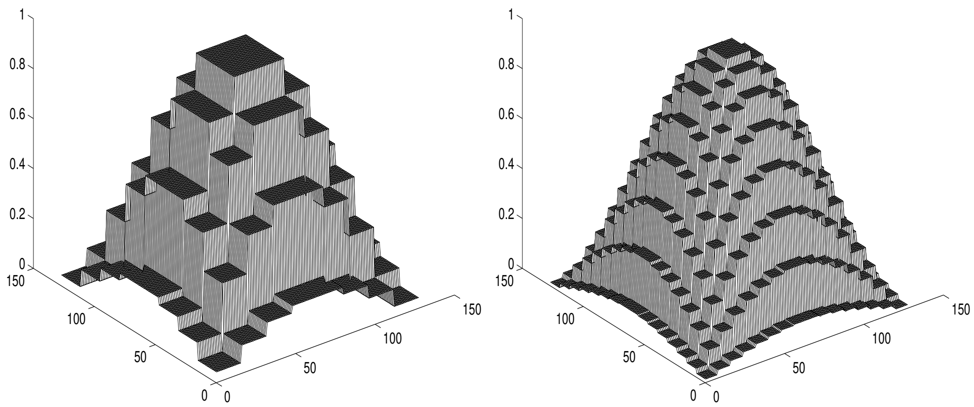

Figure 9.2: Approximation using piecewise constant basis functions.

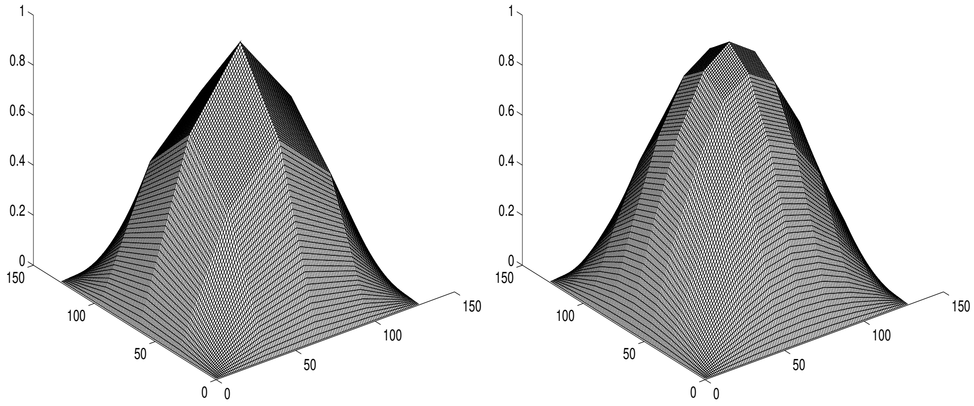

Figure 9.3: Approximation using piecewise linear basis functions.

Constant basis functions are in general not good approximation functions. Figures 9.2 and 9.3 show a comparison of a function approximation using piecewise constant and piecewise linear basis functions. If we compare the approximation resulting from same number of basis functions in the constant and linear case we see that linear basis function gives better approximation. This observation can be generalised and we will say that higher degree polynominal functions are preferable for the solution of radiosity equation.

In the following section we shall discuss general piecewise basis functions and choose a particular basis which will allow us to get non-constant approximation of radiosity function and which will also be amenable to hierarchical algorithm discussed in the earlier chapter to efficiently solve the radiosity equation.

9.4 Wavelet Basis Functions

We discussed in the above section about piece-wise polynomials. They belong to the class of piecewise basis functions. Any linear combination of these functions will generate piecewise polynomial functions. Thus the projection of any function on such basis is a piecewise polynomial.

Wavelets [13] make another important class of piecewise basis functions. Some of the properties of these basis functions are :

1. There are two types of functions in a wavelet. One is the *scaling function*, ϕ, and the other is the *detail function*, ψ. Translates and dilates of these functions make the basis set. By translates and dilates we mean starting with any one of the functions, say ϕ, we can create $\phi_j^{(l)}(t)$ such that

$$\phi_j^{(l)}(t) = 2^{l/2}\phi(2^l t - j + 1)$$

   where j, l are any integer. l is said to define the *level* or *resolution* of the function. Each such function has a compact support which is proportional to the interval $[2^{-l}(j-1), 2^{-l}j[$. With $l \geq 0$ we can create a hierarchy of basis functions such that functions with $l = 0$ are on the root of this hierarchy.

2. The characteristic of ϕ is:

$$\int \phi(t)dt = 1$$

   and the set $\{\phi_j^{(l)}(t) \,|\, j \in \mathcal{Z}\}$ at any l is an orthonormal basis which can be used for approximating any function.

3. The characteristic of ψ is:

$$\int \psi(t)t^{m-1}dt = 0 \quad \text{for } m = 1 \ \ldots \ M \tag{9.14}$$

where M is some positive integer and is said to be the number of *vanishing moments* of the wavelet. $\{\psi_j^{(l)}(t)|j \in \mathcal{Z}\}$ at any level l forms an orthonormal basis. In addition the set $\{\psi_j^{(l)}(t)|j, l \in \mathcal{Z}\}$ is also an orthonormal basis. ψ is orthogonal to ϕ and has the unique property that if the basis set constructed from ϕ is used to approximate a function B then the difference between the approximations at two consecutive levels $(l+1)$ and l is same as the projection of the function on $\{\psi_j^{(l)}(t)\}$ at level l. If we denote the approximation of $B(t)$ in the scaling basis at level l as $\hat{B}^{(l)}(t)$ and the approximation in the detail basis at levels l as $\check{B}^{(l)}(t)$ then

$$\left(\hat{B}^{(l+1)} - \hat{B}^{(l)}\right)(t) = \check{B}^{(l)}(t)$$

4. In the hierarchy described above, there exists an intimate relationship between the basis functions defined at one level with the functions at the adjacent level. The relationship is given as below:

$$\int \phi_j^{(I)}(t)\phi_k^{(I+1)}(t)dt = \begin{cases} c_{k-2j+2} & \text{iff } (1 \leq (k-2j+2) \leq 2M) \\ 0 & \text{otherwise,} \end{cases}$$

The resulting finite set, $\{c_i\}$, has a size $2M$ and is called the discrete filter set. This set is independent of the level of the hierarchy and is the characteristics of a particular wavelet.

We shall discuss now a particular wavelet basis known as *multiwavelets*[1, 22] which is currently popular for radiosity computation.

9.4.1 MultiWavelet Basis

Multiwavelets are a special class of wavelets. The distinguishing feature of this basis is that, unlike the standard wavelets, it has multiple number of ϕ and ψ. If the multiwavelet has M vanishing moments then it has M number of ϕ and ψ functions. If we denote each of these M functions by array indexing mechanism *i.e.* $\phi[1], \phi[2], \ldots, \phi[M]$, then i-th scaling function, $\phi[i]$, for each i, is constructed from the Legendre polynomial of degree $(i-1)$ and each of them is defined over the same parametric support $[0, 1]$. The exact mechanism of construction and a few sample scaling functions are given in figure 9.4. (Note that the scaling functions of multiwavelets with 1 vanishing moment are simple piecewise basis functions.) As for dilation and translation, dilating and translating are carried out on all these M functions together. Thus in multi-wavelet basis sets one will find groups of M functions sharing the same support. This group of M basis function at any level l has the parametric support of $[2^{-l}(j-1), 2^{-l}j[$. The set $\left\{\phi_j^{(l)}[m](t) \mid m = 1 \ldots M \text{ and } j = 1 \ldots 2^l\right\}$ for each $l \geq 0$, is an orthonormal basis for functions with parametric support $[0, 1]$. In addition to this, the set $\left\{\phi_j^{(l)}[m](t) \mid m = 1 \ldots M\right\}$ for every $j = 1 \ldots 2^l$ in any level l is also an orthonormal basis for functions with parametric support $[2^{-l}(j-1), 2^{-l}j[$. Any linear combination of polynomials of degree up to $(M-1)$ is a polynomial of degree up to $(M-1)$. Hence the projection of any function on basis set $\left\{\phi_j^{(l)}[m](t) \mid m = 1 \ldots M\right\}$ is a polynomial of degree up to $(M-1)$ within the parametric support $[2^{-l}(j-1), 2^{-l}j[$.

We shall extend the array indexing mechanism to distinguish the coefficients of projection will be denoted as $B[1] \ldots B[M]$. Example of an hierarchical approximation of $B(t)$, where $t \in [0, 1]$, at different levels of the multiwavelet hierarchy is given in figure 9.5. Because of the presence of M different functions in a group there are M discrete filter sets, one corresponding to each function of the group and each such set has a size

The relationship : $\phi[m](t) = \sqrt{2m+1}\mathcal{P}^m(2t-1)$, where P^m is the Legendre polynomial of $(m-1)$-th degree in the parametric space $[-1,+1]$, is used to create the multiwavelet scaling functions. The resulting construction defines the scaling functions in the parametric space $[0,1]$. Some example functions are given below.

Legendre polynomials up to degree 3.	Scaling functions of Multiwavelet with 4 vanishing moments.
1	1
t	$\sqrt{3}(2t-1)$
$\frac{1}{2}(3t^2-1)$	$\sqrt{5}(6t^2-6t+1)$
$\frac{1}{2}(5t^3-3t)$	$\sqrt{7}(20t^3-30t^2+12t-1)$

Figure 9.4: Expressions for the Multiwavelet scaling functions with vanishing moments 4.

$$
\begin{aligned}
\hat{B}^{(0)}(t) &= \sum_{m=1}^{M} B_1^{(0)}[m]\phi_1^{(0)}[m](t) \\
\hat{B}^{(1)}(t) &= \sum_{m=1}^{M} B_1^{(1)}[m]\phi_1^{(1)}[m](t) + \sum_{m=1}^{M} B_2^{(1)}[m]\phi_2^{(1)}[m](t) \\
\hat{B}^{(2)}(t) &= \sum_{m=1}^{M} B_1^{(2)}[m]\phi_1^{(2)}[m](t) + \sum_{m=1}^{M} B_2^{(2)}[m]\phi_2^{(2)}[m](t) + \\
&\qquad \sum_{m=1}^{M} B_3^{(2)}[m]\phi_3^{(2)}[m](t) + \sum_{m=1}^{M} B_4^{(2)}[m]\phi_4^{(2)}[m](t)
\end{aligned}
$$

At each level, the number of basis functions and so the number of approximation coefficients are twice that of the previous level. The function support is subdivided by a factor of 2 for each increase of the level. At the very first level, *i.e.* $l = 0$, the whole function is approximated by M coefficients and the resulting approximation is a single polynomial piece of degree up to $M-1$. At the next level, the function is approximated by $2M$ coefficients, first M of which are approximating coefficients of the left half of the function and the next M are for the right half of the function. The function in now approximated as 2 pieces of polynominal of degree up to $M-1$. At level l the function is approximated as 2^l polynomial pieces.
Note : The polynomial pieces in the resulting approximation at levels $l > 0$ are not guaranteed to be continuous.

Figure 9.5: Hierarchical approximation of a function using multiwavelet basis.

$2M$.

$$\int \phi_j^{(I)}[m](x)\phi_k^{(I+1)}[l](x)dx = \begin{cases} c_l[m] & \text{iff } (k = 2j-1) \\ c_{(M+l)}[m] & \text{iff } (k = 2j) \\ 0 & \text{otherwise,} \end{cases}$$

The table below gives the filter values corresponding to the multiwavelet with 4 vanishing moments.

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
c[1]	$\frac{1}{\sqrt{2}}$	0	0	0	$\frac{1}{\sqrt{2}}$	0	0	0
c[2]	$-\frac{\sqrt{6}}{4}$	$\frac{\sqrt{2}}{4}$	0	0	$\frac{\sqrt{6}}{4}$	$\frac{\sqrt{2}}{4}$	0	0
c[3]	0	$-\frac{\sqrt{30}}{8}$	$\frac{\sqrt{2}}{8}$	0	0	$\frac{\sqrt{30}}{8}$	$\frac{\sqrt{2}}{8}$	0
c[4]	$\frac{\sqrt{14}}{16}$	$\frac{\sqrt{42}}{16}$	$-\frac{\sqrt{70}}{16}$	$\frac{\sqrt{2}}{16}$	$-\frac{\sqrt{14}}{16}$	$\frac{\sqrt{42}}{16}$	$\frac{\sqrt{70}}{16}$	$\frac{\sqrt{2}}{16}$

9.5 Multiwavelet Radiosity

We can now proceed to use the multiwavelet basis functions for the Galerkin solution of radiosity. Our main aim in using a basis function other than piecewise constants is that we wish to approximate the radiosity function as higher degree polynomials. So we shall use multiwavelets with vanishing moments > 1. We shall discuss below the various issues involved in the use of multiwavelets.

9.5.1 Multiwavelet Functions Defined Over Surface Space

We defined above the multiwavelet scaling functions to be the Legendre polynomials of single parameter t. For the solution of radiosity equation we require a function defined over surface points. To create this type of functions we make assumption that the surfaces in the environments are biparametric. This assumption defines each point in a surface uniquely by two parameters (s, t) where each parameter is defined in a particular range. If we force the parametric range to be $[0, 1]$ then a multiwavelet scaling function Φ over the surface can be constructed as:

$$\Phi[m](\bar{x}) = \phi[m_1](s)\phi[m_2](t) \quad \text{where } s, t \in [0, 1], \quad m_1, m_2 = 1 \dots M, \quad m = 1 \dots M^2 \tag{9.15}$$

Thus there are M^2 scaling functions to start with and hence the hierarchical level $l = 0$ will also have a group of M^2 basis functions and the projection will have M^2 coefficients. The support of each of the functions in the group will be the full surface area. Each dilation will have 2^2 times more basis functions and hence 2^2 times more coefficients than the level above. Also the support of the basis functions at one level will be one fourth the support of the functions above. If we wish to project the radiosity function of a surface p on the basis set at a particular level l of the multiwavelet hierarchy then the resulting approximation will be

$$\hat{B}_{\mathrm{p}}(\bar{x}) = \sum_{j=1}^{2^{2l}} \sum_{m=1}^{M^2} B_{\mathrm{p},j}^{(l)}[m]\Phi_j^{(l)}[m](\bar{x})$$

where $\Phi^{(l)}$ is the scaling function at the hierarchical level l.

As the surface parametrisation of each surface in the environment will be different the construction described above will make the wavelet functions to be unique to each individual surface. In the radiosity equation we have so far assumed radiosity to be globally defined over the whole environment. So the basis functions constructed here cannot be directly used for its solution. To use these basis function, we shall use a reformulation of the radiosity solution. If $\bar{x} = \bar{x}(s, t)$ and $\bar{y} = \bar{y}(u, v)$ in the parametric space $[0, 1] \times [0, 1]$

ξ_i		w_i
	$N = 1$	
0		2
	$N = 2$	
-0.577350269189626		1.0
0.577350269189626		1.0
	$N = 3$	
-0.774596669241483		0.555555555555556
0.0		0.888888888888889
0.774596669241483		0.555555555555556
	$N = 3$	
-0.861136311594053		0.347854845137454
-0.339981043584856		0.652145154862546
0.339981043584856		0.652145154862546
0.861136311594053		0.347854845137454

Table 9.1: Points and weights for the Gauss-Legendre Quadrature.

the following reformulation [52] will define the radiosity function over a surface in terms of radiosity function over another surface.

$$B_{\mathrm{p}}(s,t) = E_{\mathrm{p}}(s,t) + \sum_{\mathrm{q}} \int_{u,v=0,0}^{1,1} K_{\mathrm{p}\leftarrow\mathrm{q}}(s,t,u,v) B_{\mathrm{q}}(u,v)\,dv\,du \tag{9.16}$$

$$\text{where} \quad K_{\mathrm{p}\leftarrow\mathrm{q}}(s,t,u,v) = \rho_{\mathrm{q}}(s,t) \frac{\cos\theta_{\bar{x}} \cos\theta_{\bar{y}}}{r^2_{\bar{x}\bar{y}}} h(\bar{x},\bar{y}) \left\| \frac{\delta \bar{y}(u,v)}{\delta u} \times \frac{\delta \bar{y}(u,v)}{\delta v} \right\| \tag{9.17}$$

where $B_{\mathrm{p}}(.,.)$ and $B_{\mathrm{q}}(.,.)$ are radiosity functions over the surface p and q respectively, $K_{\mathrm{p}\leftarrow\mathrm{q}}$ is the interaction kernel between p and q.
Now to the equation corresponding to each surface we can apply the Galerkin method and generate the set of linear equations. The linear equations resulting from equations of all the surfaces will make the linear system. The projection coefficients of the kernel function in the resulting linear equation will be

$$K_{\mathrm{p}\leftarrow\mathrm{q},i,j} = \int_{s,t=0,0}^{1,1} \int_{u,v=0,0}^{1,1} K_{\mathrm{p}\leftarrow\mathrm{q}}(s,t,u,v) \Phi_i^{(l)}[m](s,t) \Phi_j^{(l)}[m'](u,v)\,dv\,du\,dt\,ds \tag{9.18}$$

Setting up of the linear equations will require the evaluation of these coefficients. We shall discuss one quadrature technique below for such evaluation.

9.5.2 Computation of Inner Products

Evaluations of the inner products of the type given in equation 9.18 require the use of a numerical quadrature technique. Here we shall briefly discuss one such technique. Its based on the following principle:
Given a problem of evaluation of $\int f(s)ds$, choose a set of points $\{\xi_i, i = 1, \ldots, N\}$ and a set of weights $\{w_i, i = 1, \ldots, N\}$ and estimate the inner product as $\sum_{i=1}^{N} w_i f(\xi_i)$.
The points and the weights chosen should be such that the estimate is as accurate as possible for a class of functions $f(x)$. Various methods for making the choice are give in [15]. The simplest and widely used method is Gauss-Legendre quadrature for which the points and weights for different N are given in table 9.1. The integral domain for which the values are applicable is $-1 \ldots +1$. The values have been derived in such a way that the quadrature is exact for an integrand which is a polynomial up to degree $2N-1$. So if

we are calculating an inner product of the type

$$f_i = \int f(t)\mathcal{N}(t)dt,$$

in which $\mathcal{N}(t)$ and $f(t)$ are polynomials of degree m, then the integrand will be a polynomial of degree $2m$. Thus one must choose at least the points and weights corresponding to the $N = m + 1$.

Our particular interest in the evaluation of $K_{\mathrm{p}\leftarrow\mathrm{q},i,j}$ in equation 9.18. In this case the integration is over 4 parametric variables. We can extend the above quadrature of the integrand with one variable. If we are using a Multiwavelet basis of $M = 2$ the Φ has a maximum polynomial degree of 1 and the kernel is almost linear in each variable then our integrand in equation 9.18 becomes a polynomial of degree of 2 in each variable. Thus we must choose the Gauss-legendre points corresponding to $N = 2$, *i.e.* 2 points in each variable and hence 16 evaluations of the kernel. In addition to this one must take care of two sources error in this calculation.

- **Visibility:** The visibility is a part of the kernel function. Because of its boolean nature (zero or nonzero) the evaluation of the kernel just at the Gauss points during quadrature can lead to a very erroneous result. So Gortler *et al* in [22] suggest that, during the quadrature, the kernel be evaluated by completely ignoring the visibility. To this quadrature result, a visibility factor, between 0 and 1, be multiplied. The visibility factor between the two patches involved is independently evaluated (say by the method given in [28]).

- **Singularity:** The kernel is singular at those points where the two surface patches involved touch each other. Thus, if the Gauss points belong to the singularity then we get into a division by zero problem. This division by zero problem is however not serious, as, it can be detected and hence the points can be displaced. The serious problem is that, in the presence of singularity, the approximation of the kernel to the polynomial of certain degree is quite crude. So the Gauss quadrature involving such kernel function will be very erroneous. There is no clear cut solution to this problem. In such situation, one may resort to analytical integration, if possible. Or subdivide the patches containing the singularity to such small pieces that the error resulting in the quadrature does not add much to the overall error.

9.6 Hierarchical Multiwavelet Radiosity

We shall now extend the use of multiwavelets for hierarchical radiosity solution. The hierarchical method proceeds by making a boolean decision on whether a pair of surfaces of the environment can be linked for light exchange. If the decision is negative then one or both of the surface are subdivided and the verification process is continued. The subdivision involved creates a hierarchical representation of the surface. The refinement decision is carried out by an *oracle* process. During the solution step, radiosity is gathered though the links and are collected at various levels of the same surface. For proper representation of the collected radiosity function, a **push/pull** method is utilised.
Thus the minimum requirement for developing a multiwavelet radiosity method will be take care of each of these special features.

From the definition of the multiwavelet basis and function approximation using this basis we have already seen that subdividing a surface patch means associating with each of subdivided patches a group of M multiwavelet scaling functions belonging to a level higher than the current level. All we have to do now is to define an appropriate refinement function (*i.e.* oracle) and to define the **push/pull** functions.

Oracle

In the hierarchical radiosity method (discussed in the previous section) involving constant function we saw an oracle. It was based on the value of form-factor. It allowed a link between two surface patches only when the form-factor between them was below a threshold. The basis behind such choice was that the radiosity function over a receiver surface can be constant only if the kernel function is constant. When a form factor is below a threshold the deviation of the kernel function from a constant function is likely to be very small. Using Multiwavelet we are trying to extend the constant radiosity assumption to higher degree polynomial radiosity assumption. So accordingly, before creating a link between two surface patch we must make sure that the kernel function is of the same polynomial degree. So the key to the oracle will be finding a polynomial fit for the kernel. The pseudo code for the oracle is given below:

```
oracle(patch *p, patch *q)
{
        Evaluate K_{p<-q} at Gauss Points corresponding to N = M;
            /* M is the vanishing moment of the multiwavelet */
        Fit a (M - 1) degree polynomial through the evaluated points;
        Let {calc_i} be the values of the above polynomial at some test points.
        Let {actual_i} be the evaluated K_{p<-q} at the same test points;
        if [Σ abs(actual_i - calc_i) < ε] /* sum is over all the test points*/
                return True;
        else
                return False;
}
```

We can write the refinement algorithm for linking two surfaces as follows:

```
Refine(int l, patch p, patch q)
{
        patch *which;

        if (oracle(p,q) or l = l_max)
                K = quadrature(K(.,.),p,q);
                link(K,p,q);
        else
                which = Subdiv(p,q);
                if(which == q)
                        Refine(l+1,p,q -> nw);
                        Refine(l+1,p,q -> sw);
                        Refine(l+1,p,q -> se);
                        Refine(l+1,p,q -> ne);
                else
                        Refine(l+1,p -> nw,q);
                        Refine(l+1,p -> sw,q);
                        Refine(l+1,p -> se,q);
                        Refine(l+1,p -> ne,q);
}
```

9.6.1 Push and Pull: A Reprojection Mechanism

The gathering of radiosity at any surface in a hierarchical method results in a distribution of radiosity over its various nodes. Radiosity received at nodes higher in the hierarchy must be pushed down and the radiosity received at the nodes lower in the hierarchy must be pulled up. In the Multiwavelet radiosity method radiosity at each node is represented by M^2 coefficients. Pushing this radiosity below the hierarchy will require calculating $4M^2$ coefficients. Similarly pulling the radiosity will involve computing M^2 coefficients from $4M^2$ coefficients. We must develop a mechanism to do that. For that first we shall discuss a general mechanism called reprojection and from that deduce the mechanism for push pull operation.

We shall describe reprojection as a mechanism which is simply the projection of an already projected function on a different basis set. Let us take for example, $\hat{B}$ to be the projection of B over basis $\{\mathcal{N}_i\}$. Let $\{\chi_i\}$ be another basis set and we wish to project $\hat{B}$ on this basis. Using the discussions given above we can write that:

$$
\begin{aligned}
\hat{B}(t) = \sum_{j=1}^{n} B_j\, \mathcal{N}_j(t) \qquad \text{where } B_j &= \int B(t)\mathcal{N}_j(t)dt \\
\check{B}(t) = \sum_{k=1}^{n'} B'_k\, \chi_k(t) \qquad \text{where } B'_k &= \int \hat{B}(t)\chi_k(t)dt \\
&= \int \left[\sum_{j=1}^{n} B_j\, \mathcal{N}_j(t)\right] \chi_k(t)dt = \sum_{j=1}^{n} B_j \left(\int \mathcal{N}_j(t)\chi_k(t)dt\right)
\end{aligned}
$$

If we take a special case of reprojection in which the two basis sets are the hierarchical multiwavelet basis of a single parameter t, in particular the ones belonging to two adjacent levels, *say* $\{\phi^{(I)}\}$ and $\{\phi^{(I+1)}\}$, then because of the the discrete filter functions associated with the multiwavelet we shall get very simple expression for evaluating this reprojection. One of the reprojections will be $Pull$ and the other will be $Push$.
Case I : *Pull* : Reprojection from level $(I+1)$ to level I

$$
\begin{aligned}
B_k^{(I)}[m] &= \sum_{j=1}^{2^{I+1}} \sum_{l=1}^{M} B_j^{(I+1)}[l] \left(\int \phi_k^{(I)}[m](t)\phi_j^{(I+1)}[l](t)dt\right) \\
&= \sum_{l=1}^{M} c_l[m] B_{2k-1}^{(I+1)}[l] + \sum_{l=1}^{M} c_{M+l}[m] B_{2k}^{(I+1)}[l] \qquad (9.19)
\end{aligned}
$$

Case II : *Push* : reprojection from level I to level $(I+1)$

$$
\begin{aligned}
B_{2k-1}^{(I+1)}[m] &= \sum_{j=1}^{2^I} \sum_{l=1}^{M} B_j^{(I)}[l] \left(\int \phi_k^{(I+1)}[m](t)\phi_j^{(I)}[l](t)dt\right) = \sum_{l=1}^{M} c_m[l] B_k^{(I)}[l] \\
\text{and } B_{2k}^{(I+1)}[m] &= \sum_{j=1}^{2^I} \sum_{l=1}^{M} B_j^{(I)}[l] \left(\int \phi_k^{(I+1)}[m](t)\phi_j^{(I)}[l](t)dt\right) = \sum_{l=1}^{M} c_{M+m}[l] B_k^{(I)}[l] \qquad (9.20)
\end{aligned}
$$

where c is the discrete filter function.
We can extend this to the two variable case of radiosity. This extension will give 4 terms in the right hand side of the expression for the *pull* operation (instead of 2 terms in the one variable case in equation 9.19). Each term will be a summation over M^2 coefficients of the node below the hierarchy (instead of M coefficients in the one variable case). The *push* will have 4 expressions. Each expression will be for the coefficients of each of the 4 nodes below the hierarchy and each expression will have only one term which is a summation of M^2 coefficients.

Now we have all the ingredients of a multiwavelet hierarchical algorithm.

Chapter 10

Implementation of one-pass model

The implementation of a global illumination model can be performed according to three approaches: one-pass methods [30, 41, 30, 40, 18], two-pass methods [43, 48], or multi-pass methods [42, 6].

The one-pass methods perform all the illumination computations independently of the view point, allowing then a fast rendering of the same scene from different view points. However, these methods need a large amount of memory to store data. Another drawback is the aliasing defects due to sharp variations of specular reflections and specular transmissions. To avoid these defects, a very dense sampling of the scene is indispensable, which would significantly increase the data to be stored.

In the two-pass methods, the diffuse and specular components (from reflection or transmission) are computed separately; the notion of form-factors are then extended to account for the specular effects contributing to the global diffuse component.

Even though the multi-pass methods are better suited to the rendering of caustic effects, they are very time consuming since they involve several passes: Monte Carlo path tracing, light tracing, progressive refinement radiosity...

In this chapter we will describe a one-pass method for global illumination computation. This method is due to Aupperle and Hanrahan [18]. In the following, we will use the same notations as in [18].

10.1 Formulation

Consider figure 10.1, where A, A' and A'' are surfaces and x, x' and x'' are points on these surfaces respectively.

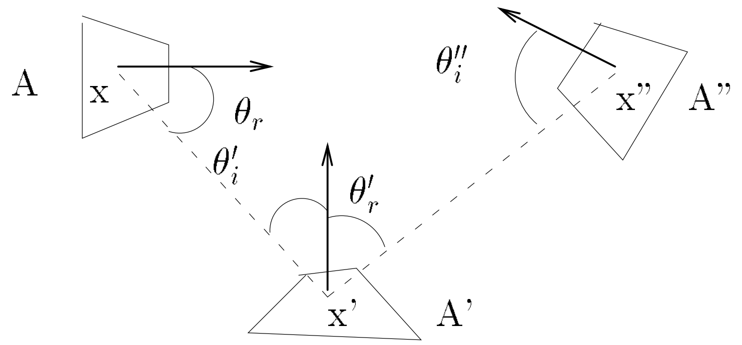

Figure 10.1: Geometry of reflection

The objective is to examine the transport of light incident at A', originating at A and reflected toward A'' by using the radiance equation introduced in a previous chapter.

We can write :

$$L(x', x'') = \int_A f_r(x, x', x'') L(x, x') G(x, x') dx \tag{10.1}$$

where :

$$G(x, x') = \frac{\cos\theta_r \cos\theta_i'}{\mid x - x' \mid^2} h(x, x').$$

The total flux, reflected by a point x' toward surface A'' when x' is illuminated by surface A, is given by :

$$\Phi_{A, dA', A''} = \int_{A''} L(x', x'') G(x', x'') dx''.$$

Using equation 10.1 we get :

$$\int_{A''} L(x', x'') G(x', x'') dx'' = \int_A \int_{A''} f_r(x, x', x'') L(x, x') G(x, x') G(x', x'') dx'' dx.$$

If we integrate the above equation over surface A', we get the total flux reflected by surface A' toward surface A'' when A' is illuminated by A:

$$\int_{A'} \int_{A''} L(x', x'') G(x', x'') dx'' dx' = \int_A \int_{A'} \int_{A''} f_r(x, x', x'') L(x, x') G(x, x') G(x', x'') dx'' dx' dx. \tag{10.2}$$

Now we suppose that the 3 surfaces A, A' and A'' are subdivided into subsurfaces A_i, A_j and A_k respectively, such that $L(x, x')$ and $L(x, x')$ are nearly constant over the pairs of surfaces (A_i, A_j) and (A_j, A_k) respectively.

If we write $L(x, x') = L_{ij} = constant$ and $L(x', x'') = L_{jk} = constant$ and bringing out of the integrals the radiance terms, equation 10.2 becomes:

$$\pi L_{jk} A_j F_{jk} = \pi L_{ij} A_i F_{ij} R_{ijk}, \tag{10.3}$$

where R_{ijk} is defined such that :

$$\pi A_i F_{ij} R_{ijk} = \int_{A_i} \int_{A_j} \int_{A_k} f_r(x, x', x'') G(x, x') G(x', x'') dx'' dx' dx.$$

If we consider all the subsurfaces A_i illuminating surface A_j, then we get:

$$\pi L_{jk} A_j F_{jk} = \sum_i \pi L_{ij} A_i F_{ij} R_{ijk}. \tag{10.4}$$

Due to the symmetry of f_r and $G()$ we have:

$$A_i F_{ij} R_{ijk} = A_k F_{kj} R_{kji}.$$

Thus equation 10.4 becomes:

$$L_{jk} = \sum_i L_{ij} R_{kji}.$$

Taking into account self emission we obtain a discretized form of the radiance equation:

$$L_{jk} = L_{jk}^e + \sum_i L_{ij} R_{kji}. \tag{10.5}$$

10.2 Area Reflectance

10.2.1 Physical meaning

The quantity R_{kji} is no more than the proportion of energy originating at surface A and reflected by surface A' toward surface A''. It is called area reflectance. We can write:

$$R_{kji} = \rho(A_i, A_j, A_k)$$

This is thus a physical quantity which satisfies both the principle of energy conservation over reflection and symmetry property:

- $\sum_k R_{kji} \leq 1$, for fixed i, j,
- $A_i F_{ij} R_{ijk} = A_k F_{kj} R_{kji}$.

10.2.2 Evaluation of the area reflectance

Let us rewrite the expression of R_{kji} as:

$$R_{ijk} = \frac{\int_{A_i}\int_{A_j}\int_{A_k} f_r(x, x', x'')G(x, x')G(x', x'')dx''dx'dx}{\int_{A_i}\int_{A_j} G(x', x'')dx'dx''}.$$

If we assume that A_i, A_j and A_k are small enough that f_r and $G()$ are relatively constant over their surfaces, then:

$$\begin{aligned} R_{kji} &= \frac{S_{kji}G_{kj}G_{ji}A_kA_iA_j}{G_{kj}A_kA_j} \\ &= S_{kji}G_{ji}A_i \end{aligned}$$

where S is the discretized value of f_r such that $S_{kji} = S_{ijk} = S_{x_kx_jx_i}$

The average value of $G(x', x)$ over A_i and A_j is $\pi F_{ij}/A_i$. Thus we can estimate $G_{ji}A_i$ by πF_{ji} and compute R_{kji} as:

$$R_{kji} = \pi F_{ji} S_{kji}.$$

Since it is not possible to compute the exact values of F_{ji} and S_{kji}, they are estimated with error bounds. Let ΔF_{ji} and ΔS_{kji} be the error estimates. Then an estimate for area reflectance is given by:

$$\begin{aligned} R_{kji} &= \pi(F_{ji} + \Delta F_{ji})(S_{kji} + \Delta S_{kji}) \\ &= \pi(F_{ji}S_{kji} + \Delta F_{ji}S_{kji} + \Delta S_{kji}F_{ji} + \Delta F_{ji}\Delta S_{kji}) \\ &\approx \pi(F_{ji}S_{kji} + \Delta F_{ji}S_{kji} + \Delta S_{kji}F_{ji}) \end{aligned}$$

If we neglect the last term, the error estimate for R_{kji} becomes:

$$R_{kji} = \pi(\Delta F_{ji}S_{kji} + \Delta S_{kji}F_{ji}).$$

The accuracy of the estimates for F_{ji} and S_{kji} is dependent on the size of the patches over which reflectance is computed, relative to the distance between them. As the relative size decreases so does error, leading to the adaptive refinement strategy for illumination as shown in the following.

```
typedef struct interaction {
        Patch *from ;
        Patch *to ;
        Color L;
        Color Lg;
        List *gather;
        struct interaction *nw, *sw, *se, *ne;
} Interaction;
```

Figure 10.2: Interaction Data Structure

10.3 Algorithm

10.3.1 Principle

Let us recall the discretized radiance equation:

$$L_{jk} = L^e_{jk} + \sum_i L_{ij} R_{kji}.$$

This is in fact a linear system of equations whose solution can be obtained by gathering. The unknowns are the radiances L_{ji} with $i, j \in [1, N]$, N being the number of patches.

As all illumination is expressed as the radiance at a given patch toward another, the patch-patch interactions form the primary data structure necessary for the system solution. All operations will be over interactions. A data structure representing an interaction is given in figure 10.2.

A given interaction *ij* is defined by two patches $ij \rightarrow from$ and $ij \rightarrow to$ and represents the radiance at *from* toward *to*. This radiance is stored within the interaction as attribute *L*. *Lg* is the radiance gathered during the current iteration from interactions contained in the list *gather*. Subinteractions *nw, sw, se, ne* are the children of *ij* induced by the subdivision of *from* and *to*.

In the next subsection we will give the algorithm for the refinement and computation of illumination over a hierarchy of interactions. The algorithm refines pairs of interactions (*ij*, *jk*) such that $ij \rightarrow from = ij \rightarrow to$ to ensure the computed reflectance across the interaction pairs and associated patch triples satisfy user specified error bounds. If a given interaction pair (*ij*, *jk*) is satisfactory, the interactions are linked to record that radiance may be gathered from *ij* to *jk*, otherwise one of both interactions are subdivided and refinement is applied to their children.

After refinement, a gathering iteration may be carried out, each interaction gathering radiance from interactions to which it has been linked. The gathered radiances are then distributed within each receiving interaction hierarchy, and subsequent iterations are computed till convergence is met.

Regarding the eye, it may considered as a small patch interacting with the other patches. This special patch will have not to reflect or to be responsible for any occlusion.

10.3.2 Adaptive Refinement

The adaptive refinement is carried out as shown in figure 10.3.

The Refine() procedure computes pairs of interactions by subdividing and recursively refining if estimated errors exceeds specified error bounds, linking the interactions for gathering if the bounds are satisfied or no subdivision is possible.

F_ϵ and S_ϵ are the bounds for geometric and reflection error respectively. A_ϵ is the minimum area for a patch. GeometryErrorEstimate() and ReflectionErrorEstimate() provide estimations for $\pi\Delta F_{ji} S_{kji}$ and $\pi\Delta S_{kji} F_{ji}$.

Refine(Interaction $*ij$, Interaction $*jk$,
float F_ϵ, float S_ϵ, float A_ϵ)
{
float f_ϵ, s_ϵ;

f_ϵ = GeometryErrorEstimate(ij);
s_ϵ = ReflectionErrorEstimate(ij,jk);
if ($f_\epsilon < F_\epsilon$ && $s_\epsilon < S_\epsilon$)
Link(ij,jk);
else if ($s_\epsilon \geq S_\epsilon$)
switch(SubdivS(ij, jk, A_ϵ)){
case PATCH_I :
Refine($ij \rightarrow nw$, jk, S_ϵ, F_ϵ, A_ϵ);
Refine($ij \rightarrow sw$, jk, S_ϵ, F_ϵ, A_ϵ);
Refine($ij \rightarrow se$, jk, S_ϵ, F_ϵ, A_ϵ);
Refine($ij \rightarrow ne$, jk, S_ϵ, F_ϵ, A_ϵ);
case PATCH_J :
/* refine over children of ij and jk */
case PATCH_K :
/* refine over children of jk */
case NONE :
Link(ij, jk);
}
else /* $f_\epsilon \geq F_\epsilon$ */
switch(SubdivG(ij, jk, A_ϵ)){
/* refine over children, or link as */
/* directed by PATCH_I, J, K or NONE. */
}
}

Figure 10.3: **Refine** pseudocode.

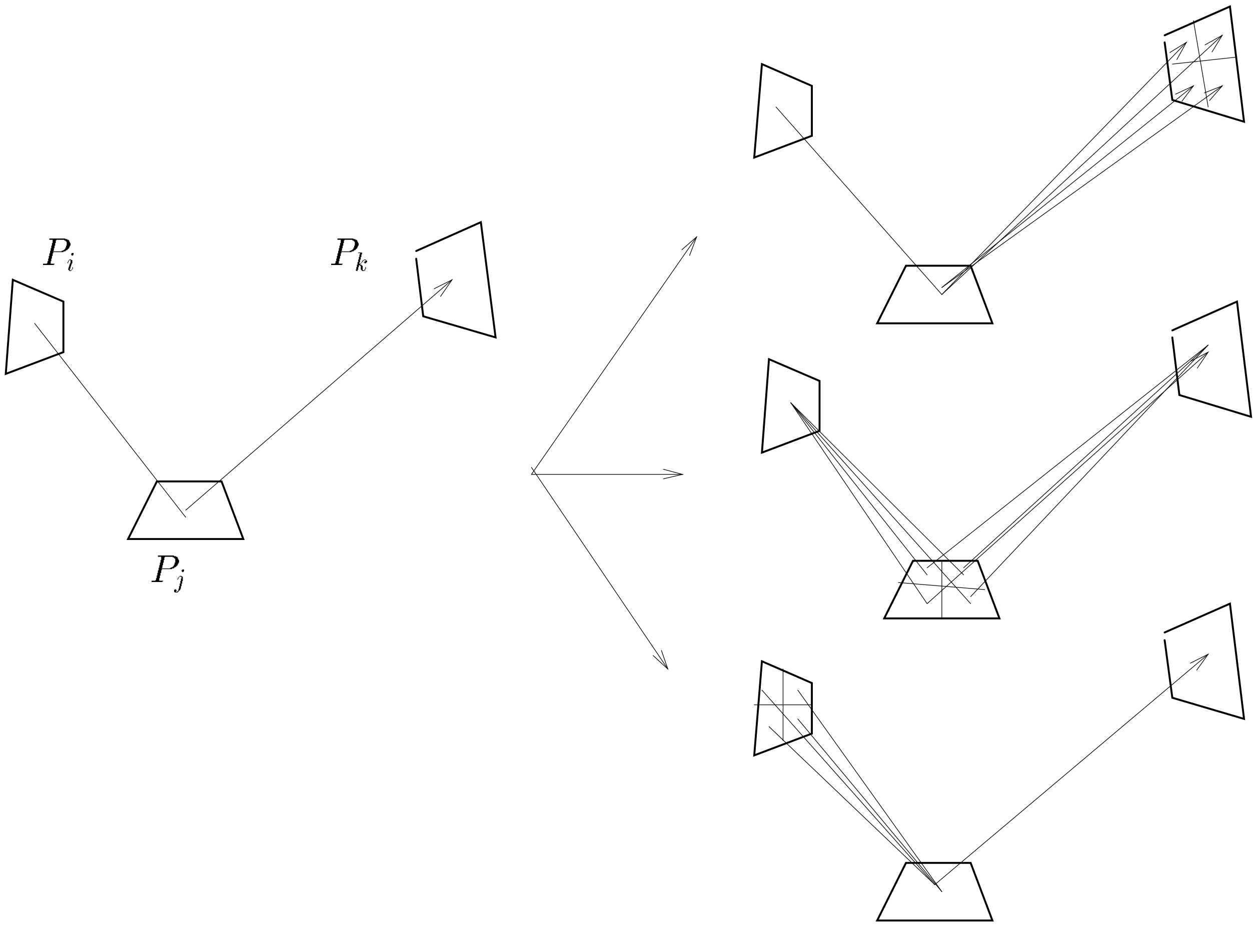

Figure 10.4: Refinement and subdivision

SubdivS() and SubdivG() control refinement for reflection and geometry error respectively. Both routines select a patch for refinement, subdividing the patch and associated interactions if required. An identifier for the selected patch is returned. If no patch may be subdivided, then none is passed back. Note that a given interaction/patch may be refined against many different interactions within the system, and thus may have already been subdivided when selected by a Subdiv procedure. In this case, the procedure simply returns the proper identifier.

The Subdiv procedures (S and G) should select for refinement patches of large size relative to their distance in the surface triple. Form factor estimation is a convenient criterion for the determination of such patches. A large differential area-area form factor $F_{dp,q}$ indicates that patch q is of large relative size.

The Subdiv procedures thus choose for refinement the patch of size at least A_ϵ that is of greatest form factor within ij and/or jk that will not introduce multiple sets of children over either interaction. If patch p_j is of greatest form factor over both ij and jk, and of area greater than A_ϵ, then it is chosen for refinement (figure 10.4 at middle). Otherwise, if p_j is selected over one interaction but p_i or p_k is selected over the other, then the *outside* patch is chosen for refinement. Given two selected *outside* patches, SubdivS() selects the one of greater form factor relative to p_j, while SubdivG() selects p_i over p_k as p_k has no direct effect on geometry accuracy. Note however that even under SubdivG(), if only p_j and p_k are allowed subdivision, p_k will be selected.

```
Gather(Interaction *jk)
{
        Interaction *ij;
        if(jk)
                jk → L_g = 0;
                for(all elements (ij,jk → gather))
                        jk → L_g += ij → L * ρ(ij, jk);
                Gather(jk → nw);
                Gather(jk → sw);
                Gather(jk → se);
                Gather(jk → ne);
}
```

Figure 10.5: **Gather** pseudocode.

10.3.3 Gathering

The gathering algorithm is given in figure 10.5. This algorithm gathers radiance into $jk \rightarrow Lg$ rather than directly into $jk \rightarrow L$ to avoid a push/pull operation with every invocation of the Gather() procedure. Jacobi's resolution method is used by this procedure.

Bibliography

[1] B. Alpert. A class of bases in l^2 for the sparse representation of integral operators. *SIAM Journal on Mathematical Analysis*, 24(1), 1993.

[2] III A.T. Campbell and Donald S. Fussell. *Analytic Illumination with Polygonal Light Sources.* Technical Report Technical Report TR-91-15, Dept. of Computer Sciences, Univ. of Texas at Austin, April 1991.

[3] M. Born and E. Wolf. *Principles of optics.* Pergamon press, 1970.

[4] C. Bouville, J. L. Dubois, I. Marchal, and M. L. Viaud. Monte-carlo method applied to an illumination model. In *EUROGRAPHICS'88 Conference Proceedings*, pages 483–497, September 1988.

[5] E. Catmull. A tutorial on compensation tables. *Computer Graphics*, 1979.

[6] S. E. Chen, H. E. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. *Computer Graphics*, 25(4):165–174, August 1991.

[7] S. Clavé and M. Groß. A rendering pipeline for street lighting simulation. In *Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 13-15 1991.

[8] M. Cohen and D. Greenberg. The hemi-cube, a radiosity solution for complex environments. *Computer graphics*, 19(3), 1985.

[9] M. F. Cohen, D. P. Greenberg, D. D. Immel, and P. J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(2):26–35, March 1986.

[10] Michael F. Cohen, Shenchang E. Chen, John R. Wallace, and Donald P. Greenberg. A progessive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988.

[11] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics*, 18(3):165–174, July 1984.

[12] R.L. Cook and K.E. Torrance. A reflectance model for computer graphics. *ACM transactions on graphics*, 1(1):7–24, January 1982.

[13] Ingrid Daubechies. *Ten Lecture on Wavelets.* SIAM, 1992.

[14] Commission Internationale de L'Eclairage. Informal report on the standardisation of luminance distribution of clear skies. 1973.

[15] L. M. Delves and J. L. Mohamed. *Computational Methods for Integral Equations.* Cambridge University Press, 1985.

[16] Maria Lurdes Dias. Ray tracing interference color. *IEEE Computer Graphics and Applications*, 54–60, March 1991.

[17] A. Dubec et L. Goussot. *Télévision en couleur, Photométrie, Colorimétrie, Le tube image.* Technical Report 1.80, Radiodiffusion-Télévision.

[18] A Hierarchical Illumination Algorithm for Surfaces with Glowwy Relfection. A hierarchical illumination algorithm for surfaces with glossy reflection. In *Proceedings of SIGGRAPH annual conference 1993*, pages 155–162, 1993.

[19] M. Garcia, M. Perraudeau, and P. Chauvel. Experimental measurements of reflectance and transmittance with a spectrophotometer. February 1992. CSTB, private communication.

[20] Reid Gershbein, Peter Schroder, and Pat Hanrahan. Texture and radiosity: controlling emission and reflection with texture maps. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28:51–58, 1994.

[21] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):213–222, July 1984.

[22] Steven Gortler, Peter Schroder, Michel F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):221–230, 1993.

[23] A. Roy Hall and Donald P. Greenberg. A testbed for realistic image synthesis. *IEEE Computer Graphics and Applications*, 3(8):10–20, November 1983.

[24] Roy A. Hall. In *Illumination and Color in Computer Generated Imagery*, Springer-Verlag, New York, 1989.

[25] P. Hanrahan and D. Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. In *Photorealism in Computer Graphics*, pages 151–169, EurographicSeminars, Springer-Verlag, 1992.

[26] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991.

[27] Pat Hanrahan and David Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, 151–171, June 1990.

[28] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.

[29] X. D. He, K. E. Torrance, F. X. Sillon, and D. P. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics*, 25(4):175–186, August 1991.

[30] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffused environments. *Computer Graphics*, 20(4):133–142, July 1986.

[31] K. Kaneda, T. Okamoto, E. Nakamae, and T. Nishita. Photorealistic image synthesis for outdoor scenery under various atmospheric conditions. *The Visual Computer*, 7:247–257, 1991.

[32] Eric Languénou and Pierre Tellier. Including physical light sources and daylight in a global illumination model. In *Proceedings of the Third Eurographics Workshop on Rendering*, Bristol, UK, May 1992.

[33] Dani Lischinski, Brian Smits, and Don Greenberg. Bounds and error estimates for radiosity. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):67–74, July 1994.

[34] G. Meyer, W. Cowan, M. Stone, and B. Wandell. ACM SIGGRAPH'89 Course Notes. In *Introduction to practical issues in color reproduction and selection*, August 1989.

[35] Gary W. Meyer. Wavelength selection for synthetic image generation. *Computer Vision, Graphics, and Image Processing*, 41:57–79, 1988.

[36] Gary W. Meyer and P. Greenberg. *Colorimetry and Computer Graphics.* Technical Report, Cornell University, June 1983.

[37] M.F. Cohen and J.R. Wallace. Academic Press Professional, 1993.

[38] L. Neumann and A. Neumann. *PHOTOSIMULATION : Interreflection with Arbitrary Reflectance Functions and Illuminations.* Internal report, Software Development, MIKROKER Cooperative, Budapest, Hungary, 1988.

[39] H. Rushmeier and K. Torrance. Extending the radiosity method to include reflecting and translucent materials. *ACM Transaction on Graphics*, 9(1):1–27, January 1990.

[40] B. Le Saec and C. Schlick. A progressive ray-tracing-based radiosity with general reflectance functions. In *PhotoRealism in Computer Graphics*, pages 101–113, EurographicSeminars, Springer-Verlag, 1992.

[41] M. Shao, Q. Peng, and Y. Liang. A new radiosity approach by procedural refinements for realistic image synthesis. *Computer Graphics*, 22(4):93–101, August 1988.

[42] P. Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Graphis Interface Conference Proceedings*, pages 205–212, May 1990.

[43] F. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics*, 23(3):335–344, July 1989.

[44] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance driven radiosity algorithm. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):273–282, July 1992.

[45] S. Spencer. The hemisphere radiosity method: a tale of two algorithms. In *Photorealism in Computer Graphics*, pages 127–135, Springer-Verlag, EurographicSeminars, 1992.

[46] Jack Tumblin and Holly E. Rushmeier. *Tone Reproduction for Realistic Computer Generated Images.* Tech. Report GI -GVU-91-13, Graphics, Visualization & Usability Center, Coll. of Computing, Georgia Institute of Tech., 1991.

[47] J. R. Wallace, M. F. Cohen, and D. P. Greenberg. A two-pass solution to the rendering equation: a synthesis of ray tracing and radiosity methods. *Computer Graphics*, 21(4):311–320, August 1987.

[48] J. R. Wallace, K. A. Elmquist, and E. A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):315–324, July 1989.

[49] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, July 1992.

[50] R.C. Weast, D.R Lide, M.J. Astle, and W.H. Beyer. *CRC Handbook of Chemistry and Physics.* CRC Press, Inc. Boca Raton, Florida, 1989-1990.

[51] G. Wyszecky and W. S. Stiles. *Color Science, Concepts and Methods, Quantitative Datas and Formulas.* J. Willey and sons, 1982. 2nd Edition.

[52] Harold R. Zatz. Galerkin radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):213–220, 1993.

[53] O. C. Zienkiewicz and K. Morgan. *Finite Elements and Approximation.* McGraw-Hill Book Company.